

The Journal of Computing Sciences in Colleges

**Papers of the 30th Annual CCSC
NorthEastern Conference**

April 10th and April 11th, 2026
Smith College
Northampton, MA

Abbas Attarwala, Editor
California State University, Chico

Jeremiah Johnson, Regional Editor
University of New Hampshire

Volume 41, Number 8

April 2026

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors	9
CCSC National Partners	11
Welcome to the 2026 CCSC NorthEastern Conference	12
Regional Committees — 2026 CCSC NorthEastern Region	13
Ethics-Driven Computing Education Through Experiential Learning Labs	14
<i>Ursula Parker, Rochester Institute of Technology; Farzana Rahman, Syracuse University; Samuel Malachowsky, Rochester Institute of Technology; Daniel Krutz, Rochester Institute of Technology</i>	
Hands On Data Structures: Making Abstraction Tangible with Manipulatives	18
<i>Halie M. Rando, Smith College; Jade Lilian Palosky, Smith College; Hayley Ma, Smith College; Vivian Myers, Smith College; Sofia Jurusz, Smith College</i>	
Next-Gen Teaching: Harnessing AI in Course Design	21
<i>Patrick Ester, Codio, Inc.; Mohit Chandarana, Codio, Inc.</i>	
PLCC: A Tool Set for Teaching Programming Languages Courses	23
<i>Timothy Fossum, SUNY Potsdam; James Heliotis, Rochester Institute of Technology; Stoney Jackson, Western New England University; Alain Kägi, Lewis Clark College</i>	
Amp Stack: A System for Teaching CS1 Modularity	25
<i>Angel Rivera, Utica University</i>	
A Summer Curriculum in Data Visualization and Analysis for High School Students	27
<i>Paolien Wang, Kean University; Ching-yu (Austin) Huang, Kean University</i>	
Creating a Web Application to Scaffold Student Practice Work for a Discrete Mathematics Course	29
<i>Christopher M. Morales, SUNY Plattsburgh</i>	

Common Student Mistakes in Building Data Marts for Effective Data Visualization	31
<i>Ching-yu (Austin) Huang, Kean University</i>	
Growing STEM Student Success with Interdisciplinary Data Science Courses	33
<i>Beryl Hoffman, Elms College; Nina Theis, Elms College; Stephen Walsh, Elms College</i>	
Bridging the Gap with AI Literacy	35
<i>Carolyn C. Matheus, Marist University; Kevin M. Gaugler, Marist University</i>	
Ransom Sentinel (RaSe): Designing Ransomware-Resilient Systems Through Adversarial Simulation and Cryptography	37
<i>Chetan Jaiswal, Quinnipiac University</i>	
Rediscovering Collaborative Whiteboarding for Problem Solving in the Era of AI	39
<i>Karen H. Jin, University of New Hampshire</i>	
Essential Computing Concepts (Draft): An Alternative to CS2023 for Colleges	41
<i>Alyce Brady, Kalamazoo College; Megan Olsen, Loyola University Maryland; Rick Blumenthal, Regis University; Grant Braught, Dickinson College; Janet Davis, Whitman College; Bruce A. Maxwell, Northeastern University; Karl R. B. Schmitt, Trinity Christian College; Andrea Tartaro, Furman University; Henry Walker, Grinnell College; Amanda M. Holland-Minkley, Washington & Jefferson College</i>	
Designing Interdisciplinary Neuro-divergent Inclusive CUREs in Computer Science	44
<i>Karina Assiter, Landmark College; Sue Vincent, Landmark College</i>	
Toward True Differentiated Instruction with PathMX and Curriculum as Code	46
<i>Mark Johnson, Furman University; Chris Alvin, Furman University</i>	
Apache Sudoku: A Web-Based Game for Learning and Demonstrating Web Development Concepts	61
<i>Colin Zouck, Utica University; Unnati Shah, Utica University</i>	

Eye-Assist: An AI Visual Navigation System as a Multi-Disciplinary Computing Education Platform	69
<i>Nicholas Brown, Quinnipiac University; Kruti Shah, Quinnipiac University</i>	
A Model for AI Across the Curriculum at an Urban Community College	77
<i>Mohammad Q. Azhar, Borough of Manhattan Community College; Rifat Salam, Borough of Manhattan Community College; Elizabeth Wissinger, Borough of Manhattan Community College; Melissa Zavala, Borough of Manhattan Community College</i>	
Experiential Educational Ethics Activities in Undergraduate Education: Instructor Observations	90
<i>Ursula Parker, Rochester Institute of Technology; Farzana Rahman, Syracuse University; Ashish Aggarwal, University of Florida; Samuel Malachowsky, Rochester Institute of Technology; Daniel Krutz, Rochester Institute of Technology</i>	
Supporting High School Computing Education with Easily Adoptable Experiential Learning Modules	105
<i>Ursula Parker, Rochester Institute of Technology; Farzana Rahman, Syracuse University; Cathleen Cerosaletti, University of Rochester; Zenon Borys, University of Rochester; Samuel Malachowsky, Rochester Institute of Technology; Daniel Krutz, Rochester Institute of Technology</i>	
Teaching Computer Security Through Lab Demonstrations and Simulators Created as Senior Programming Projects	120
<i>Edwin Dauber, Widener University</i>	
Detection of Spinning Behavior with a Known Solution	135
<i>Mark Sherman, Emmanuel College; Mohit Chandarana, Codio, Inc.; Erica Hinkle, Emmanuel College</i>	
Teaching Algorithmic Bias	150
<i>Kevin McCullen, SUNY Plattsburgh</i>	
Reverse Engineering Student Misconceptions	162
<i>Anthony Notaro, United States Military Academy; Geoffrey Moores, United States Military Academy; Maria R. Ebling, United States Military Academy</i>	

Integrating COIL into an Undergraduate Software Engineering Course: A Cross-Cultural Experience Report	177
<i>Olivera Grujic, California State University, Stanislaus</i>	
Integrating Smart Learning Content with Project-Based Introductory Programming Course at Community Colleges	191
<i>Mohammad Hassany, University of Pittsburgh; Can Kultur, Carnegie Mellon University; Rafaella Sampaio de Alencar, University of Pittsburgh; Kamil Akhuseyinoglu, Carnegie Mellon University and University of Pittsburgh; Christopher Bogart, Carnegie Mellon University; Jaromir Savelka, Carnegie Mellon University; Peter Brusilovsky, University of Pittsburgh</i>	
AI-Enhanced Guided-Inquiry Platform for Collaborative Learning in Introductory Programming	206
<i>Bo Kim, Southern New Hampshire University; James Skon, Kenyon College</i>	
Teaching Programming at a Small College in the Era of GenAI	220
<i>Christine F. Reilly, Skidmore College</i>	
ConnectPlus: A Demonstration of a Full-Stack Collaboration Tool for Academic and Team Projects	233
<i>Makayla Coleman, Utica University; Unnati Shah, Utica University</i>	
Structured Post-Evaluation Interviews and Remediation (SPEIR): A Formative Assessment Workflow	243
<i>Pablo Frank-Bolton, Smith College; Michael Robson, Smith College; R. Jordan Crouser, Smith College; Rahul Simha, George Washington University</i>	
Teaching Software Development in the GenAI Era: Experiences and Course Design	258
<i>Weidong Liao, Shepherd University</i>	
ReRead Text Pattern Editor for Open-Ended and Coding Questions	273
<i>Vladislav D. Veksler, Caldwell University</i>	
Teaching AI the Learner-Centered Way: Designing a Meaningful AI Literacy Lesson Model for Middle School	281
<i>Yewon Lee, Mount Holyoke College; Hia Ghosh, Smith College</i>	

The Consortium for Computing Sciences in Colleges

Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Bryan Dixon, President (2026), bcdixon@csuchico.edu, Computer Science Department, California State University Chico, Chico, CA 95929.

Shereen Khoja, Vice President/President-Elect (2026), shereen@pacificu.edu, Computer Science, Pacific University, Forest Grove, OR 97116.

Abbas Attarwala, Publications Chair (2027), aattarwala@csuchico.edu, Department of Computer Science, California State University Chico, Chico, CA 95929.

Ed Lindoo, Treasurer (2026), elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Haiyan Cheng, Membership Secretary (2028), hcheng@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.

Judy Mullins, Central Plains Representative (2026), mullinsj@umkc.edu, University of Missouri-Kansas City, Kansas City, MO (retired).

Michael Flinn, Eastern Representative (2026), mflinn@frostburg.edu, Department of Computer Science & Information Technologies, Frostburg State University, Frostburg, MD 21532.

Gabe Ferrer, Midsouth Representative (2028), ferrer@hendrix.edu, Department

of Computer Science, Hendrix College, Conway, AR 72032.

David Largent, Midwest Representative (2026), dllargent@bsu.edu, Department of Computer Science, Ball State University, Muncie, IN 47306.

Michael Gousie, Northeastern Representative (2028), gousie_mike@wheatoncollege.edu, Computer Science Department, Wheaton College, Norton, MA 02766.

Ben Tribelhorn, Northwestern Representative (2027), tribelhb@up.edu, School of Engineering, University of Portland, Portland, OR 97203.

Mohamed Lotfy, Rocky Mountain Representative (2028), MohamedL@uvu.edu, Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Mika Morgan, South Central Representative (2027), mika.morgan@msutexas.edu, Department of Computer Science, Midwestern State University, Wichita Falls, TX 76308.

Karen Works, Southeastern Representative (2027), keworks@pc.fsu.edu, Department of Computer Science, Florida State University Panama City, Panama City, FL 32405.

Michael Shindler, Southwestern Representative (2026), mikes@uci.edu, Computer Science Department, UC Irvine, Irvine, CA 92697.

Serving the CCSC: These members are serving in positions as indicated:

Bin Peng, Associate Editor, bin.peng@park.edu, Computer Science and Information Systems, Park University, Parkville, MO 64152.

Lucas Cordova, Associate Treasurer, lpcordova@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.

George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood College, Frederick, MD 21701.

Megan Thomas, Membership System

Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.

Karina Assiter, National Partners Chair, KarinaAssiter@landmark.edu, Landmark College, Putney, VT 05346.

Ed Lindoo, UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Platinum Level Partner

College Board

Gold Level Partner

Rephactor

ACM2Y

Blossoms

MAP-CS: Mission-Aligned Programs in Computer Science

CCSCNE 2026 Chairs' Welcome

Welcome to Northampton, Massachusetts, and Smith College, for the Thirtieth Annual Consortium for Computing Sciences in Colleges Northeast Region Conference.

Our program features two distinguished invited speakers, Mark Guzdial of the University of Michigan and Fran Berman of UMass Amherst. In addition, we have a broad range of topics covered by paper presentations, lightning talks, workshops, tutorials, and faculty and student research posters. On Friday morning, we are hosting the traditional Frank Ford Memorial Programming Contest.

This conference would not happen without the efforts of the outstanding conference committee and dedicated board. The success of this conference is the result of the hard work of those groups, plus the reviewers, session chairs, and countless other volunteers. The conference continues to be selective; we accepted 19 of 30 papers for an acceptance rate just over 63%. This ensures the high-quality program that everyone expects.

We also want to thank the National Partners whose support helps keep the conference affordable: The College Board, acm2y, Blossoms, MAP-CS, and Rephactor, and our local partner MongoDB. Additional support is provided to the conference for being in cooperation with the ACM Special Interest Group on Computer Science Education (SIGCSE).

We are pleased to be hosting this year's conference for the first time at Smith College, on its beautiful campus in the Pioneer Valley. We hope you find the conference informative and engaging. We also look forward to seeing you next year at Southern New Hampshire University in Manchester, New Hampshire.

R. Jordan Crouser
Conference Co-Chair
Smith College and Kenyon College

Adita Kulkarni
Conference Co-Chair
SUNY Brockport

2026 CCSC Northeast Conference Committee

Adita Kulkarni, Conference Co-Chair	SUNY Brockport
Jim Teresco, Conference Co-Chair	Siena College
Mark Bailey, Program Chair	Hamilton College
Yana Kortsarts, Papers Co-Chair	Widener University
Chetan Jaiswal, Papers Co-Chair	Quinnipiac University
Jordan Crouser, Panels, Tutorials, and Workshops Co-Chair ..	Smith College
Bo Kim, Panels, Tutorials, and Workshops Co-Chair	Southern New Hampshire University
Dan Rogers, Faculty Posters and Lightning Talks Co-Chair	SUNY Brockport
Joan DeBello, Faculty Posters and Lightning Talks Co-Chair	St. John's University
Ali Erkan, Faculty Cohorts Co-Chair	Ithaca College
Mike Gousie, Faculty Cohorts Co-Chair ...	Wheaton College (Massachusetts)
Larry D'Antonio, Speakers Chair	Ramapo College of New Jersey
Sandeep Mitra, Undergraduate Posters Co-Chair	SUNY Brockport
Karyn Doke, Undergraduate Posters Co-Chair	Hamilton College
Liberty Page, Undergraduate Posters Co-Chair	University of New Haven
Drew van der Poel, Undergraduate Posters Co-Chair	Northeastern University
Rick Kline (Emeritus), Registration Chair	Pace University
Del Hart, Programming Contest Co-Chair	SUNY Plattsburgh
Adita Kulkarni, Programming Contest Co-Chair	SUNY Brockport
Darren Lim, Programming Contest Co-Chair	Siena College
Paul Olsen, Programming Contest Co-Chair	Le Moyne College
Jim Teresco, Vendors and Partners Chair	Siena College
Sandeep Mitra, Local Arrangements Co-Chair	SUNY Brockport
Evgeniya Iskrenova-Ekiert, Local Arrangements Co-Chair ..	SUNY Brockport
Ning Yu, Local Arrangements Co-Chair	SUNY Brockport

Ethics-Driven Computing Education Through Experiential Learning Labs

Conference Tutorial

Ursula Parker¹, Farzana Rahman²,
Samuel Malachowsky¹, Daniel Krutz¹

¹Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623

{ucp4496,samvse,dxkvse}@rit.edu

²College of Engineering and Computer Science
Syracuse University
Syracuse, NY 13244

frahma02@syr.edu

Abstract

Our Accessible Learning Labs initiative helps participants learn how to design ethical software while introducing them to essential concepts in Artificial Intelligence and Machine Learning (AI/ML). Several of these developed labs focus on foundational ethics-focused topics. These experiential interactive modules highlight why ethics in AI/ML matters and provide practical experiences that reveal the diverse ways ethics-focused topics can influence modern systems. The tutorial is suitable for a broad audience within the software engineering community—from students to seasoned professionals—who wish to better understand ethical implications across domains and ensure that the software they develop is ethical and fair. Complete project materials are openly available on our website: <https://all.rit.edu>

1 Introduction

To address the gap in computing ethics education, we have developed a robust suite of experiential laboratory activities that form a core foundation for teaching ethical concepts, specifically related to ethics in Artificial Intelligence (AI)

and Machine Learning (ML). In addition, to broaden understanding of ethics-focused topics, we created several educational labs that highlight the broader impact of ethics across multiple domains. Because these labs are self-contained and easy to adopt, they can be seamlessly integrated into many introductory computing courses (*e.g.*, CS I & II) [4, 2, 5], which have been demonstrated in prior studies [1, 3, 5]. This tutorial will enable participants to:

1. Explore our ethics-focused and AI/ML educational materials,
2. Offer feedback and recommendations for improving the labs, and
3. Exchange common challenges and effective strategies for incorporating ethics into computing curricula. No prior experience is required.

Each ethics-focused lab focuses on a specific ethical issue and includes the following components: I) Essential background information, II) A sample application that illustrates the ethical concern, III) A guided procedure that simulates the problem, IV) additional context highlighting real-world experiences with biased software, and V) A quiz to assess the participant's understanding.

2 Tutorial Session Agenda

Activity 1: Ethics of AI-focused Lab: (20 minutes) This lab introduces participants on the ethics behind the implementation of Artificial Intelligence models. This lab guides users through multiple scenarios where the AI is biased.

Activity 2: Algorithmic Bias: (20 minutes) This lab teaches students about the importance of mitigating machine learning bias as well as the ethical guidelines that developers should follow when building ML software.

Activity 3: Bias in Training Sets: (20 minutes) This lab teaches participant about where bias is present within neural networks and ways to help reduce the biases developed in the algorithms.

Activity 4: Lab Feedback: (15 minutes) Participants will share feedback on the materials and suggest ways the labs could evolve. Their input will directly inform and refine the design of future lab modules.

Acknowledgements

This material is based upon work supported by the United States National Science Foundation under grants #2111152, #2145010, #2336941, #2524374.

Presenter Biographies

Ursula Parker is a 2nd-year undergraduate student studying Software Engineering. She has contributed to the ALL project as an Outreach specialist.

Samuel Malachowsky is the Co-PI of the project that is developing the described accessibility labs. Malachowsky is a Principal Lecturer at RIT and has authored 9 pedagogically focused publications. He holds a Project Management (PMP) certification and has authored a textbook on team leadership.

Farzana Rahman is the Co-PI of the project that is developing the described accessibility labs. Malachowsky is a Sr. Lecturer at RIT and has authored 9 pedagogically focused publications. He holds a Project Management (PMP) certification and has authored a textbook on team leadership.

Daniel Krutz is the PI of the NSF-funded projects (#1825023, #2111152 and #2145010) that are devoted to creating the presented labs. Krutz is an Associate Professor in the Department of Software Engineering at The Rochester Institute of Technology. Krutz has taught approximately ten different graduate and undergraduate software engineering courses and is the author of over twelve pedagogical research papers.

References

- [1] Saad Khan et al. “Experiential Learning in Undergraduate Accessibility Education: Instructor Observations”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 54–68.
- [2] Yang Liu et al. “Do Users Act Equitably? Understanding User Bias Through a Large In-person Study”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 2023, pp. 83–95. DOI: 10.1109/ICSE-SEIS58686.2023.00014.
- [3] Weishi Shi et al. “ALL: Supporting Experiential Accessibility Education and Inclusive Software Development”. In: *ACM Trans. Softw. Eng. Methodol.* (Sept. 2023). ISSN: 1049-331X. DOI: 10.1145/3625292. URL: <https://doi.org/10.1145/3625292>.
- [4] Weishi Shi et al. “Experiential Learning in Computing Accessibility Education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. ICSE ’20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 250–251. ISBN: 9781450371223. DOI: 10.1145/3377812.3390901. URL: <https://doi.org/10.1145/3377812.3390901>.
- [5] Weishi Shi et al. “Presenting and Evaluating the Impact of Experiential Learning in Computing Accessibility Education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. ICSE-SEET ’20. Distinguished Paper Award. Seoul, South Korea: Association for Computing Ma-

chinery, 2020, pp. 49–60. ISBN: 9781450371247. DOI: 10.1145/3377814.3381710. URL: <https://doi.org/10.1145/3377814.3381710>.

Hands On Data Structures: Making Abstraction Tangible with Manipulatives

Conference Tutorial

Halie M. Rando¹, Jade Lilian Palosky¹, Hayley Ma¹,
Vivian Myers¹, Sofia Jurusz¹

¹Department of Computer Science

Smith College

Northampton, MA 01063 USA

`hrando@smith.edu`

Data structures are fundamental to computer science education, yet many students struggle to develop accurate mental models of abstract data, memory organization, and dynamic representations of state that are not directly observable. In CS2 in particular, students must reason simultaneously about algorithmic behavior, memory layout, and evolving structure, often without real-world experiences to ground these concepts. Abstraction skill development has therefore long been a central focus of computer science pedagogy [3, 1].

Models are widely understood to support students’ understanding of abstract concepts [2]. In computer science, models designed to bridge the tangible and the abstract are often computer-based. In many other STEM fields, however, students routinely use physical manipulatives to reason about invisible structures and processes. In chemistry, for example, comparisons across physical, symbolic, and computational representations suggest that physically manipulable models offer distinct advantages for developing accurate mental models relative to purely digital representations [5]. Although active learning in computer science is well established, the use of physical manipulatives for teaching core topics such as data structures remains less common.

Building on these insights, this tutorial introduces a set of hands-on activities that make abstract data structures tangible through physical manipulation. For example, participants will use a low-cost linked-list manipulative built from plastic Easter eggs: each egg represents a node, with color/external markings indicating the node’s “data” (so students can refer to nodes by human-readable

identity, e.g., “the green #2 egg”), and a sticky note *inside* the egg indicating the “next” reference (e.g., to another egg by ID or location). By requiring students to open each egg to retrieve the “next” pointer, the list structure cannot be inferred by lining eggs up visually, making traversal and hidden state explicit. By physically opening eggs to inspect values and rewiring “next” labels to enact insertion, deletion, and traversal, students make the hidden structure of a list concrete while practicing invariants such as head/tail updates and null termination. Similar activities use everyday materials (e.g., cardstock, mailing tubes and balls, beads and string) to model arrays, stacks/queues, trees, graphs, and more, and to support reasoning about traversal, search, recursion, and algorithmic processes. These activities are designed not to replace formal representations or code-based implementations, but to scaffold students’ development of mental models that transfer to code and support deeper engagement with abstraction. The presentation will be led by an instructor who has iteratively developed and deployed these activities over six semesters, expanding from a small set of movement-based exercises to a full suite of manipulatives, alongside students and undergraduate teaching assistants who have helped develop and refine the activities.

Hands-on activities benefit STEM learning at both the high school and college levels, particularly for students without prior exposure to course content [4]. Over the past three academic years, we have observed improvements in classroom climate after integrating these low-stakes, collaborative activities. The investment required to implement the activities is minimal, with materials that can be acquired at negligible cost per student and in many cases reused across semesters. The tutorial itself is interactive: attendees will have the opportunity to complete short manipulative-based activities and leave with classroom-ready materials and facilitation instructions adaptable to CS1, CS2, early algorithms courses, and beyond. Instructions for running activities, including those presented in the tutorial, will also be publicly available at `HandsOnDataStructures.github.io`.

References

- [1] Marjahan Begum, Julia Crossley, Filip Strömbäck, Eleni Akrida, Isaac Alpizar-Chacon, Abigail Evans, Joshua B. Gross, Pontus Haglund, Violetta Lonati, Chandrika Satyavolu, and Sverrir Thorgeirsson. A pedagogical framework for developing abstraction skills. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE 2024, page 258–299, New York, NY, USA, 2025. Association for Computing Machinery.
- [2] John K Gilbert and Carolyn Boulter. Models and modelling in science education. *Hatfield, UK: The Association for Science Education*, 1993.

- [3] Claudio Mirolo, Cruz Izu, Violetta Lonati, and Emanuele Scapin. Abstraction in computer science education: An overview. *Informatics in Education*, 20(4):615–639, 2022.
- [4] Olufunso Oje, Olusola Adesope, and Adurangba Victor Oje. Work in progress: The effects of hands-on learning on stem students’ motivation and self-efficacy: A meta-analysis. In *2021 ASEE Virtual Annual Conference Content Access*, Virtual Conference, 7 2021. ASEE Conferences.
- [5] Vesna Ferk Savec, Margareta Vrtacnik, and John K Gilbert. Evaluating the educational value of molecular structure representations. In *Visualization in Science Education*, pages 269–297. Springer, 2005.

Biography

Halie Rando has been an Assistant Professor of Computer Science at Smith College since 2023. As an information scientist, she began designing hands-on activities during her first semester of teaching CS2 to help students internalize strategies for information storage and retrieval.

Jade Lilian Palosky is a senior undergraduate computer science major at Smith College. She has supported over 100 students in learning to reason about core data structures concepts in her role as the teaching assistant for CSC 210: Data Structures.

Hayley Ma and Vivian Myers are sophomore undergraduate students at Smith College double majoring in computer science and biology. They will present their hands-on final project on priority queues from Spring 2024, which has since been integrated into the curriculum.

Sofia Jurusz is a junior math major at Smith College who is also pursuing coursework in computer science and education, specifically STEM education. She is interested in the theory of why hands-on activities help students retain and apply concepts in math and computer science.

Next-Gen Teaching: Harnessing AI in Course Design*

Conference Workshop

Patrick Ester, and Mohit Chandarana
Codio, Inc.
Boston, MA

{patricke, mchandarana}@codio.com

This hands-on workshop shows how to integrate generative AI into course design, assessment, and feedback—without surrendering pedagogy to tooling. After a quick live poll to select a familiar “Intro to Computing” –style topic, we’ll co-build a starter module: outline learning outcomes, generate draft content, create fresh question banks, and configure autograded and rubric-based assessments. We’ll demonstrate platform-agnostic patterns (prompt libraries, context packaging, and process-based grading) and then illustrate the same patterns in an integrated demo environment for speed and scale. Participants will also configure AI assistants that give targeted hints and feedback while preserving academic integrity. All attendees receive a reproducible checklist and a starter pack that they can integrate with their own LMS or adapt to their own platform. Free Instructor accounts are pre-provisioned to minimize setup time. (No platform purchase is required.)

By the end of this workshop, participants will be able to:

- Configure a course-aware AI authoring workflow to generate an outcomes-aligned mini-module (outline, content, and activity) from existing teaching materials.
- Diagnose the limits of disconnected web platform use (context loss, reproducibility, privacy) and compare efficiency and output quality against an integrated approach.

*Copyright is held by the author/owner.

- Author two AI-ready/AI-resilient assessments in modalities relevant to their domain (e.g. problem-solving, code/data, short-answer/essay), with rubrics and refreshable variants.
- Deploy a course-scoped learning assistant that provides formative, context-aware support under explicit guardrails (no full solutions; transparency, logging).
- Draft a practical AI-use statement and implementation checklist that address academic integrity, equity, and accessibility for their own course.

Planned Schedule of Workshop

3h (includes break)

- Welcome, goals & quick poll (15 min)
- Workflows: disconnected vs. integrated (20 min)
- Q&A and Ethics Checkpoint (5-10 min)
- Course skeleton (15 min)
- Content drafting (20 min)
- Q&A and Ethics Checkpoint (5-10 min)
- Break (15-20 min)
- Assessments (deep dive) (30 min)
- Learning assistants: configuration & safeguards (30 min)
- Policy, share-outs & end of workshop Q&A (10-15 min)

Presenters

- Patrick Ester, *Director of Content*

Biography: Patrick is a veteran teacher, with ten years of experience in the classroom. He worked to integrate computer-based activities into the curriculum and taught students how to code. He has also led workshops on physical computing. At Codio, he leads the content and curricula team. Patrick enjoys creative coding, reading, and baseball.

- Mohit Chandarana, *Principal Research Scientist*

Biography: Mohit has a BE in Computer Engineering and an MS in Computer Science. From generating insightful learning analytics for CS Educators to prototyping novel product features and algorithms, he works towards bridging the gap between cutting-edge academic research and its application in the industry in his role at Codio.

PLCC: A Tool Set for Teaching Programming Languages Courses*

Conference Workshop

Timothy Fossum, plcc@python.net

Professor Emeritus, SUNY Potsdam

James Heliotis, jehics@rit.edu

Professor Emeritus, Rochester Institute of Technology

Stoney Jackson, stoney.jackson@wne.edu

Professor, Western New England University

Alain Kägi, alaink@clark.edu

Assistant Professor, Lewis

Clark College

This is a hands-on laptop-recommended workshop that introduces participants to the PLCC compiler-compiler tool set for building Java-based language interpreters in an upper-level Programming Languages course. One approach to teaching this course is to show how programming language features such as language syntax, variable lifetime, procedure application, parameter passing, recursion, and object-orientation are implemented "under the hood". This workshop is designed for CS educators who want to explore how PLCC can be used to support such an approach.

1 How PLCC Works

The PLCC system contains a compiler-compiler (implemented in Python 3). It also contains a small collection of Java support programs. Input to the PLCC compiler-compiler is a single specification file with three sections that describe the lexical, syntactic, and semantic structure of a programming language. The PLCC compiler-compiler takes this specification and generates Java source files that implement a self-contained interpreter for the language, including a front-end scanner, a recursive descent parser, and an evaluation engine including

*Copyright is held by the author/owner.

a read-eval-print loop. The specification file defines language tokens using simple regular expressions, language syntax using straightforward BNF rules, and language semantics as Java code segments. These code segments become part of Java classes, built by PLCC, that correspond to each of the BNF rules in the syntax specification.

2 The Workshop

Participants will be asked to work in small teams so they can learn from each other and benefit from problem-solving discussions. Each team will share a laptop computer owned by one of the team members. The laptop will be used only to log into a workshop-provided server that contains all of the resources necessary to run PLCC on the workshop examples.

Teams will work through a sequence of examples and problems that illustrate, in turn, each of the essential parts of a language specification:

- Lexical specification examples for token recognition
- Lexical and grammar specifications for illustrating syntax recognition
- Interpretation of programs written in simple programming languages

Each example will be presented by (1) describing a worked-out PLCC example specification that teams can test on the server; (2) proposing a similar problem whose solution requires modification or extension of the given example; and (3) having the participant teams implement the proposed modification on the server by modifying and testing the PLCC specification file. These examples will be used as a springboard to illustrate how to implement language features, including scope, functional programming, call semantics, object-oriented programming, type systems, and logic-based programming.

3 Workshop Preparation by Participants

A laptop is **not necessary** to participate in this workshop. However, the workshop requires a sufficient number of participants with laptops to allow for putting together teams of two or three. Laptops need a Web browser installed (ideally Google Chrome or Microsoft Edge) and Wi-Fi access.

To interact with the workshop materials, participants will use GitHub and Codespaces. Participants need a free-tier GitHub account, and are asked to create one before attending the workshop.

All of the PLCC workshop materials are available on the PLCC website: <https://plcc.python.net>, and on GitHub <https://github.com/ourPLCC>.

Amp Stack: A System for Teaching CS1 Modularity*

Conference Workshop

Angel Rivera
Department of Computer Science
Utica University
Utica, NY 13413
arivera@utica.edu

Abstract

Introductory programming students routinely struggle with the “invisible” aspects of modular programs: where values come from, how they move between functions, and what it means to “return” a result. Amp Stack is a lightweight, visual methodology that makes data flow and modular structure explicit by having students design programs as top-to-bottom stacks of “amps” (Input, Compute, and Output blocks) wired together by named ports before they write any code. In this 3-hour, hands-on workshop, participants will experience Amp Stack from the student point of view, translate diagrams into runnable Python, and then step back to examine how the approach scaffolds CS1 concepts such as IPO structure, parameter passing, and modular decomposition. Attendees will leave with ready-to-use classroom materials (worksheets, sample exercises, and rubrics) and a plan for integrating Amp Stack into their own introductory courses with minimal disruption to existing syllabi.

1 Context and Motivation

Even in well-designed CS1 courses, novices often miss how a program transforms input into output. Common symptoms include confusing arguments with parameters and believing the names must match; forgetting to capture returned values or to use them in subsequent computations; and mixing I/O and computation in ways that hinder testing, reuse, and reasoning.

*Copyright is held by the author/owner.

Amp Stack addresses these issues by giving students a concrete, visual language for program structure before and alongside textual code. Programs are drawn as vertical stacks of block types, each with typed ports connected by named “wires.” Data is required to flow strictly top-to-bottom, mirroring the structure of simple Python scripts and providing a disciplined IPO (Input–Process–Output) pattern. Each block maps directly to a single Python statement or function, keeping syntax overhead low while reinforcing a strong data-flow mental model. The concept is then extended to the creation of “custom blocks,” implemented as functions, to introduce modular decomposition, abstraction, and reusability. Amp Stack offers a practical way to (a) introduce modular thinking before loops and conditionals, (b) foreground separation of concerns, and (c) reduce recurring misconceptions about parameter passing and return values.

2 Workshop Structure and Activities (3 hours)

The session is designed as an active-learning experience with brief mini-lectures interleaved with guided practice, pair work, and whole-group discussion. Timing is approximate but tuned to CCSC’s 3-hour pre-conference format.

Phase I – Conceptual Introduction (0:00–0:30)

- Mini-lecture & demo (15 min).
- Individual quick-draw activity (10–15 min).

Phase II – Guided Practice & Mapping (0:30–1:45)

- Hands-on Practice 1: IPO to Code (30–35 min).
- Hands-on Practice 2: Modular Programs (35–40 min).
- Whole-group debrief (10 min).

Break (1:45–1:55)

Phase III – Collaborative Problem-Solving & Diagnosis (1:55–2:55)

- Faulty Code Clinic (30 min).
- Refactor & Card Sort Activity (20–25 min).
- Group discussion (10 min).

Phase IV – Q&A, Adaptation, and Assessment (2:55–3:10)

- Implementation planning (10–12 min).
- Q&A and wrap-up (3–5 min).

A Summer Curriculum in Data Visualization and Analysis for High School Students *

Poster Abstract

Paolien Wang, Ching-yu (Austin) Huang
Department of Computer Science & Technology
Kean University
Union, New Jersey, USA
{pawang, chuang}@kean.edu

As data-driven decision making becomes increasingly vital across industries, skills in online data visualization and analysis are in high demand. While many colleges now offer coursework in data analytics and visualization, providing earlier access to these concepts can better prepare motivated high school students for advanced study and early internship opportunities in a competitive job market.

This study presents the design and implementation of a four-week summer curriculum in data visualization and analysis tailored for high school students. The program meets four days per week for six hours per day, combining three hours of lecture with three hours of hands-on lab work. The curriculum focuses on three-tier architecture for data visualization design, development, and analysis. No prior programming experience is required; however, students are expected to have completed Algebra and to demonstrate strong interest in computing.

The learning outcomes of the curriculum are for students to:

- understand the purpose and impact of data visualization and analysis;
- develop online data visualization and analysis applications;
- create interactive visualizations that support user engagement; and
- apply storytelling techniques to communicate insights and patterns.

*Copyright is held by the author/owner.

Students learn the fundamentals of Structured Query Language (SQL) to query and aggregate data within a MySQL database; PHP and MySQL functions for application-layer development; and JavaScript, event handling, and the Google Charts Library for presentation-layer design. Instruction covers core chart types—including line, pie, bar, histogram, and scatter plots—and guides students through retrieving data from a database and visualizing it effectively. Analytical components introduce students to interpreting distributions, identifying correlations and outliers, and validating findings with AI-assisted tools such as ChatGPT.

Overall, this accelerated curriculum provides high-achieving high school students with foundational skills in computer science, data science, and AI-related fields, positioning them for early success in college-level coursework, internships, and future career pathways.

Keywords: 3-Tier Architecture, Database, Data Mart, Data Visualization, High School, SQL

Creating a Web Application to Scaffold Student Practice Work for a Discrete Mathematics Course *

Poster Abstract

Christopher M. Morales
Department of Computer Science
SUNY Plattsburgh
101 Broad Street
Plattsburgh, NY 12901
mora1722@plattsburgh.edu

While teaching discrete mathematics using open educational resources, I observed that many students needed additional practice with immediate, targeted feedback. Standard non-interactive practice problem sets suffer from the limitation that they must either be graded by the instructor or distributed with solutions so that the student can check their own work. Instructor-graded problems necessarily limit the time that a student can usefully spend practicing. Problem sets distributed with solutions present a temptation to the student to read the answers rather than attempt the problems independently and limit the amount of feedback a student can receive per problem, since they are typically not able to detect a mistake at any step without viewing the complete solution.

I developed a web application that provides scaffolded practice in constructing truth tables for arbitrary logical statements, and optionally for using the resulting truth table to answer questions on equivalence and argument-form validity. I also created a series of procedural question generators, allowing students to encounter a wide range of distinct problems.

Existing interactive web applications for practicing truth table construction [4, 1, 2] and equivalency problems [3] present students with a fully-structured truth table, including all column headers, and ask students only to compute the entries. In contrast, my application begins earlier in the reasoning process.

*Copyright is held by the author/owner.

When using my application, a student is first asked to indicate all logical operators in a given statement. Once this task is completed, the application highlights each operator in turn, asking the student to indicate all symbols (whether logical operators or statement variables) belonging to sub-statements of the indicated operator. After successfully processing all operators, the student is asked to choose an order in which to evaluate the sub-statements when constructing the truth table. If a valid order is specified, the student is provided with a blank truth table to fill in, with the columns in their chosen order. For logical equivalence and argument-form validity questions, the student is further asked to identify the columns and, for argument questions, rows needed to answer the final question. This structure allows feedback to be immediately provided at each intermediate stage of reasoning, rather than only on final answers.

For each student, the application asks a series of questions of escalating complexity before switching to a free-practice mode once the student has successfully completed their question series. Because questions are generated procedurally, each student will continue to receive fresh questions during free-practice, though since questions are of finite size duplicates may be encountered as well during free-practice.

In this poster, I present my experiences designing the software and using it during the Fall 2025 semester and the first part of the Spring 2026 semester, including observations about how the application supported more flexible and sustained student engagement with discrete mathematics practice.

The source code for my application is available at <https://github.com/drchrismorales/TruthTableApp>. It is licensed under the Apache 2.0 License.

References

- [1] IXL - Truth tables (Geometry practice).
- [2] Truth tables numbas quiz.
- [3] Douglas E Ensley and J. Winston Crawley. Equivalent Propositions, 2023.
- [4] Douglas E Ensley and J. Winston Crawley. Truth Table Practice, 2023.

Common Student Mistakes in Building Data Marts for Effective Data Visualization*

Poster Abstract

Ching-yu (Austin) Huang
Department of Computer Science & Technology
Kean University
Union, New Jersey, USA
`chuang@kean.edu`

Data visualization is a powerful tool for identifying patterns and interpreting complex datasets. However, effective visualizations critically depend on the accuracy of the underlying data marts—typically created through SQL GROUP BY clauses and aggregate functions. Because each visualization type (e.g., line charts, bar charts, pie charts, scatter plots, histograms) requires a correctly structured data mart, even small mistakes in data preparation can lead to misleading graphs and incorrect conclusions.

This study examines several recurring errors made by students when constructing data marts for visualization tasks and illustrates how these mistakes can distort subsequent analyses. Using the Simplemaps US ZIP code demographic dataset—where each record represents a ZIP code—we present practical examples and demonstrate correct and incorrect approaches. For instance, when creating a scatter plot to compare average income and unemployment rate by state, students often incorrectly apply SQL’s AVG() function directly to ZIP code-level data, failing to account for population differences that introduce unintended weighting. Another common error occurs when comparing county-level populations by grouping solely on county_name without including the state, inadvertently merging counties with the same name across different states.

Because users often focus on charts rather than on how the underlying data marts were generated, these errors can remain hidden yet have significant consequences for interpretation and decision-making. This work highlights

*Copyright is held by the author/owner.

the importance of teaching students how to construct accurate data marts and provides guidance for avoiding common aggregation pitfalls in SQL-based analytics.

Keywords: aggregation, data mart, data visualization, SQL, GROUP BY, graphs

Growing STEM Student Success with Interdisciplinary Data Science Courses *

Poster Abstract

Beryl Hoffman, Nina Theis, Stephen Walsh
Elms College
Chicopee, MA 01013

`{hoffmanb, theisn, walshst}@elms.edu`

As part of a NSF S-STEM grant, Elms College has piloted requiring students who are STEM scholars to take two cohort-based interdisciplinary Data Science courses to strengthen student skills and student engagement across STEM disciplines. First-year STEM scholars in the majors of Biology, Biotechnology, Chemistry, Computer Science, Comp. Info. Tech. and Security, and Data Science & AI, enroll as a cohort in a newly developed course, Computational Statistics, that fulfills a mathematics core requirement. The course introduces statistical reasoning, data analysis, and programming in R using authentic datasets drawn from multiple STEM fields. Transfer STEM scholars and juniors complete a second course, Data Analytics and Visualization, taught in Python. This course culminates in a discipline-specific final project in which students select and analyze real-world datasets relevant to their majors. These courses have helped our STEM students build peer networks early in their program and engage in cross-disciplinary collaborative problem solving.

Embedding data science across disciplines has helped students develop transferable skills in quantitative reasoning, computational skills, and data analysis that prepare students for advanced research in STEM fields. Shared learning experiences promote peer support and sustained engagement, particularly for students entering with diverse academic backgrounds. Since 2022, 47 STEM scholars (26 currently active) have participated in these cohort-based data science courses, providing multiple cohorts to observe patterns of engagement and skill development. In addition to positive student feedback and increased participation in research, internships, or advanced courses, the initiative has contributed to growth in the institution's Data Analytics minor,

*Copyright is held by the author/owner.

Computer Science, and Data Science and AI major. This work highlights how interdisciplinary data science instruction can serve as a powerful mechanism to improve student learning and engagement in STEM programs.

Bridging the Gap with AI Literacy*

Lightning Talk

Carolyn C. Matheus¹, Kevin M. Gaugler²

¹Department of Computing Technology

²Department of World Languages and Cultures

Marist University

Poughkeepsie, NY 12601

`carolyn.matheus@marist.edu`

Artificial intelligence (AI) is rapidly transforming higher education, prompting both excitement and concern among academic leaders. Designed to pair technical computer-science focused concepts with critical thinking and ethical reasoning, this course prepares students not only for an AI driven workforce but for thoughtful participation in an AI mediated society. Through hands-on projects, lectures, and weekly labs, it provides a foundational understanding of the core technical concepts of AI along with societal and ethical implications of AI and generative AI (GenAI). This course equips students with skills to navigate an AI-driven world. By the end of the course, students will:

- Demonstrate knowledge of the fundamental principles of AI
- Identify and differentiate AI technologies for applications and contexts
- Explain foundational methods for collecting and training data
- Utilize AI tools for problem-solving and creative tasks
- Critically analyze AI systems and articulate societal implications
- Demonstrate an understanding of the AI development process

Weekly modules are organized around three central themes: foundational AI knowledge, ethical and societal implications, and interdisciplinary applications. Students are introduced to key concepts such as GenAI founda-

*Copyright is held by the author/owner.

tional models, model functionality (e.g., tokenization, algorithms, reinforcement learning), prompt engineering, machine learning, bias (i.e., infrastructure, training data, and prompt bias), environmental sustainability, information literacy, data privacy, and governance. Instruction combines short lectures with active learning activities that emphasize comparison, experimentation, and critical reflection. Experiential activities include evaluating AI models and tools for research and productivity, designing prompts, building AI collaborators and conversation agents, vibe coding, exploring creative and computational applications, and examining ethical trade-offs related to bias, environmental sustainability, and data governance. Rather than positioning AI as a replacement for human judgment, the course reinforces the importance of critical evaluation, transparency, and human responsibility. Course assignments include:

- **Project 1:** Students work in groups to research foundational AI topics (e.g., machine learning, deep learning, neural networks, computer vision, robotics, LLMs, and testing and training data). Groups design and deliver an instructional presentation to the class that includes an interactive learning activity. Students are encouraged to use AI tools to support lesson planning and reflection on prompt design techniques.
- **Project 2:** Students work in groups to investigate how GenAI is impacting specific disciplines or societal domains, examining applications, benefits, limitations, and ethical implications. Deliverables include a formal research paper and a class presentation. AI use disclosure forms are required.
- **Project 3:** Students individually design and document a semester-long learning project about how personal learning can be enhanced by using AI as a collaborator. The portfolio emphasizes goal-setting, reflection, iteration, and evaluation of GenAI as a learning partner. A mid-semester progress report supports metacognitive reflection.
- **This Just In...AI in the News:** Students spend 2 minutes sharing emerging developments in AI and GenAI in a lightning-talk presentation to the class.

AI is transforming professional landscapes. This course demonstrates how institutions of higher education can proactively address these challenges through a flexible, interdisciplinary framework that integrates technical understanding with ethical awareness and societal reflection. This course encourages students to move beyond the passive use of AI tools toward critical engagement with their development, capabilities, and consequences.

Ransom Sentinel (RaSe): Designing Ransomware-Resilient Systems Through Adversarial Simulation and Cryptography*

Lightning Talk

Chetan Jaiswal
School of Computing & Engineering
Quinnipiac University
Hamden, Connecticut
cjaiswal@qu.edu

Abstract

Ransomware remains one of the most disruptive cyber threats to critical infrastructure, with healthcare systems particularly vulnerable due to their stringent availability and privacy requirements. This lightning talk presents Ransom Sentinel (RaSe), a multi-layered ransomware resilience framework that integrates offensive attack simulation with cryptographic and systems-level defense mechanisms. Unlike many proprietary or black-box solutions, RaSe is grounded in reproducible adversarial modeling: reconnaissance, unauthorized access, and payload execution were emulated using tools such as Nmap and custom client-server exploits (like Heartbleed) to expose real attack surfaces and failure modes. Building on these insights, RaSe combines Shamir’s Secret Sharing for distributed key management, Reed–Solomon erasure coding for fault-tolerant storage, entropy-based anomaly detection for early encryption discovery, Zero Trust access control, and immutable audit logging to support forensic integrity and regulatory compliance. The framework is evaluated in the context of healthcare workflows, emphasizing the Durability–Accessibility–Loss (DAL) principles and demonstrating how secure recovery and “break-glass” access can coexist with strong cryptographic safeguards. This work highlights how integrating offensive experimentation with defensive architecture can transform ransomware response from reactive recovery to proactive resilience. The lightning talk will summarize the system design, threat model, and educational value of

*Copyright is held by the author/owner.

using adversarial simulation to teach and research next-generation cyber defense strategies.

1 Background

This lightning talk introduces RaSe [1], a comprehensive solution designed to provide multi-dimensional security and is resilient to ransomware attacks. This work addresses the countermeasures to defend against attacks. It also offers a breach model in which the critical data remains durable and accessible even during a live cyberattack. It offers complete data recovery in case of total system shutdown. This work was supported and funded by The Glenn W. Bailey Foundation. It finds its foundation in developing a research framework for undergraduate Computer Science students. The research team consisted of Chetan Jaiswal (mentor) with Jessica Theodore & Dawit Kasy as two students. Using agile development with a sprint cycle of 2 weeks, this work was completed in one academic year. The workflow followed the following steps:

- a. Idea Conception & Problem Identification
- b. Problem statement and developing vision for potential solutions
- c. Study of state-of-the-art ransomware attacks and countermeasures
- d. Formulation of multi-layered approach towards solution
- e. Evaluation against risk assessments and threat models
- f. Academic Recognition and Dissemination [1]

2 Future Works

Future work will focus on developing a full-scale prototype of RaSe and conducting performance evaluations to measure recovery latency, cryptographic overhead, and system scalability under realistic attack conditions. The framework can be extended with adaptive AI-driven anomaly detection to identify zero-day ransomware, along with stronger insider-threat modeling using hardware-backed key protection and behavioral monitoring. Additional research will explore forensic recovery and policy driven response, deployment in cloud, edge, and medical IoT environments, and formal verification of durability and availability guarantees. Finally, RaSe will be packaged as an instructional platform, enabling students to study ransomware by simulating real adversarial behavior and evaluating resilient system design in hands-on settings.

References

- [1] Jessica Theodore, Dawit Kasy, and Chetan Jaiswal. “Ransom Sentinel (RaSe): A Study Towards Defense Against Ransomware”. In: *2026 IEEE 16th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2026, pp. 0125–0134.

Rediscovering Collaborative Whiteboarding for Problem Solving in the Era of AI *

Lightning Talk

Karen H. Jin¹

¹Department of Applied Engineering and Sciences
University of New Hampshire, Manchester, NH 03861

`karen.jin@unh.edu`

As AI tools continue to improve automated code generation, CS educators must ensure that students develop problem-solving skills of their own. This work presents collaborative whiteboard-based activities used in an Introduction to Data Structures course. Through these hands-on activities, students work together to reveal and address their conceptual gaps while building their problem-solving abilities.

Zviel-Girshin [4] showed students appear to progress while lacking genuine comprehension while using AI tools. Güner and Er [2] identified student interaction profiles with ChatGPT, showing that many relied on AI for direct solutions without engaging in the problem-solving process. Meanwhile, research demonstrates the effectiveness of using traditional whiteboards. Chapin and Bowen [1] showed that whiteboard activities improve mental models and computational thinking by making abstract concepts visible and manipulable. Wong et al. [3] found that students prefer traditional whiteboards over digital tools for in-person learning.

In this work, we describe whiteboard activities used in our Data Structures classroom, using the example of teaching linked lists. Our four-step approach as described below ensures that students actively engage with the problem-solving process, while the visible nature of whiteboard work allows peers and the instructor to collectively identify misconceptions and facilitate improvements. We show how we use this whiteboarding approach in teaching linked lists and how students learn from each of these steps with concrete examples.

- Step 1. Each group works independently to discuss and develop their

*Copyright is held by the author/owner.

own solutions on paper.

- Step 2. All groups write their solution on the whiteboard, making their thinking process and solution visible to the entire class.
- Step 3. Students engage in peer critique by analyzing and discussing each other's approaches, moderated by the instructor.
- Step 4. Each group revises and improves their work on the whiteboard.

Student feedback to this approach was very positive. They particularly appreciated the peer to peer interactions and the hands-on collaborative nature of the activities. They also reported a greater confidence in their ability to solve problems.

We thus argue that traditional teaching methods such as our whiteboarding activities are more valuable than ever in the AI era. By having students externalize their thinking on a whiteboard, they can identify gaps in their understanding and improve through working through solutions by hand, all in a group setting. We plan to conduct a more rigorous evaluation of how these whiteboarding activities affect students' learning compared to AI coding assistants in our Data Structures course.

References

- [1] Chapin, J., & Bowen, B. (2023). Whiteboarding: A tool to improve CS1 student self-efficacy. In *Proceedings of the 2023 ACM Conference on Global Computing Education Vol 2 (CompEd 2023)* (pp. 161-167). ACM. <https://doi.org/10.1145/3576882.3617925>
- [2] Güner, H., & Er, E. (2025). AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies*, 30, 12681–12707. <https://doi.org/10.1007/s10639-025-13337-7>
- [3] Wong, S. S., Wong, S. F., Mahmud, M. M., & Yong, W. K. (2023). Learning in the margins: Student choice for digital and traditional whiteboards in in-person learning at university. In *Proceedings of the 2023 6th International Conference on Educational Technology Management (ICETM 2023)* (pp. 162-168). ACM. <https://doi.org/10.1145/3637907.3637969>
- [4] Zviel-Girshin, R. (2024). The good and bad of AI tools in novice programming education. *Education Sciences*, 14(10), 1089. <https://doi.org/10.3390/educsci14101089>

Essential Computing Concepts (Draft): An Alternative to CS2023 for Colleges*

Lightning Talk

Alyce Brady¹, Megan Olsen², Rick Blumenthal³, Grant Braught⁴, Janet Davis⁵, Bruce A. Maxwell⁶, Karl R. B. Schmitt⁷, Andrea Tartaro⁸, Henry Walker⁹,
Amanda M. Holland-Minkley¹⁰

¹Department of Computer Science
Kalamazoo College, Kalamazoo, MI 49006
abrady@kzoo.edu

²Department of Computer Science
Loyola University Maryland, Baltimore, MD 21210
mmolsen@loyola.edu

³Computer and Cyber Sciences Department
Regis University, Denver, CO 80221
rblument@regis.edu

⁴Department of Mathematics and Computer Science
Dickinson College, Carlisle, PA 17013
braught@dickinson.edu

⁵Department of Computer Science
Whitman College, Walla Walla, WA 99362
davisj@whitman.edu

⁶Khoury College of Computer Science
Northeastern University (Seattle), Seattle, WA 98109
b.maxwell@northeastern.edu

⁷Computing and Data Analytics Department
Trinity Christian College, Palos Heights, IL 60463
krbschmitt@gmail.com

⁸Department of Computer Science
Furman University, Greenville, SC 29690
andrea.tartaro@furman.edu

⁹Department of Computer Science
Grinnell College, Grinnell, IA 50112
walker@cs.grinnell.edu

¹⁰Department of Computing & Information Studies
Washington & Jefferson College, Washington, PA 15301
ahollandminkley@washjeff.edu

This lightning talk will briefly introduce and solicit feedback on “Essential Computing Concepts”, a draft set of proposed guidelines developed as an alternative to the ACM/IEEE CS2023 guidelines for curriculum design and assessment. The publication of CS2023 resurfaced a long-known tension between curriculum guidance through large and detailed lists of topics, and the curricular priorities and constraints of some programs. For example, smaller institutions, liberal arts colleges, and computing programs looking to innovate around CS+X, would benefit from a smaller list of what is truly essential to computing majors, which can then be expanded based on that program’s focus to fully develop the major.

The SIGCSE Committee on Computing Education in Liberal Arts Colleges hosted two sessions at SIGCSE TS 2025 to collect specific feedback on how programs would benefit from a smaller definition of core concepts and how they think that core should be defined. Roughly 50 attendees participated in those sessions. From those participants, interested faculty from a variety of institutions formed a working group committed to developing this smaller core that would support flexibility, exploration, and interdisciplinary work within a CS curriculum. The group reviewed feedback from the SIGCSE TS 2025 sessions to determine the necessary features of an alternative set of curriculum guidelines. These features support educators’ goals for a smaller core for a CS major that could also serve as the basis for a CS minor or CS+X program, could feasibly be taught within a small program with limited faculty, and would be small enough in scope to ensure space for academic exploration, or other student or institutional priorities.

The group then designed a draft set of guidelines, structured around 12 learning goals that are organized into four categories: Read, Understand, and Analyze Models; Design and Create Using Models; Use Math, Data and Analytics for Computer Science; and Be a Responsible Computing Practitioner. Each high-level learning outcome is supported by component outcomes, essential computing concepts connected to the outcome, and illustrative examples of what might be taught to achieve the outcome. The guidelines are presented in a way that avoids presuming a particular mapping of concepts to courses and encourages programs to think about how high-level learning outcomes are achieved across an entire major.

This lightning talk introduces this draft of Essential Computing Concepts and associated learning goals (available at <https://tinyurl.com/ECCSIGCSE>), and invites attendees to provide feedback on this work in progress. The authors appreciate the feedback already received from our event attendees and the contributions of James Teresco at Siena University to this work.

*Copyright is held by the author/owner.

Designing Interdisciplinary Neuro-divergent Inclusive CUREs in Computer Science*

Lightning Talk

Karina Assiter¹, Sue Vincent²
STEM Department
Landmark College, Putney VT 05346

¹KarinaAssiter@landmark.edu

²SueVincent@landmark.edu

Course-Based Undergraduate Research Experiences (CUREs) engage students in authentic, open-ended research in which learners collaboratively investigate meaningful questions, iterate on methods, and contribute research products beyond the classroom. While well established in the life sciences, CUREs remain less common—and less clearly articulated—in computer science.

This lightning talk will introduce a proposed CURE-informed training framework being developed through two coordinated *Vermont Biomedical Research Network* (VBRN) Pilots at Landmark College, a college specializing in the education of neuro-divergent learners. The Pilots are designed to work in tandem, providing an opportunity to make research workflows, expectations, and collaboration practices explicit—features expected to support neuro-divergent students while benefiting a broad range of learners.

The project will consist of a Computer Science Pilot and a complementary Biology Pilot. The Pilots will interface through shared research questions, coordinated training activities, and aligned workflows, together forming an interdisciplinary CURE model that supports collaboration rather than treating disciplinary preparation in isolation.

The Pilots will center on research-team-based CUREs situated outside formal coursework, preserving core CURE elements—authentic research, discovery, collaboration, and iteration—while reducing barriers associated with grading pressure and rigid pacing. Within this structure, students in the Computer

*Copyright is held by the author/owner.

Science Pilot will collaborate with Biology students to design reproducible Python-based workflows for analyzing open biomedical imaging datasets.

Although developed within a neuro-divergent-serving institutional context, the training framework emerging from the Pilots is intentionally designed to be transferable and neuro-diversity inclusive. By sharing this work at the design stage, we aim to incorporate practitioner insight into the Pilot design itself, strengthening both the framework and its relevance for computer science educators across institutional contexts.

Toward True Differentiated Instruction with PathMX and Curriculum as Code*

Mark Johnson¹ and Chris Alvin¹
¹Department of Computer Science
Furman University
Greenville, SC, 29613
{mark@fellowhumans.com, chris.alvin@furman.edu}

Abstract

Students enter CS1 courses with widely varying programming experience, mathematical maturity, and learning preferences. We propose a framework called PathMX that addresses this challenge through a “curriculum as code” methodology that treats curriculum materials as structured, version-controlled repositories editable by both instructors and LLM-based coding agents. When curriculum is authored as code following clear conventions, agents can generate personalized content on demand, enabling scalable differentiated instruction that was previously impractical. Our CS1 case study demonstrates that this approach makes true differentiated instruction achievable within the constraints of real classroom teaching.

1 Introduction

CS1 instructors routinely face cohorts where students arrive with vastly different readiness levels. The diversity arises from differences in prior programming experience, mathematical maturity, formal reasoning abilities, and learning preferences. Some students demonstrate their mastery of Python syntax because they have been coding since middle school. Others are confounded by

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

the first for loop. Some grasp algorithmic thinking immediately, while others require multiple explanations and/or varied examples before achieving understanding. This is not a flaw in our students. It is the reality of differences in cognitive abilities, prior experiences, and educational backgrounds.

An effective response to this diversity is differentiated instruction, where each student receives a truly tailored learning experience instead of a one-size-fits-all model. As Tomlinson describes in her foundational work, differentiated instruction is “a philosophy of teaching rooted in deep respect for students, acknowledgment of their differences, and the drive to help all students thrive” [17]. When done well, differentiation promotes inclusivity and equity by (1) addressing diversity in cognitive abilities, prior experiences, and learning profiles, (2) enhancing engagement and motivation by connecting to student interests and backgrounds, and (3) improving academic outcomes through appropriately scaffolded instruction [16]. These goals align with Universal Design for Learning (UDL) principles of providing multiple means of representation, engagement, and expression [1].

However, true differentiated instruction is nearly impossible to deliver at scale. For a typical CS1 class, an instructor implementing differentiated instruction might need to (among other tasks): (a) Create personalized learning guides for students struggling with functions; (b) Generate project variants that reflect diverse student interests, backgrounds, experiences, and identities; (c) Provide detailed, individualized feedback on every coding assignment; and (d) Adapt examples to match different backgrounds and interests. This represents only a fraction of what genuine differentiation demands. The time investment required for true differentiation is unsustainable for any educator.

Thus, we make compromises. We create one set of assignments and *hope* they work for most students. We provide individual feedback when possible, but the time required limits how detailed or frequent it can be. We use the same examples semester after semester. We know some students are not getting what they need, but manually creating differentiated materials for each student is unrealistic given time constraints and the limits of human creativity.

PathMX (Path Markdown eXtension) addresses this challenge. PathMX is a “curriculum as code” methodology that treats curriculum materials as structured, version-controlled repositories editable by both instructors and LLM-based coding agents. Existing platforms optimize for human authoring or interactive delivery; none provide conventions specifically designed for agent-based curriculum generation. When curriculum is authored as code following clear conventions, agents can generate personalized content on demand. This includes creating individualized student guides in minutes, generating project variants tailored to different interests, drafting detailed code feedback for instructor review, and adapting materials to specific learning needs. As depicted

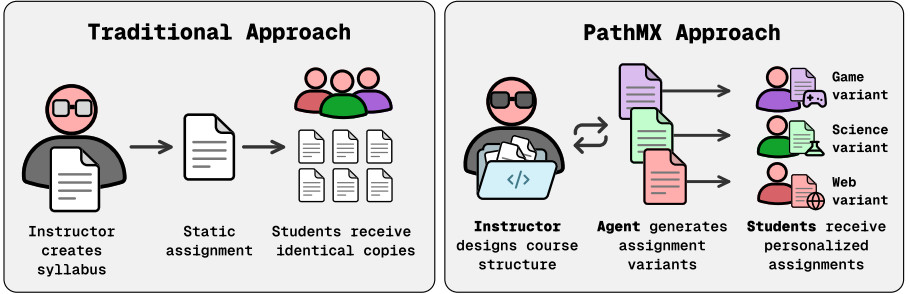


Figure 1: Traditional approaches deliver identical materials to all students; PathMX generates personalized variants through structured repositories and agent-based content generation.

in Figure 1, the instructor stays in control of pedagogical decisions while agents handle the time-consuming work of content generation and personalization.

We demonstrate this approach through a real classroom deployment: a CS1 course at Furman University in Fall 2025. Using PathMX, we created personalized learning guides for individual students, generated themed project variants, provided detailed code review feedback, and rapidly iterated on curriculum improvements, all while maintaining version control and a unified curriculum repository. Tasks that would have required many hours of manual authoring and customization were often completed in minutes with agent assistance.

Our contributions described in this paper include:

- The PathMX methodology: markdown conventions, metadata structures, and file organization that enable agent-based curriculum authoring.
- Practical workflows for creating personalized and differentiated curriculum at scale, demonstrated through our CS1 case study.
- Lessons learned from real classroom deployment, including both benefits realized and challenges encountered.

Collectively, this work establishes agent-based curriculum authoring as a viable path toward scalable differentiated instruction.

2 Related Work

Several platforms enable instructors to create interactive computing textbooks. Runestone Interactive [12] pioneered the open-source interactive textbook model, supporting embedded code execution, visualizations, and auto-graded

exercises. The platform [4] now serves thousands of daily learners across many textbooks authored in reStructuredText. OpenDSA [15] introduced a modular, community-driven approach for algorithm visualization textbooks, where instructors can customize content selections for their courses. LiaScript [2] represents the closest technical analog to PathMX, extending standard Markdown with interactive elements and using a browser-based compiler that requires no server infrastructure.

The application of software engineering practices to curriculum management has gained traction in the past few years. Rodriguez et al. [13] defined “courseware as code” principles at the U.S. Army Cyber School, where all educational content is stored in machine-readable markup in GitLab repositories. RepoBee [9] provides a command-line interface for managing student Git repositories at scale, enabling batch repository generation and automated assignment distribution. Hsing and Gennarelli [6] found that GitHub use in CS courses predicted better learning outcomes in a large-scale study of students.

Recent work has explored using large language models for educational content creation in computing courses. Sarsa et al. [14] demonstrated that LLMs could generate programming exercises with problem descriptions and solutions. However, only approximately 31% of generated exercises had sample solutions that passed their accompanying automated tests. This was primarily due to issues with test quality rather than solution correctness. MacNeil et al. [11] deployed GPT-generated code explanations in a web development textbook, finding them more helpful and accurate than student-created explanations. Leinonen et al. [10] confirmed that LLM explanations were rated as more accurate and easier to understand than those created by students.

Adaptive learning systems adjust content difficulty during student interaction. Ericson et al. [3] studied adaptive Parsons problems that dynamically adjust difficulty based on student performance, both within individual problems and between successive problems. Their study found that students solving adaptive Parsons problems completed practice exercises more efficiently than those writing equivalent code, while achieving similar learning gains. Fonseca et al. [5] explored differentiated instruction in a PHP programming course using ongoing progress monitoring to adjust content based on student performance. While these systems successfully personalize learning experiences, they require pre-authored content pools and adapt only within existing materials, leaving the challenge of creating differentiated content in the first place.

PathMX synthesizes elements from these traditions while enabling instructors to generate differentiated curriculum at scale. Existing platforms excel at delivering interactive content to students or supporting human collaboration on curriculum, but lack the structured conventions needed for agent-based editing. LLM tools can generate individual exercises or explanations, but cannot

maintain consistency across an entire curriculum repository. Adaptive systems personalize content during student interaction, but require pre-authored content pools and offer no solution to the authoring bottleneck. Similarly, UDL [1] advocates for curriculum designed from the beginning with multiple means of representation, engagement, and expression. However, implementing these principles at scale faces the same authoring bottleneck as differentiated instruction. Even in computing education contexts where UDL has been successfully applied [7], creating multiple pathways requires substantial upfront investment that grows with class size and content diversity.

PathMX addresses these limitations by treating curriculum as version-controlled repositories with explicit file organization and metadata conventions. These conventions make curriculum both human-editable and agent-processable. This enables LLM agents to generate differentiated content while instructors maintain control through templates, style guides, and review workflows. Rather than adapting content during student interaction, PathMX generates differentiated variations in advance, enabling instructor review and preserving agency over pedagogical decisions.

3 PathMX: Curriculum as Code

PathMX (Path Markdown eXtension) is a set of conventions for authoring human and agent-friendly curriculum repositories. Hyperlinks between documents create paths through the curriculum that can be followed by humans and agents alike. The three primary principles of PathMX are depicted in Figure 2 and are described as follows:

1. Curriculum is represented as Markdown files with minimal metadata in a single version-controlled repository
2. Markdown files are “type hinted” to indicate their role within the curriculum domain
3. Hyperlinks are used to link related files, create learning sequences, and provide additional context for agent and human authors

Storing curriculum in a source-controlled repository provides automatic version control, history, and branching. These are essential capabilities for any collaborative authoring process, including collaboration with agents. Agents and humans make mistakes, and version control is critical for inspecting, comparing, and recovering from those mistakes.

Markdown serves as the foundation of PathMX for several reasons. It is both human-readable and machine-parseable, is the AI-native *lingua franca* for documentation and LLM-based tooling, and it is widely supported and easily

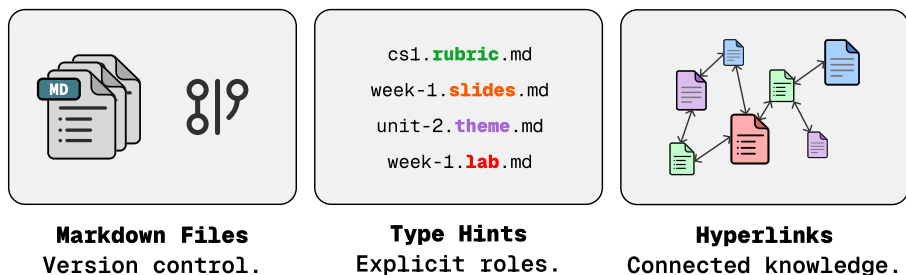


Figure 2: PathMX principles: version-controlled Markdown with type hints and hyperlinked knowledge graphs.

transformed into other formats. Modern coding models (e.g., GitHub Copilot, Gemini Code, etc.) are trained extensively on Markdown and are adept at reading, writing, and editing it without additional training or customization. In addition, many popular document authoring tools natively support Markdown, making it a natural choice for curriculum authors.

The repository also acts as an automatic context boundary (or “harness”) for agents. Modern agents use standard file-based tools (e.g., `ls`, `find`, `grep`) to navigate and search within this boundary. Many agent systems allow repository-specific instructions to be provided via a file such as `AGENTS.md`, and expose tools for code exploration and editing. In addition, many modern agents maintain lightweight embeddings of repository contents, effectively creating a semantic index for efficient search and natural language retrieval.

Another important feature of Markdown is its support of hyperlinks. Agents can follow these links to navigate the curriculum and to understand the relationships between different parts of the repository. This is a powerful capability for bringing in additional context simply by following links. On the tooling side, as shown in Figure 3, this allows for a knowledge graph to emerge naturally from the curriculum content as it is built.

While simply adding and linking Markdown files in a repository provides benefits out-of-the-box, we found that applying domain-specific “type hints” produced additional inference gains. Because agent-based tools often browse a repository in directory-tree form, we add sub-extensions to markdown files by encoding their role in the filename. For example:

- `cs1.outcomes.md` indicates a file containing course learning outcomes
- `lecture-1.slides.md` indicates lecture slides
- `week-4.lab.md` indicates a lab assignment

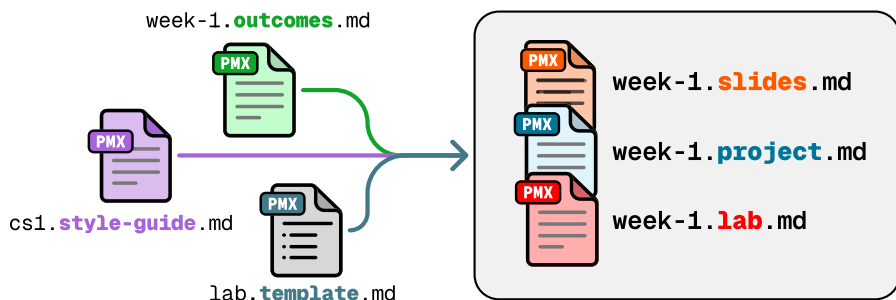


Figure 3: An example markdown file graph with links, type hints, and front-matter illustrating automatic agent inference.

- `project.template.md` indicates a project template
- `week-4.project.spec.md` infers a project specification using the project template

See the companion repository¹ for an example specification file and the resulting output.

We found that agents were often able to infer the necessary context and next actions from file listings alone. This lightweight naming convention provides structure for agent inference while remaining flexible enough to accommodate new material types as the curriculum evolves. In some cases, additional metadata is necessary to guide agent inference and build tools. We use YAML frontmatter as a simple mechanism to attach metadata to Markdown files. This metadata can include learning outcomes, difficulty levels, prerequisites, tags, and other information that supports deterministic processing (e.g., build pipelines) and inference-based behaviors (e.g., personalized recommendations).

Hyperlinks are used to link in the necessary context for both agents and students. Agents will naturally follow links when prompted to perform a task. For example, linking a rubric to a project (e.g., `project-1.project.md` → `project-1.rubric.md`) allows the agent to automatically evaluate work against the rubric without additional instruction.

With these conventions in place, an instructor can scaffold a new course by creating a folder with a small set of Markdown files and high-level instructions to the agent (e.g., “Generate a lab assignment that introduces loops using examples from the cohort’s interests”). Instructors may reuse existing materials as templates or author new materials from scratch. In our deployment,

¹<https://github.com/pathmx/cs1-example>



Figure 4: Sample export pipeline configuration used in our deployment.

agents performed best when expectations and standards were clearly specified in this initial scaffolding. Style guides, example project specifications, outcome descriptions, and type-hinted ontologies provided a robust starting point for agent-assisted authoring in the CS1 context.

Once the initial scaffolding exists, agents can generate a variety of curriculum artifacts, including personalized learning guides, project variants, code review comments, and other course materials. Instructors then review and refine these artifacts using standard code-review tools and workflows (e.g., pull requests, inline comments).

When materials are ready for student use, the same repository serves as the source of truth for export into multiple delivery formats, including PDF, HTML, GitHub repositories, and web applications (see Figure 4). These export pipelines are optional and can be adapted to the constraints of a particular course or institutional environment. Because Markdown is a widely supported source format, PathMX repositories can be integrated with existing tools and learning management systems with relatively little additional infrastructure.

4 CS1 Case Study

PathMX was developed in practice during a CS1 course teaching introductory Python programming to a cohort of 18 students. The class included a wide range of prior programming experience, and most were not computer science majors. The goal was to introduce fundamental programming concepts through project-based learning that mirrored real-world practices. Because of the diverse interests and backgrounds, the course was designed to be highly personalized and flexible. The instructor defined course outcomes at a high level and then tailored them to students’ interests and level using the PathMX methodology described in Section 3.

Throughout the course, when students showed interest in particular tools or instructors identified gaps in understanding, we created projects and tutorials

to support their learning. We recorded student preferences in anonymized “persona” files (e.g., `student-<anon_id>.persona.md`) to avoid sharing personally identifiable information with the agent. This allows for robust personalization while complying with FERPA and other privacy regulations.

The PathMX methodology supported the lightweight collection of qualitative and quantitative information about students’ aptitudes, interests, and learning goals. This information could be pasted or dictated into the agent tool, then automatically placed in the appropriate persona and cohort files. Information about the student’s working environment, such as operating system, IDE, and prior programming experience, was also collected and used to generate more aligned and effective materials. Much of it was collected during lecture ice-breakers (for example, favorite artist, movie, or book), with more detailed information about goals and learning gaps gathered during office hours and direct interactions with the instructor. This kind of rapid transfer of qualitative data to a structured file would have been less tenable with traditional authoring tools.

Once enough information had been collected into personas, agents could perform cohort analysis by querying the repository. A taste profile for the cohort was created based on students’ expressed interests and used to generate themed labs and practice exercises. Based on instructor observation, students showed higher interest and engagement with the themed curriculum than in prior sections of the same course that used more traditional static material. As an example, one student with prior Python experience worked on more advanced projects while keeping the same general theme and goals as the rest of the class. This allowed the student to be challenged at an appropriate level while still participating in shared discussions and activities.

Templates were used to scaffold new labs based on the cohort’s taste profile and the course’s topical flow. We wrote brief lab specification files (e.g., `lab-<number>.spec.md`) that described each lab’s learning objectives, requirements, and constraints. The agent then used the template, the theme, and the taste profile to generate lab materials. We instructed agents to avoid letting thematic elements overshadow core learning objectives. For the CS1 course, the instructor used Deckset, a Markdown-based presentation tool, so lab materials, guides, and quizzes could be created directly from class lectures.

Mid-to-late term projects were presented in a “choose your own adventure” style (reflecting UDL’s multiple means of engagement), where students selected a focus area and completed the project from that perspective. For example, a data science project was offered with Web, Game, or Scientific Computing focus areas. The same datasets were shared across tracks and the students could complete one or more based on their interests. Table 1 shows that students appreciated being able to align projects with their own interests and goals, with

Table 1: Student perceptions of CS1 using PathMX ($n = 17$).

Prior Experience	None: 41%, <1 yr: 47%, 1–3 yrs: 12%
<i>Key Measures (1–5 scale, mean $\pm$ SD)</i>	
Materials relevant to interests	4.00 $\pm$ 0.79
Project choice increased motivation	4.47 $\pm$ 0.72
Themed examples increased engagement	3.82 $\pm$ 0.88
Feedback specific and helpful	4.35 $\pm$ 1.00
Guides helped when struggling	4.24 $\pm$ 0.83

mean ratings of 4.0–4.5 (on a 5-point scale) for materials relevance, project choice motivation, feedback quality, and guide helpfulness.

Differentiated instruction also raises challenges with feedback. To provide high-quality and fair feedback across curriculum variants, we created feedback harnesses (e.g., `<student_id-project_id>.feedback.harness.md`). These harnesses were generated from students’ code pulled into the repository via the GitHub CLI and converted to Markdown using a pre-established template. Each harness included commit history, code snapshots, project specification, a rubric, and agent-specific instructions. Agents generated accurate, specific feedback from these harnesses, and the same artifacts were helpful in human review as a record of student progress. In some cases, feedback was delivered as GitHub Issues with checklists and line-specific comments for each student.

The instructor remained central in this workflow and actively reviewed and iterated on the curriculum and feedback alongside agents. Agentic tools supported side-by-side collaboration and naturally created checkpoints that could be reviewed or rolled back as needed. Agent-based IDEs are continuing to improve these workflows and make a strong copilot for curriculum authoring. The version-controlled repository provided a history of the curriculum’s evolution and captured snapshots of student work for later reference and analysis.

Overall, the PathMX methodology enabled rapid iteration and mid-semester improvements. Student feedback was incorporated into the repository, and agents generated new versions of the curriculum materials in response. This allowed the curriculum to be updated in real time and for students to receive timely feedback on their work. We found that this instructor-mediated on-demand generation was preferable to real-time student-facing agents that lacked editorial oversight or sufficient course-specific context.

5 Student Impact and Educational Integration

PathMX enabled just-in-time instruction and the deployment of curriculum materials to where students were already engaged. Students could access the curriculum in a variety of formats, including GitHub repositories, PDFs, and web applications. A custom integration with GitHub Classroom enabled bi-directional syncing of feedback-based curriculum materials with students' work. The custom integration itself was built with agentic assistance in the same repository as the curriculum materials.

Because the content was presented in the most appropriate medium for the students, a centralized Learning Management System (LMS) was not needed or used. Students were able to access the curriculum and feedback in the context of their own work. This allowed for much more direct and personalized instruction and feedback as the class progressed. In some cases, students would request additional materials for concepts they were struggling with. We generated personalized tutorials and guides for these concepts using the PathMX methodology, which helped them catch up with the class and get back on track. The line-by-line feedback was constructive for students to learn how to refactor and improve their code.

Several students expressed interest in more creative applications of coding, such as gaming or building out web interfaces. We were able to create project variants for these interests, and students were able to choose from a variety of paths to follow. The culminating project was the final project in which students defined their own project requirements with guidance from the instructor. These student-authored specifications were transformed into unique guides for each student. As the project progressed, snapshots of students' work were created at key checkpoints. Guides were then automatically updated to reflect progress, changes, and next steps for the project based on the students' current committed work.

Overall, students were able to engage with the curriculum in a way that was more aligned to their own interests and goals and calibrated to the proximal edge of their learning curve. They received direct, detailed, and actionable feedback throughout the course and the instructor perceived that students progressed more quickly and confidently than in prior offerings of the course.

6 Instructor Experience and Lessons Learned

The benefits of the PathMX methodology were evident in the instructor's experience. The methodology allowed for much more personalized instruction and feedback as the class progressed. In practice, creating personalized assignments and guides took under five minutes of agent interaction plus a few

minutes of instructor review, compared to an estimated 30–60 minutes of manual authoring. Because of the time savings, the instructor was able to focus on higher-level pedagogical decisions and personal attention to students rather than on routine content creation.

For instructors considering PathMX, we recommend the following based on our experience:

- Start with a small set of materials in a version-controlled repository.
- Build a style guide and clearly defined outcomes and scope early and iterate on them as the course progresses.
- Keep students' personal data private, secure, and anonymized for agent interactions.

Despite significant benefits, several challenges emerged. In early projects, when fewer examples were present, agents sometimes generated material that was beyond the students' current level of understanding. More explicit documentation about the scope of the course was created to avoid content drift. This underscores the importance of the instructor's role in vetting material before it is released to students.

Agents were often too helpful and would spell out the solution to the problem rather than guide the student to the solution. In practice, this required explicitly instructing agents not to reveal complete solutions, but instead to provide scaffolded guidance that led students through the problem-solving process without giving away answers.

We invite instructors to explore the PathMX methodology and tools to realize the benefits of personalized, differentiated, and scalable instruction in their own courses. We have provided a sample repository with instructions and guides for how to get started². We also encourage instructors to share their experiences and feedback with us so we can continue to improve the methodology and tools.

7 Future Work

This work represents an initial exploration of agent-assisted curriculum development. Our observations come from a single semester deployment with 18 students. Larger-scale research studies are warranted to formally measure the impact of the PathMX methodology on student learning and instructor efficiency. We are currently designing a controlled study to evaluate PathMX across multiple CS contexts.

²<https://github.com/pathmx/cs1-example>

Additional areas of exploration include assistive automated grading and assessment, the use of PathMX agents for real-time feedback and tutoring, as well as automated skill mapping derived from student code and pre-assessment data. The PathMX methodology also lends itself to generating student-facing support materials that surface and correct common student misconceptions, building on prior work that systematically identifies misconceptions in introductory programming [8].

We are also investigating implications for agent-friendly Learning Object Repositories (LoRs), alternative learning object formats, adaptive learning pathways, and student-facing tutoring agents.

8 Conclusions

Authoring curriculum as structured, version-controlled code enables modern agent-based tools to deliver differentiated instruction at a scale that was previously impractical. PathMX provides a practical path forward, with conventions and workflows that work today while remaining flexible enough to evolve as the technology matures.

Our CS1 case study suggests that this approach makes differentiated instruction more attainable in actual practice. By enabling rapid generation of multiple representations and engagement pathways, PathMX operationalizes UDL principles at scale. In a single semester, we created personalized learning artifacts for each student, generated project variants tailored to different interests, and provided detailed, actionable code review feedback while preserving instructor control and oversight using familiar software engineering workflows. This allowed us to iterate on the curriculum throughout the semester and maintain consistency and quality across all materials. The same conventions that support agent inference, namely type hints, frontmatter metadata, and hyperlinks, also improved human navigation, reuse, and mid-term iteration.

At the same time, this work is exploratory and intentionally modest in scope. Questions about long-term outcomes, instructor workload, and student equity across diverse contexts remain open and motivate the future work described in the previous section. We argue that PathMX provides a practical way forward and a viable design space for next-generation computing education: one that keeps instructors in control but allows for truly scalable differentiated instruction that meets students in their own context.

References

- [1] CAST. *Universal Design for Learning Guidelines version 3.0*. 2024. URL: <https://udlguidelines.cast.org>.

- [2] André Dietrich. “LiaScript: a domain-specific-language for interactive online courses”. In: *Proceedings of the International Conference on e-Learning 2019*. Porto, Portugal, 2019, pp. 186–194.
- [3] Barbara J. Ericson, James D. Foley, and Jochen Rick. “Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems”. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ICER ’18. Espoo, Finland: Association for Computing Machinery, 2018, pp. 60–68. ISBN: 9781450356282. DOI: 10.1145/3230977.3231000.
- [4] Barbara J. Ericson and Bradley N. Miller. “Runestone: A Platform for Free, On-line, and Interactive Ebooks”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 1012–1018. ISBN: 9781450367936. DOI: 10.1145/3328778.3366950.
- [5] Nuno Gil Fonseca, Luís Macedo, and António José Mendes. “Supporting differentiated instruction in programming courses through permanent progress monitoring”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 2018, pp. 209–214.
- [6] Courtney Hsing and Vanessa Gennarelli. “Using GitHub in the Classroom Predicts Student Learning Outcomes and Classroom Experiences: Findings from a Survey of Students and Teachers”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 672–678. ISBN: 9781450358903. DOI: 10.1145/3287324.3287460.
- [7] Maya Israel et al. “Teaching Elementary Computer Science through Universal Design for Learning”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 1220–1226. ISBN: 9781450367936. DOI: 10.1145/3328778.3366823.
- [8] Lisa C. Kaczmarczyk et al. “Identifying Student Misconceptions of Programming”. In: *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. SIGCSE ’10. Milwaukee, Wisconsin, USA: Association for Computing Machinery, 2010, pp. 107–111. DOI: 10.1145/1734263.1734299.
- [9] Simon Larsén and Richard Glassey. “RepoBee: Developing Tool Support for Courses using Git/GitHub”. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’19. Aberdeen, Scotland Uk: Association for Computing Machinery, 2019, pp. 534–540. ISBN: 9781450368957. DOI: 10.1145/3304221.3319784.

- [10] Juho Leinonen et al. “Comparing Code Explanations Created by Students and Large Language Models”. In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2023. Turku, Finland: Association for Computing Machinery, 2023, pp. 124–130. ISBN: 9798400701382. DOI: 10.1145/3587102.3588785.
- [11] Stephen MacNeil et al. “Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 931–937. ISBN: 9781450394314. DOI: 10.1145/3545945.3569785.
- [12] Bradley N. Miller and David L. Ranum. “Beyond PDF and ePub: toward an interactive textbook”. In: *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’12. Haifa, Israel: Association for Computing Machinery, 2012, pp. 150–155. ISBN: 9781450312462. DOI: 10.1145/2325296.2325335.
- [13] Julianna Rodriguez et al. “Courseware as Code Setting a new bar for transparency and collaboration”. In: *2018 IEEE Frontiers in Education Conference (FIE)*. 2018, pp. 1–4. DOI: 10.1109/FIE.2018.8658928.
- [14] Sami Sarsa et al. “Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models”. In: *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*. ICER ’22. Lugano and Virtual Event, Switzerland: Association for Computing Machinery, 2022, pp. 27–43. ISBN: 9781450391948. DOI: 10.1145/3501385.3543957.
- [15] Clifford A. Shaffer et al. “OpenDSA: beginning a community active-eBook project”. In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. Koli Calling ’11. Koli, Finland: Association for Computing Machinery, 2011, pp. 112–117. ISBN: 9781450310529. DOI: 10.1145/2094131.2094154.
- [16] Carol Ann Tomlinson. *How to Differentiate Instruction in Academically Diverse Classrooms*. 3rd. Alexandria, VA: ASCD, 2017. ISBN: 9781416623335.
- [17] Carol Ann Tomlinson. *The Differentiated Classroom: Responding to the Needs of All Learners*. 2nd. Alexandria, VA: ASCD, 2014. ISBN: 9781416618607.

Apache Sudoku: A Web-Based Game for Learning and Demonstrating Web Development Concepts*

Colin Zouck and Unnati Shah
Computer Science
Utica University
Utica, NY 13495

`{cbzouck,unshah}@utica.edu`

Abstract

In today's digital world, understanding how websites function is essential, yet the inner workings of web applications remain unclear to many learners. Practical opportunities to explore client-side scripting, dynamic content, and user interface design in a cohesive project are often limited in the classroom. This paper presents Apache Sudoku, a web-based game designed both as an educational tool and as a demonstration of core web development principles. The application integrates multiple programming languages, including HTML, CSS, and JavaScript, to create an interactive and playable user experience. Apache Sudoku illustrates fundamental software engineering concepts such as modular design, DOM manipulation, event handling, and dynamic content updates. Beyond its entertainment value, the tool provides a hands-on platform for students to explore the integration of front-end technologies and server interactions in a complete, functional application. By combining applied software development with interactive learning, Apache Sudoku exemplifies an innovative approach to teaching web development, supporting student engagement, experiential learning, and practical understanding of modern software engineering practices.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

The project began with defining a clear objective: to explore web development concepts through a practical, interactive application. Web development and cybersecurity principles are central to this exploration, while browser-based games such as Sudoku provide a simple and engaging format for demonstration. After evaluating these concepts, a web-based Sudoku game hosted on an Apache server was selected as the focus. For simplicity, the application is hosted locally, though future versions could be made publicly accessible. The project integrates multiple tools, software, and programming languages to produce a cohesive final product. The central software component is the Apache HTTP server, an open-source platform that enables access to locally or publicly hosted websites. On Ubuntu systems, the default directory for web files is `/var/www/html`, where the `index.html` or `index.php` file serves as the entry point. This file can be edited or replaced to construct a custom website. Additional files placed in this directory are loaded by the locally hosted server. The application leverages a combination of HTML, PHP, JavaScript, and CSS to create a rich, interactive user experience. Apache primarily serves to display the index file in the browser, which in turn links to subscripts and resources to form the complete Sudoku application. This setup demonstrates how multiple web technologies can work together within a single server-hosted environment to deliver a functional, interactive application.

2 Methodology

The methodology for this project focused on selecting an appropriate web server and technology stack to deliver a fully functional Sudoku game. The software needed to be straightforward to install, configure, and maintain. Apache HTTP Server was chosen for its simplicity, reliability, and open-source availability. The `index.php` file was selected as the main entry point instead of a standard HTML file to allow potential integration with a database for storing player data. JavaScript was used to implement the game logic and interactive features. Once the stack was established, system design, feature planning, and implementation proceeded in an iterative development process.

The project leveraged knowledge from a previously implemented Sudoku prototype in Python to guide the JavaScript implementation. Core features essential for a functional Sudoku game were identified as follows:

- Random puzzle generation
- Solution validation to check correctness
- Timer functionality to track puzzle-solving duration

JavaScript was used to implement these features, requiring a detailed study of syntax, event handling, and DOM manipulation [kennyycopcoding2022]. The iterative approach ensured that each feature was implemented and tested systematically to achieve expected behavior.

The project directory, located at /var/www/html, follows a modular structure:

1. PHP index file: Serves as the main entry point, organizing page content and invoking the Sudoku game script.
2. CSS file: Styles the webpage and game interface to ensure a visually appealing and user-friendly layout.
3. JavaScript file: Implements the complete Sudoku game, including puzzle generation, validation, and interactive user input.
4. Image directory: Stores graphical elements used for site design or game buttons, referenced in the CSS or PHP files.

No additional server-side modules were required, ensuring simplicity for local deployment while maintaining flexibility for future enhancements or the integration of advanced features [kennyycopcoding2022].

3 Demo Description

The design process began with the default Apache index file, which provided a testable base for integrating the Sudoku game. The interaction between PHP/HTML files and JavaScript sub-scripts was established, following best practices for client-server communication [geekcoder22982025]. The Sudoku board is represented as a matrix, initially populated with predefined values for testing. A board generation function converts this matrix into HTML elements displayed in the browser. Zeros represent empty cells to be completed by the user (Figure 1). Validation functions, including checkBoard and checkSolution, verify each row, column, and 3x3 subgrid to ensure puzzle correctness (Figure 2). Boolean expressions are used to confirm valid input. Iterative testing refined the layout, sizing, and readability of the board in the browser, including clear demarcation of rows, columns, and subgrids (Figure 3). After establishing basic functionality, additional features were integrated sequentially:

1. Random board generation
2. Timer for puzzle-solving duration
3. Local leaderboard to track user performance and solve times (stored in the browser with potential for database integration via PHP)

```
function generateBoard() {
  const board = document.getElementById('board');
  board.innerHTML = '';

  for (let row = 0; row < 9; row++) {
    let rowElem = document.createElement('tr');

    for (let col = 0; col < 9; col++) {
      let cellElem = document.createElement('td');
      let input = document.createElement('input');
      input.type = 'number';
      input.min = 1;
      input.max = 9;

      let value = sudokuBoard[row][col];
      input.value = value !== 0 ? value : '';
      input.disabled = value !== 0;

      input.id = `cell-${row}-${col}`;

      // Add keyboard navigation and input validation
      input.addEventListener('keydown', handleCellKeyDown);
      input.addEventListener('input', validateInput);

      cellElem.appendChild(input);
      rowElem.appendChild(cellElem);
    }
  }
}
```

Figure 1: A Board Generation Function

```
function isValidMoveInBoard(board, row, col, value) {
  // Check row and column
  for (let i = 0; i < 9; i++) {
    if (i !== col && board[row][i] === value) return false;
    if (i !== row && board[i][col] === value) return false;
  }

  // Check 3x3 box
  let boxRow = Math.floor(row / 3) * 3;
  let boxCol = Math.floor(col / 3) * 3;

  for (let r = boxRow; r < boxRow + 3; r++) {
    for (let c = boxCol; c < boxCol + 3; c++) {
      if ((r !== row || c !== col) && board[r][c] === value) return false;
    }
  }

  return true;
}
```

Figure 2: Validation Functions

4. UI and design improvements for enhanced user experience

The random board generation leveraged algorithms from the initial prototype and existing Sudoku implementations, adapting them to the JavaScript environment. Continuous testing and revision ensured that all components worked cohesively and could be maintained or extended in future iterations.

Following multiple revisions, the game proved fully operational, with all implemented features performing as intended. For simplicity, the project is hosted locally on the machine where it is installed, although steps could be taken to enable public hosting in the future. The demonstration highlights that the core objectives of the project have been achieved. The fully revised site, with properly applied CSS styling, presents a visually clear and user-friendly interface. Each component in the website directory, including the PHP index file, CSS, JavaScript game logic, and image assets, operates seamlessly in coordination, ensuring smooth interaction and functionality. The successful integration of these files validates the design and implementation strategy, confirming that the Apache server effectively serves the web-based Sudoku game while maintaining modular and scalable architecture [mozillacontributors2025].

Figures 4 and 5 illustrate the final contents of the index.php file, providing

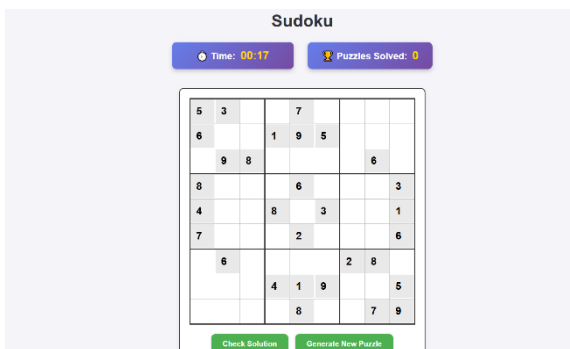


Figure 3: Random Board Generation

a detailed view of the website's underlying structure and layout. These figures highlight the backend organization of the HTML/PHP file, demonstrating how site components are linked and rendered. The HTML integrates seamlessly with the `sudoku.js` file, allowing dynamic interaction for the game board, puzzle validation, and timer functionality.



Figure 4: index.php file

A banner section, stored in the `images` subdirectory, is displayed at the top of the site—currently showing an American flag as a proof of concept and test of image integration within the HTML structure. This demonstrates the ability to include external assets and maintain a cohesive front-end layout. The directory structure within `/var/www/html/` is organized as follows:

```

/var/www/html
├── index.php
└── style.css

```

```

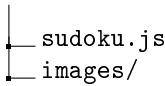
<!-- Leaderboard Section -->
<div class="leaderboard-container">
  <h2> Top 20 Fastest Times</h2>
  <div class="leaderboard-content">
    <ol id="leaderboard-list" class="leaderboard-list">
      <li class="leaderboard-item">
        <div>
          <div>Rank</div>
          <div>Time</div>
          <div>Player Name</div>
          <div>Score</div>
        </div>
      </li>
    </ol>
  </div>
</div>

<!-- New Input Model -->
<div id="new-model" class="model">
  <div class="model-content">
    <h3> Congratulations!</h3>
    <p>You solved the puzzle!</p>
    <p>Time: <span id="time"></span> / Final Time: <span id="final-time"></span> / Attempts: <span id="attempts"></span> / Score: <span id="score"></span> / Language: <span id="language"></span> / Main Computer: <span id="main-computer"></span> /
    <div class="model-buttons">
      <button onclick="submitScore()" class="submit">Submit Score</button>
      <button onclick="skipBoard()" class="skip">Skip</button>
    </div>
  </div>
</div>

<script src="main.js"></script>

```

Figure 5: Backend Organization of the index.php file



This structure illustrates a modular approach, where the HTML/PHP file provides the site framework, the CSS file manages styling, the JavaScript file handles game logic and interactivity, and the images directory stores visual assets. The separation of concerns enhances maintainability and supports future feature expansion.

4 Educational Integration

The Apache Sudoku project was completed as an undergraduate student project, with the student actively engaged in all phases of development, including design, implementation, testing, and evaluation. This hands-on involvement allowed practical application of web development concepts such as HTML, CSS, JavaScript, PHP, and Apache server configuration, reinforcing key skills including modular programming, frontend-backend integration, input validation, and dynamic interface design. The project demonstrates how theoretical concepts from classroom instruction such as user interface construction, algorithmic logic for puzzle generation, and validation workflows can be applied in a tangible, functional system. It provides a clear example of the interaction between multiple programming languages and file types to produce an interactive web application.

As a teaching and learning resource, the project supports computer science pedagogy by illustrating problem-solving approaches, debugging strategies, and iterative testing. It enables students to explore and extend the system independently, fostering active learning and critical thinking.

This project is a practical reference for undergraduate students learning web development, bridging the gap between classroom concepts and real-world application while promoting collaborative discussion, experimentation, and peer-supported learning.

5 Conclusions and Future Work

The web-based Sudoku game successfully demonstrates the integration of HTML, CSS, JavaScript, and the Apache web server to create an interactive, visually engaging, and functional web application. HTML provides the structural framework for the game board, CSS enhances usability and visual appeal through responsive and intuitive design, and JavaScript drives the core gameplay logic, including puzzle generation, validation, and user interaction. Hosting the application on an Apache server ensures reliable delivery of all resources and smooth operation in modern web browsers.

This project highlights the effectiveness of combining front-end technologies with a stable server platform to produce a complete, user-friendly application. It also illustrates practical web development concepts, such as client-server communication, dynamic page updates, and modular code organization, making it a valuable educational tool for students learning full-stack development. Several potential enhancements were identified for future development

1. **Public Hosting:** Deploying the game on the internet to allow access without requiring local downloads.
2. **Enhanced Gameplay Features:** Adding a difficulty selector, hint system, and fail-stop mechanism after multiple incorrect attempts to improve user experience and challenge.
3. **Database Integration:** Connecting the PHP backend to an SQL database to store user scores and gameplay data, demonstrating persistence, retrieval, and data management in a full-stack web application.

These improvements would expand the system's functionality and demonstrate more advanced web development practices. Overall, the project provides a concrete example of full-stack web development, illustrating the interplay between front-end and server-side technologies while reinforcing the educational value of hands-on learning in web application design.

References

- [1] firozshaml6a. (2024, June 21). How to use Apache Webserver to host a website? GeeksforGeeks. <https://www.geeksforgeeks.org/websites-apps/how-to-use-apache-webserver-to-host-a-website/>
- [2] geekcoder2298. (2025, July 23). Create a Sudoku Puzzle using HTML CSS & JavaScript. GeeksforGeeks. <https://www.geeksforgeeks.org/create-a-sudoku-puzzle-using-html-css-javascript/>

- [3] Kenny Yip Coding, How to Build Sudoku JavaScript Tutorial. (2022, February 4). [Video]. YouTube. <https://www.youtube.com/watch?v=S4uRtTb8U-U>
- [4] Mozilla Contributors. (2025, November 13). CSS styling basics. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics

Eye-Assist: An AI Visual Navigation System as a Multi-Disciplinary Computing Education Platform*

Nicholas Brown¹ and Kruti Shah²

¹School of Computing and Engineering

Quinnipiac University

275 Mount Carmel Ave, Hamden, CT 06518

{nicholas.brown, kruti.shah}@qu.edu

Abstract

This paper describes Eye-Assist, an AI visual navigation system developed by three undergraduate and one graduate student, and evaluates its role as a multi-disciplinary computing education platform. Eye-Assist converts camera input into prioritized audio feedback for people with visual impairments using YOLO26-based TensorFlow Lite detection, Intel RealSense D435i depth sensing, and a multi-factor priority algorithm. Preliminary developer testing demonstrates that the system achieves real-time inference suitable for pedestrian navigation, with low end-to-end latency. We are currently seeking IRB approval for formal user evaluation with visually impaired participants. We present the architecture, quantitative benchmarks, failure mode analysis, and student learning outcomes spanning Android development, computer vision, sensor fusion, and Agile methodology.

1 Introduction

Environments designed for sighted users present significant navigation challenges for individuals with visual impairments [1]. Traditional tools such as

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

white canes, guide dogs, and GPS-based applications are invaluable but limited: canes and dogs detect only nearby obstacles without semantic context, while GPS apps cannot identify arbitrary obstacles or function reliably indoors ([2],[3],[4]). These limitations raise a question relevant to computing education: can building a real-time assistive navigation system serve as a multi-disciplinary learning platform requiring machine learning, computer vision, mobile and embedded development, accessibility design, and real-time systems engineering?

This paper presents Eye-Assist, a visual navigation system that turns camera input into prioritized audio feedback using a YOLO26-based TensorFlow Lite model ([5],[6]), Intel RealSense depth sensing [7], and a priority-scoring algorithm. The system supports Navigation, Read Mode, and Explain Surroundings modes, running on Android or in a phone–Raspberry Pi hybrid. We describe the architecture and demo, report quantitative benchmarks from preliminary developer testing, discuss failure modes, and reflect on student learning outcomes.

2 Educational Context and Motivation

Eye-Assist began during Spring 2025 as an undergraduate research project with three students each contributing approximately 8 hours per week. During Summer 2025, a graduate student joined and, together with one undergraduate, committed approximately 40 hours per week, while the other two undergraduates continued at approximately 8 hours per week. Development continued through Fall 2025 as part of the graduate student’s thesis, with the project spanning Spring, Summer, and Fall 2025.

The project was motivated by a comparative analysis of existing tools (Table 1). Students reviewed literature and systematically identified feature gaps, informing the system’s requirements. Table columns represent: Real-Time Detection (RT Det.), RGB camera (RGB), Infrared (IR), Distance estimation (Dist.), Light adaptation (Light), Night detection (Night), Navigation (Nav), Explain Surroundings (Expl.), and Read Text (Read).

Table 1: Feature Gaps Identified During Student Research Phase

App	RT Det.	RGB	IR	Dist.	Light	Night	Nav	Expl.	Read
Eye-Assist	✓	✓	✓	✓	✓	✓	✓	✓	✓
BlindSquare	×	×	×	×	×	×	✓	×	×
GoodMaps	×	×	×	×	×	×	✓	×	×
NaviLens	×	×	×	✓	×	×	✓	×	×

The project required competencies in machine learning, computer vision,

mobile development (Android, CameraX), embedded systems (Raspberry Pi), and accessibility design.

3 System Architecture

Eye-Assist is a real-time perception-and-feedback pipeline in four stages—sensing, perception, priority scoring, and feedback—so components can be swapped independently. Figure 1 illustrates the control flow. A lighting analyzer selects daytime (RGB) or nighttime (infrared) detection, then the user activates Navigation, Read Mode, or Explain Surroundings.

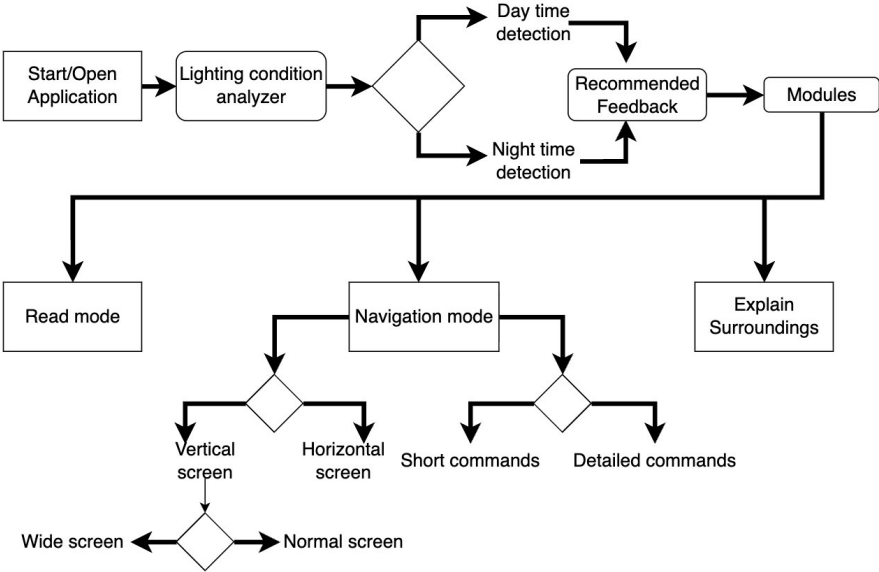


Figure 1: Application control flow with module selection.

3.1 Dual-Model Object Detection

Eye-Assist runs two models in parallel to cover 191 navigation-relevant classes: a primary YOLO26 TFLite model (31 specialized classes, FP16, 320×320 , GPU-accelerated) ([5],[6]) and a secondary YOLOE ONNX model (160 additional classes) [8]. IoU-based deduplication prevents duplicate announcements.

3.2 Depth Sensing and Distance Estimation

An Intel RealSense D435i provides metric depth aligned to the detection stream [7]. Raw depth is converted to “steps” (1 step $\approx$ 0.75 m), producing phrases such as “person, 3 steps ahead.” Objects without depth are still announced by name. An unknown obstacle detector uses depth anomalies alone to catch hazards matching no trained category within a 60-degree forward corridor.

3.3 Tracking, Priority, and Feedback

A ByteTrack-style tracker assigns persistent IDs using an 8-dimensional Kalman filter. Each object’s depth history over 10 frames classifies motion (static, moving, approaching, receding). A weighted priority algorithm scores each detected object based on its speed, distance, position in the field of view, motion state, and detection confidence—so if a trash can, a car, and a tree are all in sight, the algorithm prioritizes the moving car because its approach trajectory poses a greater immediate safety concern. Audio and haptic vibration deliver information through a four-tier suppression hierarchy; track-based deduplication prevents re-announcing static objects. For user convenience, the current system pairs a phone with the camera; future work will explore a Raspberry Pi mounted on a wheelchair for hands-free navigation. The system supports phone-only deployment (RealSense via USB-C) and a Pi 5 hybrid where the Pi handles detection while the phone provides audio over Wi-Fi.

4 Demo Description

The demonstration uses an Android smartphone with an Intel RealSense D435i mounted on a chest or head harness. The phone screen is mirrored to a projector showing live bounding boxes, distance labels, and track numbers. Figure 2 shows the hardware configurations.

The demo proceeds in three phases: Navigation mode via the “hey eye assist” wake phrase demonstrating real-time detection with distance and motion tracking; Read Mode for on-device OCR of printed signs [9]; and Explain Surroundings for scene-level summaries. The hybrid Raspberry Pi setup can also be demonstrated to show consistent behavior across deployment modes. Figure 3 shows detection output across four configurations.

5 Performance Evaluation

During preliminary developer testing using the daytime phone RGB pipeline, we collected performance data across multiple sessions. Table 2 summarizes the key metrics.

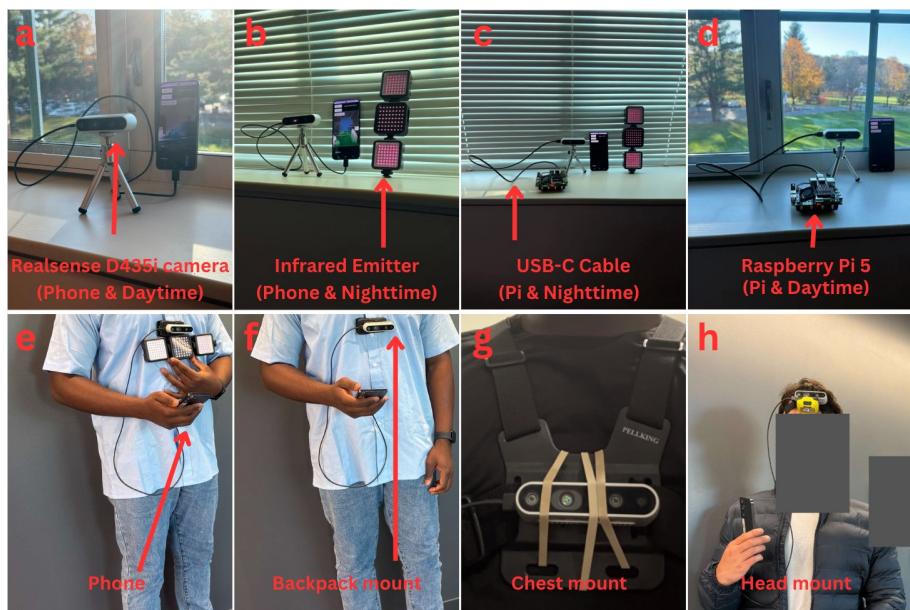


Figure 2: Hardware configurations: (a) Phone daytime, (b) Phone nighttime with IR emitter, (c) Pi nighttime, (d) Pi daytime, (e-f) handheld/backpack, (g) chest mount, (h) head mount.

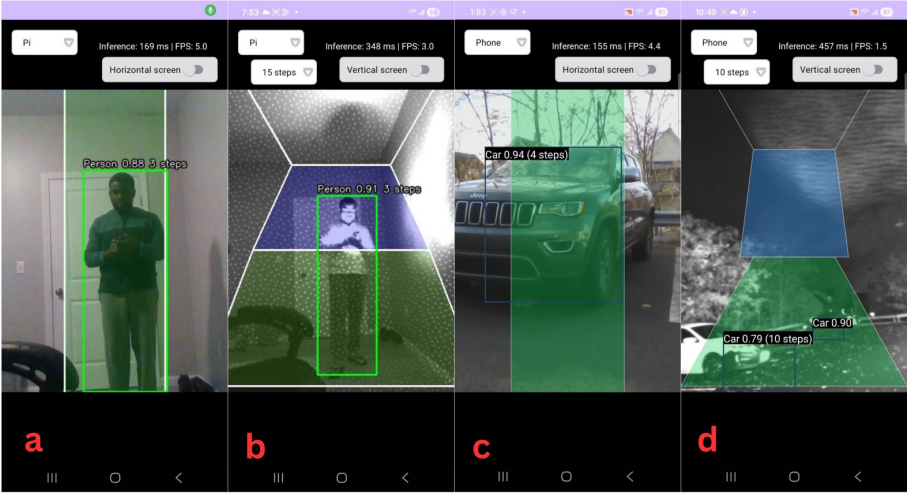


Figure 3: Detection output: (a) Pi daytime, (b) Pi nighttime, (c) Phone daytime, (d) Phone nighttime. Demo video: <https://youtu.be/zlBxsd1Ib4Q>

The mean inference of 96.5 ms (≈ 10 FPS) is sufficient for pedestrian navigation. The distribution clusters between 89–105 ms; periodic spikes at session boundaries represent fewer than 5% of frames. Pipeline overhead averages 38.8 ms ($\sim 29\%$ of total). Navigation mode adds only 2–4 ms versus detection-only, indicating well-isolated workloads.

6 Failure Modes and Error Handling

Given the safety-critical nature of navigation assistance, we identified several failure modes. Missed detections are addressed by the dual-model architecture: if YOLO26 misses an object, YOLOE’s broader vocabulary may detect it, and the unknown obstacle detector uses depth anomalies as a safety net. Depth camera failures use retry-with-backoff (200 ms, doubling to 2 s); objects without depth are still announced by name. False positive overload is managed by four-tier suppression, 3-second cooldowns, and track-based deduplication. TTS suppression failures use a 15-second timeout to automatically restore announcements.

Table 2: Detection Pipeline Performance

Metric	Mean	Std Dev (σ)	P95
Inference time	96.5 ms	7.8 ms	105 ms
End-to-end delay	135.3 ms	15.0 ms	~ 155 ms
Pipeline overhead	38.8 ms	—	—
Objects per frame	2.1	—	5
Max objects/frame	11	—	—
Battery drain	$\sim 1\%/min$	—	—

Table 3: Safety-Critical Response Parameters

Alert Type	Trigger	Response
Vehicle danger	≤ 3.0 m	Time-to-impact + escape dir.
Person critical	≤ 1.5 m	Urgency alert + action
Unknown obstacle	Risk ≥ 0.85	Depth-only corridor alert

7 Student Learning Outcomes

Four students contributed: three undergraduates and one graduate student. Undergraduates reviewed literature, identified gaps in assistive technologies, created and annotated datasets covering day/night conditions, trained detection models, and integrated depth with detection outputs. The graduate student strengthened the architecture, formalized evaluation, and improved night-time IR detection. None had prior Android experience; upskilling in Kotlin, CameraX, and TFLite required self-directed learning. Students adopted Git and Agile methodologies.

Several technical challenges produced notable learning moments. Night/IR detection required improvising with external IR emitters when the RealSense’s built-in emitter proved insufficient. Inference optimization taught the speed–accuracy tradeoff: reducing input from 640×640 to 320×320 cut inference time by nearly 50%. Object tracking identity revealed that position-hash bucketing broke for moving objects; implementing ByteTrack taught temporal reasoning beyond frame-level detection. Priority algorithm tuning required iterative field testing—a practical lesson in human-centered design. Integrating the Intel RealSense camera with Android was another significant challenge, as the RealSense SDK has limited native Android support and required custom bridging between the depth pipeline and the app’s detection framework. Eye-Assist remains a work in progress with ongoing development addressing these engineering challenges.

8 Discussion and Conclusion

Eye-Assist demonstrates that a real-time visual navigation system can run on accessible hardware with low latency, providing distance-aware, prioritized audio guidance without cloud connectivity ([5],[7]). The project spans the full AI system lifecycle and served as an effective multi-disciplinary education platform. Students encountered tradeoffs involving speed, accuracy, sensor noise, and user experience—challenges rarely present in structured assignments. The system has not yet been evaluated with visually impaired users; we are currently seeking IRB approval to conduct formal user studies. The current application is designed for outdoor navigation; future work will implement indoor navigation using map integration via voice commands. Additional planned features include currency recognition and packaging the code as a reusable capstone framework.

References

- [1] World Health Organization. *Blindness and vision impairment*. <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. 2023.
- [2] Bineeth Kuriakose, Raju Shrestha, and Frode Eika Sandnes. “DeepNAVI: A deep learning based smartphone navigation assistant for people with visual impairments”. In: (2023).
- [3] Wafa Elmannai and Khaled Elleithy. “Sensor-based assistive devices for visually-impaired people: Current status, challenges, and future directions”. In: *National Library of Medicine* (2017).
- [4] G. Galatas et al. “Virtual navigation for blind people: Transferring route knowledge from virtual to real environments”. In: *International Journal of Human-Computer Studies* (2020).
- [5] Ultralytics. *YOLO26 Documentation*. <https://docs.ultralytics.com/models/yolo26/>. 2025.
- [6] Ranjan Sapkota, Rahul Harsha Cheppally, and Ajay Sharda. “YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real-Time Object Detection”. In: *arXiv preprint arXiv:2509.25164* (2025).
- [7] Intel Corporation. *Intel RealSense Depth Camera D435i Datasheet*. <https://www.intel.com>. 2018.
- [8] Ao Wang et al. “YOLOE: Real-Time Seeing Anything”. In: *arXiv preprint arXiv:2503.07465* (2025).
- [9] Google. *Text recognition v2 | ML Kit*. 2024.

A Model for AI Across the Curriculum at an Urban Community College*

Mohammad Q. Azhar ¹, Rifat Salam ², Elizabeth Wissinger ²,
and Melissa Zavala²

¹Computer Information Systems

²Social Sciences

{mazhar,rsalam,ewissinger, mzavala}@bmcc.cuny.edu

Borough of Manhattan Community College,

The City University of New York

NY,NY 10007

Abstract

The rise of generative AI necessitates faculty development to equip diverse student populations with essential AI skills and knowledge. This paper details the findings of an AI Across the Curriculum (AIAC) initiative at an urban community college, a Hispanic Serving Institution (HSI). The project trained faculty in both STEM and non-STEM disciplines on designing and implementing formal AI learning activities. Qualitative data was collected from 13 focus groups, involving 157 students in Spring 2025 courses taught by AIAC faculty. Students consistently reported the activities were highly instructive, moving them beyond superficial tool use to a nuanced understanding of generative AI, particularly concerning AI ethics and societal implications. These learning experiences fostered increased comfort and confidence with AI technology, enhancing their ability to evaluate information reliability and apply AI in future studies and careers. The results demonstrate the potential of this interdisciplinary approach to increase AI literacy and understanding of AI's ethical implications, while reducing intimidation barriers across a diverse

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

student body. Our focus on training faculty first, who in turn implemented best practices in their courses, builds a foundation for ongoing AIAC activities on our campus. We share this AIAC model here as a practical, adaptable template for Computer Science faculty to collaborate with colleagues and successfully offer tools for expanding AI learning experiences for all students.

1 Introduction

This paper presents data gathered from students at an urban community college serving a diverse student population, where instructors participated in an AI-focused extension of an NSF HSI Implementation and Evaluation funded project entitled: Bridging the Gap: Designing a Technology Learning Community Integrating Computational Thinking to Improve STEM Engagement Across Disciplines. The original project was designed to further research several issues highlighted by ongoing scholarship on barriers to participation in STEM education including the barriers faced by non-STEM students adopting new technologies in their learning and unequal participation across gender, race, and socioeconomic lines. Our original study researched how implementing computational thinking (CT) strategies in both formal and informal learning settings could impact students facing such obstacles [1].

The student data is a culmination of an initiative in which our project team trained faculty to implement AI learning activities in formal learning experiences for STEM and non-STEM students through an AI Across the Curriculum (AIAC) initiative aimed at faculty in an urban community college. The teaching load of full-time faculty is 4-4. The data presented here was collected from students in courses taught by the AIAC-trained faculty participants in our initiative during the academic year 2024-25. The study demonstrated benefits to students through AI learning in the classroom, and activities designed to enhance students' AI literacy. Our AIAC model provides an adaptable template for Computer Science faculty seeking to expand STEM learning experiences through the incorporation of AI for their students, as well as helping their colleagues in other disciplines toward this goal.

The rise of generative AI presented an opportunity to address the growing need for pedagogical faculty development across disciplines to provide AI skills and knowledge to diverse student populations. Institutions of higher education are conceptualizing the AI skills and competencies for developing students' AI literacy [2, 7, 8, 4, 6], and must assess and understand what faculty need [5] to accomplish this. The AIAC initiative was developed as a small-scale, modified version of the AI across the curriculum model implemented at the University of Florida [9] which adapted its use of the AI literacy framework [7] for the community college setting. While on a smaller scale, this project also focused

on training faculty across disciplines and building interdisciplinary engagement to advance students' AI literacy, their understanding of responsible use of AI in their future studies, and their career readiness.

The project team consisted of a PI and co-PI from Computer Science, as well as a co-PI and senior personnel from Sociology. Their combined expertise in technology with and in social science enhanced their ability to work with faculty across disciplines, reducing faculty intimidation and, thus, expanding the reach of our interventions to students in STEM and non-STEM fields. Based on these findings, this paper argues that computer science faculty are uniquely situated to provide this urgent pedagogical support to their colleagues in other fields. Rather than focus on limiting AI or teaching its detection and eradication in the classroom, this project centered on providing positive AI learning experiences that benefit students across disciplines and programs. This paper reports findings that are particularly salient considering the rapid changes to teaching and learning brought on by generative AI. The study focused on qualitative data to provide a nuanced understanding of students' experiences of the learning activities in their own words. These results show the potential benefits of expanding similar multidisciplinary initiatives beyond the usual scope of technology and computer science education.

2 AI Across the Curriculum

Prior to developing this initiative, members of the research team facilitated multiple workshops at the college, where faculty from a wide variety of disciplines expressed concern about students' use of generative AI, as well as uneasiness over their own knowledge gaps and experience with this new technology.

AIAC began as a series of workshops open to all faculty in Spring 2024 facilitated by computer science faculty, the PI and co-PI, co-facilitated by sociology faculty with expertise in pedagogical faculty development. Interested participants were invited to apply to be AIAC Faculty Fellows. They completed a Summer Bootcamp in 2024 and committed to creating and implementing AI learning activities and lesson plans for a pilot in Fall 2024 and full implementation in Spring 2025. The workshops included foundational knowledge about generative AI, an overview of widely available generative AI tools, an introduction to AI ethics (through IBM SkillsBuild [3])) as well as hands-on activities aimed at the development of discipline and course-specific AI course policies, learning activities, and lesson plans. Model activities and lesson plans incorporating principles of culturally responsive pedagogy (CRP) were provided by facilitators for computer science courses and for a sociology course to help participants conceptualize ways of integrating these components into

their courses. All participants were required to develop 1) an AI-infused syllabus with an AI Policy; 2) an AI Ethics learning component; and 3) an AI lesson plan for discipline-specific AI learning.

These learning activities were designed to be low-barrier, open to non-STEM majors, and situated within general education and introductory-level courses. During the 2024-2025 academic year, the AIAC initiative reached a total of 829 students, from 9+ departments including health studies, social sciences, biology, chemistry, math, computer science, and business management.

Faculty across disciplines developed learning activities that incorporated course content and learning goals. Some examples include applying principles of AI ethics to facial recognition technology in a sociology course, incorporating and critiquing the use of AI editing tools in a video arts and technology course, ethical use of AI in human resources management, and using AI tools to review and improve Python projects in an analytic geometry and calculus course. The “across the curriculum” model allowed for the integration of AI learning that would provide students with exposure and experience with AI tools with course-specific concepts and skills.

3 AI Literacy Framework

As noted above, the project’s starting point was the principles of AI literacy used by the University of Florida program [9] which incorporates four dimensions of AI literacy: 1) basic knowledge: knowing and understanding how AI works; 2) using and applying AI: being able to use AI tools to solve problems and complete tasks; 3) evaluating AI and creating AI systems; and 4) AI ethics: understanding the moral and ethical implications and making informed decisions about AI use.

Because this study grew out of a more far-reaching project that explored the technological divide among under-represented minority groups including women at an urban community college, this portion of the study focused on experimenting with the impacts of a Generative AI-infused curriculum in various classes offered by different departments. This part of the project thus focused on how AI tools are employed by STEM and non-STEM students in the context of the persistent hurdles disadvantaged groups face which discourage participation in fields of scientific innovation. Consequently, the GenAI component of this study did not focus mainly on specific matters like plagiarism, citation, and policy-aligned pedagogy, particularly as universities are still grappling with these new technologies in order to construct suitable strategies that work well for diverse stakeholders. Instead, it measured how students can become more engaged with new technologies through a GenAI-infused curriculum that can help them enhance their career skills or imagine new career

alternatives.

There are key differences that we needed to account for in adapting this framework to the two-year college context, particularly as this program targeted faculty primarily teaching introductory-level courses in non-technical fields. Standalone workshops conducted by the research team uncovered the primary needs of faculty, which included the basics of AI literacy for the faculty, many of whom were unfamiliar with the technology and felt ill-equipped to address AI, let alone incorporate AI into their courses. Even those who had some familiarity with generative AI tools, such as ChatGPT, expressed interest in a sustained program where they could learn more about AI and have an opportunity to develop AI-related activities appropriate to their course and discipline. Thus, this effort focused on the aspects of AI literacy that did not involve technical competencies, such as coding and programming, and instead focused on 1) developing the AI literacy of participating faculty, and 2) having faculty develop course-appropriate AI learning activities and lessons which focused on understanding the basics of AI, using AI appropriately, i.e., to enhance course learning goals within the bounds of academic integrity understanding the basics of AI ethics, knowing the fundamental principles of AI ethics and in using and AI applying tools ethically.

Given the level of knowledge and preparation as well as the multidisciplinary makeup of faculty participants along with the emergent nature of the technology, the findings reported in this paper focus on students' experience of the learning activities across disciplinary contexts.

4 Methodology

The larger project employed mixed-methods research design and received the institution's approval. The results here, from the qualitative part of the project, were gathered according to the protocol for Human Subjects Research. Focus group interviews and observations were conducted by a research team consisting of a sociologist and a research assistant. The classes consisted of on campus and synchronous online classes taught by AIAC faculty participants. This paper focuses on this research question: *Do factors such as formal knowledge about AI, prior hands-on experience, and the intersection of gender, race, and socio-economic status influence whether integrating AI into a formal learning environment 1) enhances students' ability to solve problems, 2) reduces intimidation, and 3) enhances student interest in AI?*

The focus group interview questions centered on the activities in Spring 2025, after full implementation of the learning activities developed through AIAC. Open-ended questions allowed students to raise matters beyond those they were asked.

1. How would you describe your knowledge or understanding of generative AI tools (e.g., ChatGPT, Gemini) before doing these activities, and how would you describe your knowledge and understanding now that you have completed the AI learning activity or activities in this class?
2. How would you describe your knowledge or understanding of AI Ethics before doing these activities, and how would you describe your knowledge and understanding of them now that you have completed the AI Ethics learning activity in this class?
3. What were some of the challenges you experienced while learning about AI or while completing the AI learning activities in this class?
4. How do you think you could apply what you learned about AI after you finish this class in your future studies or career plans?
5. How can we help you and other students learn more about AI to support your future studies or career plans? For example, offering AI workshops, having AI learning activities in more classes, for all majors, providing more opportunities to participate in innovation challenges or internships. Please share any suggestions you might have.
6. Is there anything else you would like us to know?

4.1 Data Gathering Methods

The qualitative team conducted 13 focus groups with 157 students (a subset of Spring semester participants). Three of these sessions were conducted virtually, the rest were held in-person. Four focus group sessions were conducted by an interviewer working with an observer. All focus group sessions were recorded and transcribed. Verbal consent was obtained before the start of interviews after which facilitators read the focus group questions aloud. On average, the sessions lasted 21 minutes. Audio recordings were transcribed and analyzed by the researchers, using Atlas.ti, a qualitative data analysis program. Codes were created using inductive methods.

5 Research Findings and Discussion

Overall, students reported that engaging with AI learning activities provided an opportunity for them to not only use AI tools, but to broaden their understanding of the multifaceted aspects of the technology, including important issues of AI ethics and an awareness of the challenges and opportunities that AI technology brings in both educational and future career contexts. With regard to ethics in particular, student responses ranged from the usual “obviously

don't use it to completely cheat” to a more nuanced grasp of the behind the scenes effect of corporate uses of AI in hiring, for instance. The subsequent impact of that understanding paved the way for an increased sense of confidence, with students attesting a new level of comfort “with the idea of it [AI] being integrated into an academic setting” and having “a better knowledge of it now,” as well as increased interest in and engagement with STEM/AI technology in their future lives and careers.

5.1 Qualitative Results and Discussion

Focus groups for the Spring 2025 Formal Learning Activities consisted of 13 classrooms offering formal generative AI learning components. These classrooms ranged in disciplines from Video Arts and Technology to Sociology, Biology, Chemistry, Math, Speech, Business Management, and Health Studies. Focus groups respondents reported a mix of expertise with generative AI tools, with the majority reporting ease of use of the technology. Students detailed their experiences, as cited below, illustrating the effectiveness of the new materials introduced this term.

Tree map response frequencies along codes indicate the prevalence of positive responses over negative or neutral ones to questions about the impact of the new activities in the formal learning classroom setting (Figure 1).

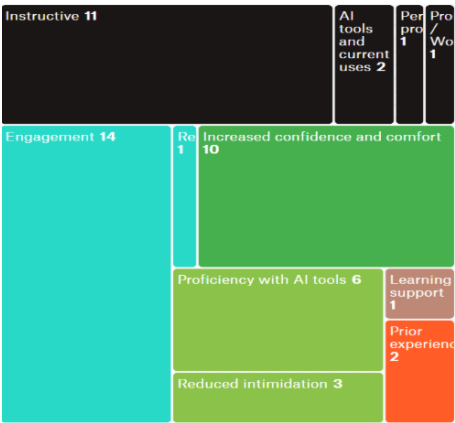


Figure 1: Tree Map about impact of the new activities in the formal learning classroom setting

Students in all classes overwhelmingly and consistently reported that the activities they participated in were instructive, a theme that appeared consistently in the tree graph. They indicated that the activities helped them

prepare for a more informed use of generative AI tools. Students illustrating this point said this in different ways:

“But, so I’m now very comfortable with the idea of it being integrated into an academic setting in a way I wasn’t before . . .” – Chemistry student.

A Business student said, “I mean, it was pretty straightforward from the beginning for me, but I feel like I just have a better knowledge of it now.”

Below are three student quotes that compellingly demonstrate the practical instruction students received which led to their more critical assessments of these new resources. A student from a media course put it this way:

“I feel like, yeah, I think AI ethics are pretty straightforward. It’s just, it was pretty—I feel like a lot of things dealing with AI goes, goes without saying just because, you know, obviously don’t use it to completely cheat on—use it for, don’t violate . . . I guess what a school integrity or your student integrity, . . . you know, don’t plagiarize with it or . . . Yeah, those are, those are the two main things, but I feel like those are, like, the biggest questions when it comes to AI, and then things like privacy all, and, sort of accessibility to what you have are also big things. So, just keeping those in mind and its restrictions.” A Chemistry student said: “I do have a lot of problems with AI and that I, I think it’s not very ethical, both because of the way it sources its data, and because of, like, the environmental impacts of it.” Another student in the same discipline added, “Before this project I just used AI and not knowing any ethics or ethics side of the AI use. But now with this project, I learned about how AI shouldn’t be used in a certain way. And yeah, I gained more knowledge about that ethic, the ethic[al] side of it.” There were several other instances like these where students made it clear that they had arrived at the problematic aspects of these new technologies and understood the ethical implications in more organic and meaningful ways.

These valuable lessons tackle the matter of sound pedagogy in hands-on ways even while the study did not necessarily set out to explore the problem of plagiarism and the best policies to address it more specifically. These ethical concerns indicate there is still much to explore from various vantage points, prompting additional research. On the other hand, a few students reported finding the activities to be an ill fit for class, adding that there was a need for additional training, especially for other students, as well as in more classes. One student noted, for example, that generative AI may interfere with the social ties that people regularly count on for learning. She wondered how human interactions can be strengthened, especially when generative AI encourages work in isolation.

Biology Student: “You know, using AI at the same time, that also promotes the human interactions, you know? So, yeah, that’s what I want to know. And I would like to have more, maybe watch [some] video[s] about, you know, yeah,

related to this.”

Comfort and ease of use along with new uses, and a background on ethical matters related to the use of generative AI in the workplace led to an increased interest and broader usage of the new tools introduced. One female student said, “Honestly, I’m not a computer science major at all. Before taking this class, I didn’t even know much about AI, so I do believe in some sense that it did help me, but I feel like I did have the general idea of AI, but the ethics part, that part, I really didn’t know about it at all, so . . .”

Tree map response frequencies along codes indicate that positive responses were more highly represented than negative or neutral ones to questions about the impact of formal learning activities incorporating generative AI tools (Figure 2).

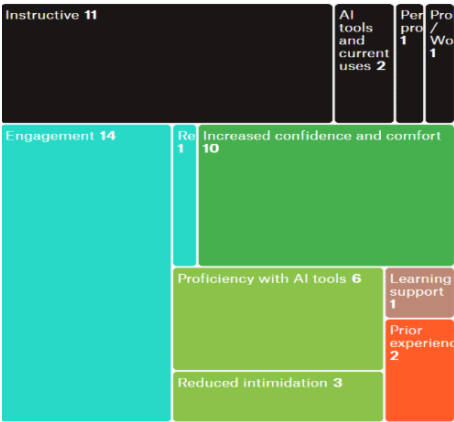


Figure 2: Tree Map about the impact of formal learning activities incorporating generative AI tools

Students expressed a deepening of their knowledge of generative AI as well as increased comfort levels with the different applications used in their classrooms, from Python, to Photoshop, SEEK, and ChatGPT. A biology student said, “But now, being in this class, and using the tool that we did, I learned that AI can be very interactive and it allows you not to do the work for you, but also, you would have to put in the work, and it doesn’t take away your focus of learning, it enhances it, in fact, and it allows you to put in your own ideas and, I don’t know.” A computer science student in a sociology course also expressed improvements to her familiarity with generative AI tools and her considerations of new ethical matters saying, which include new knowledge they will be able to apply in career settings: “I think, like, my knowledge on AI ethics wasn’t very good. I didn’t really know much about it at all until I took

the course and I've seen so many different things that goes into play, especially when it comes to even just companies using AI tools, and one of the examples that was used was, they were kind of selecting employees, to basically give a promotion to, and the AI would choose a specific person. . . . so something I had to learn was . . . AI is prejudiced, so I was like, OK, there's prejudice in the back end when it comes to AI. So, I think that's one of the things that I've personally learned from it, and you know there's actually a lot that goes into play, you know? Behind the scenes, you know? Not just for our use, but even just for companies and different things like that, you know?" Focus group participants noted that their classroom activities enhanced and reinforced their ideas about generative AI tools even while they expressed uncertainty regarding the extent to which they would use these tools in their future careers. Regarding the research question, overall, the research team found the following:

Students' sense of their ability to solve real-world problems was overall enhanced by increased familiarity with generative AI tools. Another important finding was that regardless of prior knowledge, most student participants had a positive experience of our classroom activities teaching about AI. The following quotes from students in classrooms across the different participating disciplines highlight the positive experiences gained through the introduction of generative AI-centric activities. A Speech student said, "I think it is very useful in researching different topics and also, making sure to make, that you see different viewpoints, that you might have not seen before, and different mindsets, and that's it." A Business student added: "OK, so before, I knew what AI was, but I feel like during this class we kind of broke AI down, so like, different kinds of parts and find out what it can really do to help benefit society and stuff."

Students were asked during focus group sessions if there was anything else they wanted us to know which tended to yield questions about AI use in their classes as well as additional suggestions beyond how the college could further support their learning. In some cases, students made generalized comments about the place of generative AI in society and its impact on humanity at large, at times also linked to their concerns about what they see as the stakes involved in AI use to these overarching principles. These comments were often framed by a shared notion that artificial intelligence is here to stay, thus society must engage with it. Such engagement can entail additional improved instruction, like the one they experienced over the course of their semester, but also in terms of expanding ethical guidelines beyond those promoted for consumer adherence. Students usually agreed that generative AI is the way of the future and that society has been changed by this. One Speech student put it this way: "But the thing is, AI is all around the world now, so it's normal, so that's why everybody uses it, so, yeah." Other students worried about the implications of

a world with a learning gap so wide that those with access to resources might further disadvantage those already on the margins, as the IBM Skills Build activity experienced in classes made clear. A Sociology student phrased her thoughts by saying: “Yeah. Well, I think overall, AI is a super powerful tool in the hands of the right person, you know, [be]cause it can be abused, and if you use it correctly, you can use [it] to help you study or, like, you were saying earlier, don’t have it give you the answer, have it work through the answer with you, you know? It’s something that I feel is inevitably gonna be integrated into society, whether you like it or not. It’s just such a good tool, and, yeah, that’s pretty much it.” These critical insights, embedded in an understanding of power differentials in society, found resonances in students in different classes. The potential for intensified social inequality and injustice was a poignant motif that emerged from students’ concerns about new technologies with much potential for improving social conditions, but could easily lend themselves to misuse.

Yet despite the concerns and ambivalence, students expressed an increased sense of confidence in their ability to engage with generative AI in educational, career and personal contexts. This is expressed enthusiastically by a business student who shared: “So, I was, like I previously mentioned before, I’ve gotten some insight on AI . . . in other classes, and so the fact that I’m getting more insight on AI is actually really good because as we advance as a society and civilization, AI is becoming way bigger, and it’s definitely become, become bigger in the past few years, and it’s just going to get bigger from here. So, the fact that we’re learning about it in real time and in classes such as this one, it just gives greater insight and knowledge on AI and the fact that we’ll be able to apply this into future positions and studies is actually a really good thing, and that we’re learning it in school.” A biology student notes the ways in which the AI learning activities will help in his future plans: “I do want to focus on conservation so there’s a lot of stuff with AI, you know, the good and the bad, seeing what it can do and knowing its limitations is definitely something that I can take on to a career in biology and conservation.”

5.2 Discussion Summary

The general results from classroom activities suggest that the original objectives of familiarizing students with new AI tools were achieved during formal learning activities in the classroom. Most students expressed significant comfort with the technology and expressed appreciation for having the opportunity to learn more about and/or to use AI tools as part of their classes. While the actual course learning activities, and the AI literacy skills being taught varied, students reported gaining some new skills, experience or insights about generative AI tools and making informed choices about their appropriate use.

And yet, despite the study's aims not being centered on matters of plagiarism, student testimonials nevertheless yielded promising insights into their experiences, suggesting that the ethical questions at stake in using GenAI in college classrooms were nonetheless addressed in practice if not in theory. By infusing regular curriculum in multiple fields with GenAI instruction, students were able to more directly test new technologies. In doing so, they consistently found AI sources to be unreliable and unwieldy, leading to collective nuanced realizations about the need for ethical measures and restrained uses of artificial intelligence. Most students gained a better understanding of how to use generative AI tools, their uses and implications. In many cases, even where students had prior learning and knowledge of AI, they learned more than they would have through their independent use of these tools. This was particularly true with AI ethics, a topic that few students knew about prior to the specific learning activities implemented into their classes by faculty who participated in AIAC.

6 Conclusion

The goal of this paper is to share findings about the impact of training faculty across a variety of academic and career fields to incorporate discipline and course specific AI learning activities into their courses. While the activities varied, faculty in STEM and non-STEM fields were given an opportunity to develop their own AI skills and knowledge through this pedagogical training. Sharing the results of students' experiences of the tools created in this training here as a potential model for CS faculty to contribute their expertise through interdisciplinary collaboration that benefits students and faculty in STEM and non-STEM fields. This model illustrates the benefits of interdisciplinary collaborations to foster pedagogical support for their colleagues, to facilitate AI learning experiences that benefit students across disciplines and programs, and better prepare them for their future studies and careers.

Acknowledgements

This work was supported by the U.S. National Science Foundation under Award #2122690. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

References

- [1] Mohammad Q. Azhar et al. “Implementation and Evaluation of a Virtual Hackathon in an Urban HSI Community College during COVID-19”. In: *Journal of Computing Sciences in Colleges* 38.6 (2023). URL: <https://par.nsf.gov/biblio/10547076>.
- [2] Dragan Gašević, George Siemens, and Shazia Sadiq. “Empowering learners for the age of artificial intelligence”. In: *Computers and Education: Artificial Intelligence* 4 (2023), p. 100130. DOI: 10.1016/j.caeai.2023.100130.
- [3] IBM. *IBM SkillsBuild*. <https://skillsbuild.org/>. Accessed: 2025-02-xx.
- [4] M. Kassorla, M. Georgieva, and A. Papini. “AI Literacy in Teaching and Learning”. In: (2024). URL: <https://www.educause.edu/content/2024/ai-literacy-in-teaching-and-learning/introduction>.
- [5] René F. Kizilcec. “To advance AI use in education, focus on understanding educators”. In: *International Journal of Artificial Intelligence in Education* 34.1 (2024), pp. 12–19. DOI: 10.1007/s40593-023-00351-4.
- [6] Katherine A. LaFlamme. “Scaffolding AI literacy: An instructional model for academic librarianship”. In: *The Journal of Academic Librarianship* 51.3 (2025), p. 103041. ISSN: 0099-1333. DOI: 10.1016/j.acalib.2025.103041. URL: <https://www.sciencedirect.com/science/article/pii/S0099133325000370>.
- [7] D. T. K. Ng et al. “Conceptualizing AI literacy: An exploratory review”. In: *Computers and Education: Artificial Intelligence* 2 (2021), p. 100041. DOI: 10.1016/j.caeai.2021.100041.
- [8] S. Ocen et al. “Artificial intelligence in higher education institutions: Review of innovations, opportunities and challenges”. In: *Frontiers in Education* (2025). URL: <https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2025.1530247/full>.
- [9] J. Southworth et al. “Developing a model for AI Across the curriculum: Transforming the higher education landscape via innovation in AI literacy”. In: *Computers and Education: Artificial Intelligence* 4 (2023), p. 100127. DOI: 10.1016/j.caeai.2023.100127.

Experiential Educational Ethics Activities in Undergraduate Education: Instructor Observations*

Ursula Parker¹, Farzana Rahman², Ashish Aggarwal³,
Samuel Malachowsky¹, Daniel Krutz¹

¹Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623

`{ucp4496,samvse,dxkvse}@rit.edu`

²College of Engineering and Computer Science
Syracuse University
Syracuse, NY 13244

`frahma02@syr.edu`

³Herbert Wertheim College of Engineering
University of Florida
Gainesville, FL 32611

`ashishjuit@ufl.edu`

Abstract

Although teaching ethical computing practices is essential for developing responsible technologists, ethics is still too frequently overlooked in computing education. As a consequence, students may graduate not only without the skills to identify and address ethical dilemmas, but also without an understanding of why ethical decision-making is essential, in the first place. To overcome this gap, we have created a set of

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

easily adoptable, experiential ethics-focused Accessible Learning Labs designed to systematically introduce students to core ethical concepts in computing and to emphasize the real-world consequences and importance of ethical computing practices. In the following paper, we report on instructor observations regarding the inclusion of these ethics-focused computing labs at a diverse set of categorically distinct partner institutions. The complete project materials are openly available on our website: <https://all.rit.edu>

1 Introduction

An increasing number of decisions that profoundly affect society—from hiring and healthcare to policing and information access are now driven by computing systems. Unfortunately, many of these systems are developed without adequate consideration of ethical implications such as bias, fairness, privacy, autonomy, and accountability [13]. While computing educators recognize the importance of teaching ethics, research shows that they often struggle to integrate ethics into their courses due to limited time, expertise, or access to high-quality instructional resources [13]. Such an approach frequently restricts the inclusion of this increasingly essential topic in computing and non-computing curricula.

To address the gap created by the lack of high-quality, easily adoptable computing-ethics content, we have developed a set of educational resources Accessible Learning Labs. Each lab focuses on a different ethical challenge in computing and is designed to help students both understand why ethics matters and develop practical skills for identifying and addressing ethical issues in real systems. In each lab, the participant will ‘experience’ the addressed ethical scenario. The labs are intentionally created for broad adoption and can be seamlessly incorporated into a variety of instructional contexts, including standalone modules, course integrations, workshops, or independent learning. The only requirements are a web browser and an internet connection, making the labs accessible to instructors, students, and self-guided learners alike. Complete lab materials are publicly available on the project website [1].

Over the last several recent semesters, we have included our labs in a variety of undergraduate course offerings at six universities. Additionally, we have partnered with instructors in several other computing and non-computing focused courses, both at our university and collaborating universities to include our ALL educational material in their curricula. The material was included in both conventional and online course offerings.

This publication summarizes the insights gained from incorporating our materials in various instructional settings. Key takeaways include: I) Instructors valued the modular, à-la-carte structure of the labs; II) The 30–60 minute completion time made adoption easier; III) Both instructors and students appreci-

ated that the labs were fully hosted and self-contained; and IV) Ethics-oriented, empathy-building content appeared to motivate students. Unlike existing work [13, 14, 9] on our Accessible Learning Labs, this submission focuses on instructors’ perspectives and practical experiences using the materials, rather than on statistical evaluations of lab outcomes.

To summarize, this work makes the following contributions:

- **Overview of the developed educational materials:** The ethics-focused, self-contained labs are freely available on the project website [1]¹. They require no special software or configuration and can be used directly through a standard web browser.
- **Initial instructor insights from using the materials:** We present observations from instructors who incorporated our labs, highlighting strengths of the materials and offering useful context for prospective adopters.
- **Practical guidance on effective integration:** We provide recommendations—based on observed classroom use—that may assist instructors adopting the labs or teaching accessibility-related content.

2 Related Work

Ethical issues in computing are increasingly recognized as critical concerns [7, 16]. U.S. accreditation guidelines now mandate ethics instruction in Computer Science programs, yet institutions retain significant discretion over how deeply and in what format these topics are taught [5]. Despite this requirement, there remains a notable shortage of comprehensive, accessible resources focused on the ethical dimensions of AI systems [11].

Scholars contend that embedding ethics within technical coursework helps normalize these conversations and situate them in authentic contexts, rather than treating ethics as ancillary or isolated [4]. Proposed approaches include project-based activities that highlight real-world consequences [2] and the use of science fiction to prompt reflection on future ethical challenges [5].

Recent work has begun exploring concrete strategies for integrating ethics into AI/ML education [15]. For example, [5] analyzed syllabi to examine how ethics is incorporated through standalone courses and technical AI classes, while [6] interviewed instructors to surface key challenges and highlight the need for a community-driven, interdisciplinary framework for teaching AI ethics. Our contribution builds on this emerging literature by offering the first web-based, experiential activity designed specifically to help students identify and understand AI/ML bias.

¹Note: Anonymizing the website for this submission was not feasible, as it is actively being used by adopting institutions throughout the United States.

Interest in AI ethics education extends beyond computer science, with fields such as medicine [8] and public policy [12] also emphasizing its importance. [3] similarly calls for “Ethics of AI-aware software engineering,” which integrates stakeholder values and ethical reflection across the development lifecycle.

Our labs adopt experiential learning as its foundation, an approach that has been shown to be an effective pedagogical method [10]. Although experiential learning has been used in a variety of educational settings [13, 14, 9], our work is distinct in its explicit focus on enabling learners to directly engage with and reflect upon bias in AI/ML systems.

3 Lab Structure, Topics and Goals

In the following sections, we will describe the primary objectives of the labs and their structure.

3.1 Lab Goals

The educational labs Accessible Learning Labs have several primary goals:

1. **No specialized tools are required:** Complete materials are hosted online and can be accessed with only a browser and an internet connection, allowing for effortless adoption without additional software or hardware.
2. **Accessible to learners with basic programming knowledge:** Students with a range of experience levels should have the opportunity to learn about the importance of ethical software development and gain introductory skills for addressing ethical concerns in computing.
3. **Designed to fit within already packed curricula:** Each lab can be completed in roughly 30–60 minutes, and instructors may flexibly select and incorporate only the components they need. This modular structure allows the labs to be integrated even in courses with limited available time.
4. **Includes all required teaching materials:** To streamline adoption, each lab provides complete instructional resources, including slides, background readings, hands-on activities, reflection prompts, and assessments.
5. **Emphasizes both the practice and the purpose of ethics:** Learners must not only understand *how* to build ethical systems, but also *why* ethical decision-making is critical in computing.

Figure 1 demonstrates the experiential nature of the labs, where participants encounter flawed software and then repair it. This experiential format has been found effective in previous publications [9, 13, 14].

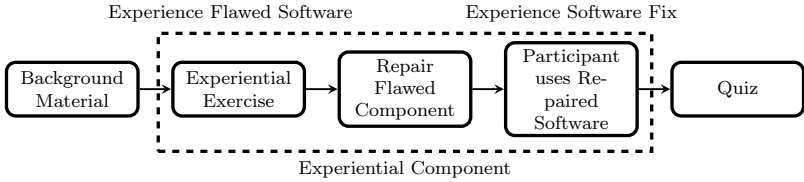


Figure 1: Utilized experiential intervention format.

3.2 Lab Structure and Components

Each lab consists of multiple elements to educate and motivate students about core ethical issues in computing. The lab workflow is presented below:

1. **Foundational Ethical Context:** The lab begins with an overview of the ethical topic being addressed, introducing key principles and guidelines that would prevent the ethical failure demonstrated in the activity.
2. **Interactive Activity:** The lab is completed using the following steps:
 - (a) **Students explore a system without ethical safeguards:** Learners first interact with the software or scenario as-is, without ethical guardrails applied, completing a simple task and observing how it behaves under typical conditions.
 - (b) **Students experience the ethical issue through a simulated consequence:** The lab includes a mechanism to simulate or highlight the ethical problem (*e.g.*, bias, privacy violation, unfair decision), allowing students to directly observe the negative impact of unethical design.
 - (c) **Students receive guidance on ethical best practices:** The lab then presents strategies, frameworks, or guidelines that can be used to mitigate or prevent the ethical concern encountered.
 - (d) **Students apply ethical improvements:** Students modify the software or scenario, implementing the recommended ethical practices and gaining hands-on experience in improving system behavior.
 - (e) **Students evaluate their ethical improvements:** With the modifications applied, students re-engage with the scenario to observe the impact of their changes and reflect on how ethical design alters system outcomes. This process helps build confidence in applying ethical principles in practice.
3. **Empathy-building reflection:** Research has shown that empathy development improves ethical awareness and motivation to act responsibly. In our labs, this is supported through short testimonials or perspectives from individuals directly affected by the ethical issue being explored.

4. **Assessment:** An optional online quiz that enables instructors to evaluate student understanding and verify completion of the activity. Learners may submit a screenshot of their results as evidence of completion.

3.3 Lab Topics

The three ethics-focused labs are described below and address the topics of:

- **Ethics of AI:** This lab familiarizes students with machine learning bias and the ethical principles for responsible ML design by demonstrating how biased data influences algorithmic outcomes and giving learners hands-on experience in reducing those biases (Section 4.1).
- **Algorithmic Bias:** Teaches about machine learning bias and the ethical principles that should guide ML design by demonstrating how biased data affects algorithmic behavior and providing hands-on experience in reducing those biases (Section 4.2).
- **Bias in Neural Networks:** Teaches about bias in machine learning and the ethical principles for designing fair ML systems by illustrating how biased data influences algorithmic results and giving participants hands-on experience detecting and reducing those biases (Section 4.3).

4 Developed Intervention Activity (Labs)

4.1 Overview: Ethics of AI Lab

Lab Learning Objectives:

Learning Objectives (LO)

LO1: Recognize different ethical challenges in AI (Comprehension)

LO2: Practice consideration of ethics in a AI-related scenario (Application)

LO3: Diagnose ethical implications of AI choices (Synthesis)

This lab provides an accessible, experiential way to engage with the ethical implications of artificial intelligence. Its primary objective is to help participants recognize how bias can emerge in AI systems, reflect on the consequences of biased decision-making, and practice strategies for mitigating these issues. The learning goals include identifying ethical challenges, diagnosing the implications of AI choices, and evaluating appropriate responses. To achieve these goals, the lab combines guided readings, interactive activities, media resources, and assessment.

The lab begins by introducing users to the foundational concepts of AI, its applications, and common misconceptions. Through an initial reading—supported

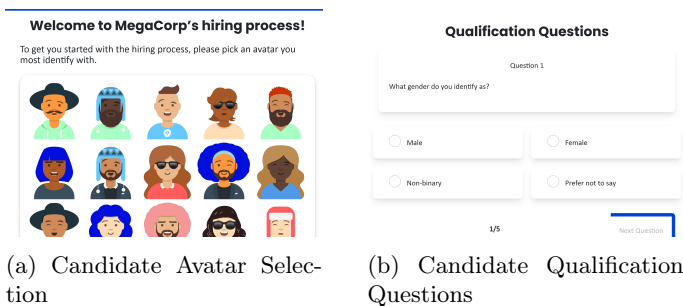


Figure 2: Hiring: Candidate Avatar Selection and Qualification Questions

by visual aids, real-world examples, and guided reflection questions—participants gain context for understanding how AI systems are integrated into daily life and why ethical considerations are essential. This foundation prepares participants for the interactive modules, where they experience AI bias firsthand and reflect on its effects.

In the main experiential activity, participants engage in a simulated hiring process. They first take the role of a job applicant (Figure 2) whose application is unfairly rejected by an AI agent, allowing them to personally experience algorithmic discrimination (Figure 3). Next, they adopt the perspective of an employer, selecting candidates while being influenced by a flawed AI recommendation system (Figure 4). By later revealing that the system discriminated on arbitrary traits such as hats or glasses, the lab underscores both the irrationality and harm of biased algorithms. Participants are then given the opportunity to “repair” the system by adjusting criteria to reflect more equitable attributes, reinforcing the role of deliberate design choices in reducing bias.

After completing the simulation, participants move into a reinforcement phase that includes supplemental media and a short quiz. This consolidation step deepens their knowledge, fosters empathy, and provides a sense of closure with a certificate of completion. Ultimately, the lab demonstrates that AI is not inherently neutral; it reflects the data and values encoded into it. By situating users in interactive, bias-laden scenarios and guiding them through corrective actions, the lab builds critical awareness and equips learners with practical insights for addressing ethical challenges in AI development and application.

4.2 Overview: Algorithmic Bias Lab

Lab Learning Objectives:

Learning Objectives (LO)

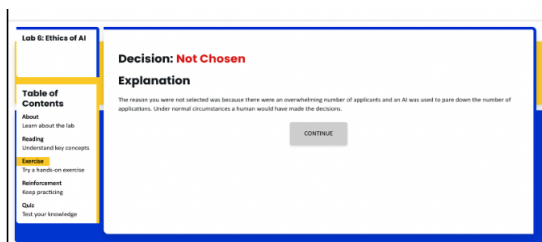


Figure 3: Lab 6: Candidate Rejected Due to Simulated AI Bias

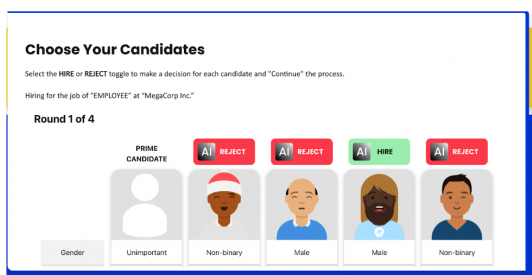


Figure 4: Lab 6: Participant Acts as Hiring Manager, Reviewing AI-Generated Recommendations

- LO1: Identify how human prejudice infiltrates datasets resulting in biased machine learning systems (Knowledge)
- LO2: Classify different types of inequitable bias in machine learning systems (Comprehension)
- LO3: Apply standard guiding principles when developing machine learning systems (Application)
- LO4: Analyze examples of real-world machine learning systems to diagnose the impact of harmful bias (Synthesis)

This lab is designed to introduce learners to machine learning (ML) bias and the ethical principles that should guide the development of ML systems. Its primary objective is to help participants recognize how biased data influences algorithmic outputs and to provide opportunities to mitigate such biases.

The lab begins with an overview of core concepts, followed by a short reading on algorithms and their role in training ML models. Participants are then placed in a simulated scenario as a live-stream chat moderator. In this task, an AI system provides recommendations on whether specific chat messages should be retained or removed (Figure 5). While participants make the fi-

nal decision, the exercise is designed to expose discrepancies between the AI’s recommendations and their own judgments.

After participants complete their selections, the system reveals that the AI was trained on an overly generalized dataset, resulting in selection bias. This disclosure is accompanied by an explanation of sentiment analysis, emphasizing how training data must be manually labeled (*e.g.*, as negative, neutral, or positive) and how such processes are prone to human-introduced biases.

In the next stage, participants are asked to “repair” the dataset by correcting sentiment labels so that they better reflect the intended meaning of the text. This interactive correction demonstrates how biased training data directly distorts AI outputs and how careful data curation can reduce such distortions. Following the repair, participants observe that the AI’s recommendations now align more closely with appropriate moderation decisions, reinforcing the impact of dataset quality on model behavior (Figure 6).

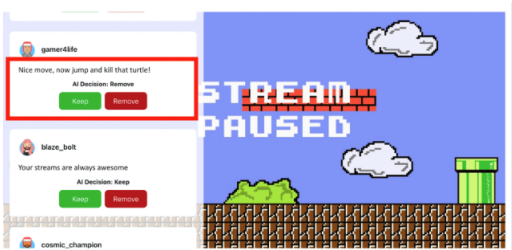


Figure 5: Example of AI Before It is Repaired by the Participant

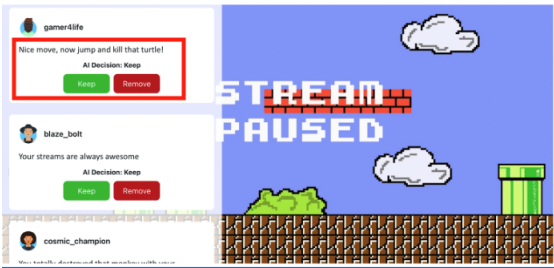


Figure 6: Example of AI After It is Repaired by the Participant

4.3 Overview: Neural Networks and Machine Learning

Lab Learning Objectives:

Learning Objectives (LO)

- LO1: Construct a basic neural network using provided components (Synthesis)
- LO2: Simulate neural network training (Comprehension)
- LO3: Demonstrate bias present in a neural network (Application)



Figure 7: Participant playing the game and avoiding blocks to generate data

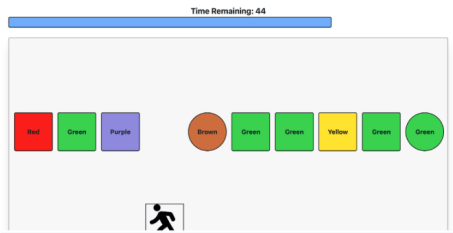


Figure 8: Participant observes the AI play the game

The lab is designed to introduce students to neural networks while illustrating how biases can emerge during their development and training. The exercise begins with a coding task in which the user must “fix” code to enable a character to move on the screen, simulating the process of building a training model from the ground up. Participants then generate training data by manually controlling the character to avoid falling blocks. This phase, which consists of an initial 30-second trial followed by one or more training sessions of 1–1.5 minutes, mimics supervised learning and demonstrates the iterative nature of data collection (Figure 7).

Participants then observe the AI-controlled character performing the same task (Figure 8). If no clear path is available, the AI avoids the color of the blocks that the participant most frequently collided with during training. This outcome is used to highlight how unintentional biases in user-generated data can propagate into the model. The system then provides individualized feedback,

showing the participant’s collision frequencies by block color and explaining how bias was introduced.

Participants are next encouraged to mitigate the imbalance by intentionally colliding with different-colored blocks at approximately equal rates in subsequent training sessions. Although complete elimination of bias is not possible, the exercise illustrates how deliberate adjustments and repeated training can reduce disparities in model behavior.

5 Classroom Inclusion Details

Over several recent semesters, we have included our labs in a variety of undergraduate course offerings at six categorically distinct universities. The ethics-focused labs were utilized in several modalities including during conventional classroom sessions, after school events and undergraduate outreach sessions. Adopting institutions ranged from a small liberal arts HBCU, to large nationally-known R1s. Courses ranged from foundational CS1/CS2 courses, to those focusing upon more specialized computing topics.

6 Inclusion Observations

6.1 Observations From Our Classroom Inclusion

Over several recent semesters, we have included our labs in undergraduate course offerings, with hundreds of students participating at six universities. Participating instructors noted that the labs were: I) Easy to include in their curriculum from a technical and materials perspective, II) Appeared to capture student attention and increase their interest, and III) Made their job easier because they didn’t need to create new material.

6.1.1 Faculty Quotes/Observations

The following are several quotes from participating instructors:

“Students found the interactive scenarios engaging and appreciated the opportunity to ‘fix’ problematic AI behaviors, which helped solidify their understanding of ethical AI development.”

“Students appreciated the hands-on approach to learning about neural networks, with many expressing that building and training their own networks helped demystify machine learning concepts.”

“Building on their work with AI ethics and neural networks, students effectively applied their understanding to autonomous cybersecurity systems, demonstrating how technical knowledge and ethical considerations must work together.”

“Integrating these labs into my weekly teaching routine has significantly strengthened students’ confidence. I could clearly see how the hands-on structure helped them bridge the gap between theory and real-world application.”

“Throughout the semester, I observed that students who engaged deeply with the labs showed noticeably stronger problem-solving skills and were more comfortable experimenting with new tools and technologies.”

“The lab activities created consistent opportunities for active learning. Even students who were initially hesitant became more curious and independent once they realized they could explore, test, and troubleshoot in a guided environment.”

“These labs supported students’ growth not only in technical proficiency but also in teamwork and communication. Their ability to explain their process improved week after week.”

6.1.2 Student Quotes/Observations

The following are student quotes collected after obtaining IRB approval.

“I enjoyed the labs this week, especially the neural network lab. As the AI industry continues to grow, I find it more fascinating. I particularly found its role in medical diagnosis interesting.”

“This week topic was Ethics of AI. This is an incredibly important topic in this era. How we approach these ethics will affect all industries, not just tech. One real example in particular is using AI in hiring, where it’s important to make sure the system doesn’t unfairly favor or reject people. The challenge is that past data can carry hidden bias that gets copied into the AI. The way to handle this is by checking the data and results often and making changes when problems show up.”

“The instructions and labs were easy to follow.”

“I felt more confident by the end of the lab.”

“The content was good and easy to use.”

6.2 Discussion, General Reflections and Recommendations

We identified several important lessons regarding effective strategies for integrating ethics-focused AI/ML content into STEM and computing classrooms. We hope that these insights are valuable not only for instructors adopting our materials but also for anyone incorporating broader AI/ML ethics modules into their curriculum. Instructors should emphasize the critical importance of developing responsible and trustworthy AI systems. As we have shown in our prior work [13, 14, 9], this can be accomplished through building empathy for real-world stakeholders affected by AI decisions or through more traditional instructional approaches. For example, the instructor opened the session with a brief narrative about the societal consequences of a biased ML system to underscore the relevance of AI ethics. Additional points of discussion included the idea that ethical responsibility is shared across all roles in the AI pipeline—not just those specializing in ethics.

Conversations regarding how industry increasingly values AI professionals who understand fairness, transparency, and responsible data practices were particularly impactful, especially for early-career students. It was also helpful to frame AI ethics alongside other foundational topics—such as safety, cybersecurity, quality assurance, or risk management—to show that ethical considerations are integral to the entire lifecycle of AI/ML development. Ultimately, designing ethical AI is part of designing inclusive AI: systems that avoid harm, serve diverse populations, and uphold societal and legal expectations.

One of our key goals is to dispel the misconception that ethical AI is “someone else’s job.” Students should see that every contributor—data collectors, modelers, evaluators, system designers, and deployment engineers—plays a critical role in ensuring that AI systems behave responsibly. This includes choices such as selecting representative datasets, building audits for bias, documenting model decisions, and questioning the assumptions embedded in algorithms. A central challenge in creating our materials was illustrating this interconnected ecosystem of responsibility.

An unexpected benefit of the labs was the depth of discussion they fostered around what it means to be impacted by AI systems. Many students, particularly those in lower-division courses, lacked a baseline understanding of how unexamined AI models can reinforce bias or exclude entire groups of users. We found that an essential first step toward teaching ethical AI development is helping students understand the lived experiences of the communities affected by automated decisions.

We also believe that barriers to adopting ethics-focused AI/ML educational resources must be minimized. We found that discussing the relevance of ethical considerations at every stage of the AI pipeline—from problem formulation and data collection to model evaluation and deployment—helped students recognize

that ethical responsibility is central, rather than peripheral, to AI/ML practice.

7 Conclusion

This work provides instructor observations regarding the inclusion of several ethics-focused Accessible Learning Labs at six universities. Our primary findings include: I) Instructors valued the modular, à-la-carte structure of the labs; II) The 30–60 minute completion time made adoption easier; III) Both instructors and students appreciated that the labs were fully hosted on the web and self-contained; and IV) Ethics-oriented, empathy-building content appeared to motivate students.

Acknowledgements

This material is based upon work supported by the United States National Science Foundation under grants #2111152, #2145010, #2336941, #2524374.

References

- [1] *Accessible Learning Labs*. <https://all.rit.edu/>.
- [2] Safinah Ali et al. “Constructionism, ethics, and creativity: Developing primary and middle school artificial intelligence education”. In: *International workshop on education in artificial intelligence k-12 (eduai’19)*. 2019, pp. 1–4.
- [3] Fatma Basak Aydemir and Fabiano Dalpiaz. “A roadmap for ethics-aware software engineering”. In: *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. IEEE. 2018, pp. 15–21.
- [4] Eric Eaton et al. “Blue sky ideas in artificial intelligence education from the EAAI 2017 new and future AI educator program”. In: *AI Matters* 3.4 (2018), pp. 23–31.
- [5] Natalie Garrett, Nathan Beard, and Casey Fiesler. “More Than “If Time Allows”: The Role of Ethics in AI Education”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 272–278. ISBN: 9781450371100. DOI: 10.1145/3375627.3375868. URL: <https://doi.org/10.1145/3375627.3375868>.
- [6] Wayne Holmes et al. “Ethics of AI in education: Towards a community-wide framework”. In: *International Journal of Artificial Intelligence in Education* (2021), pp. 1–23.

- [7] Paul Karoff. *Embedding ethics in computer science curriculum*. Jan. 2019. URL: <https://news.harvard.edu/gazette/story/2019/01/harvard-works-to-embed-ethics-in-computer-science-curriculum/>.
- [8] Gali Katznelson and Sara Gerke. “The need for health AI ethics in medical school education”. In: *Advances in Health Sciences Education* 26.4 (2021), pp. 1447–1458.
- [9] Saad Khan et al. “Experiential Learning in Undergraduate Accessibility Education: Instructor Observations”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 54–68.
- [10] Stephan Krusche et al. “Interactive Learning: Increasing Student Participation Through Shorter Exercise Cycles”. In: *Proceedings of the Nineteenth Australasian Computing Education Conference*. ACE ’17. Geelong, VIC, Australia: ACM, 2017, pp. 17–26. ISBN: 978-1-4503-4823-2. DOI: 10.1145/3013499.3013513. URL: <http://doi.acm.org.ezproxy.rit.edu/10.1145/3013499.3013513>.
- [11] K. Pretz. *Why Schools Are Getting More Serious About Teaching Engineering Students About Ethics*. <http://theinstitute.ieee.org/ieee-roundup/blogs/blog/why-schools-are-getting-more-serious-about-teaching-engineering-students-about-ethics>. 2018.
- [12] Daniel Schiff. “Education for AI, not AI for Education: The Role of Education and Ethics in National AI Policy Strategies”. In: *International Journal of Artificial Intelligence in Education* (2021), pp. 1–37.
- [13] Weishi Shi et al. “ALL: Supporting Experiential Accessibility Education and Inclusive Software Development”. In: *ACM Trans. Softw. Eng. Methodol.* (Sept. 2023). ISSN: 1049-331X. DOI: 10.1145/3625292. URL: <https://doi.org/10.1145/3625292>.
- [14] Weishi Shi et al. “Presenting and Evaluating the Impact of Experiential Learning in Computing Accessibility Education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. ICSE-SEET ’20. Distinguished Paper Award. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 49–60. ISBN: 9781450371247. DOI: 10.1145/3377814.3381710. URL: <https://doi.org/10.1145/3377814.3381710>.
- [15] R. Williams. “How to Train Your Robot: Project-Based AI and Ethics Education for Middle School Classrooms”. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021.
- [16] Stephanie Wykstra. *Fixing Tech’s Ethics Problem Starts in the Classroom*. Feb. 2019. URL: <https://www.thenation.com/article/teaching-technology-ethics-big-data-algorithms-artificial-intelligence/>.

Supporting High School Computing Education with Easily Adoptable Experiential Learning Modules*

Ursula Parker¹, Farzana Rahman², Cathleen Cerosaletti³,
Zenon Borys³, Samuel Malachowsky¹, Daniel Krutz¹

¹Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623

{ucp4496,samvse,dxkvse}@rit.edu

²College of Engineering and Computer Science
Syracuse University
Syracuse, NY 13244

frahma02@syr.edu

³Warner School of Education and Human Development
University of Rochester
Rochester, NY 14627

{ccerosaletti,zborys}@warner.rochester.edu

Abstract

Foundational STEM areas—including artificial intelligence, cybersecurity, and inclusive software development—are often missing from the computing curricula of grades 9-12. A major reason for this lack of inclusion is the shortage of high-quality, easy-to-use teaching materials. This gap is especially challenging for smaller or under-resourced schools that disproportionately serve underrepresented students. As a result,

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

many high school classrooms are unable to cover these essential topics, contributing to a broader “diversity crisis” in the U.S. STEM pipeline.

The Accessible Learning Labs are a collection of 15 experiential, computing-focused educational modules originally developed and validated for undergraduate learners. Designed to promote hands-on engagement with computing-focused topics such as machine learning, artificial intelligence, inclusive software development and ethics-focused topics, these modules have demonstrated strong effectiveness in building foundational technical skills and increasing student confidence. This paper presents the adaptation of the ALL modules for high school students in grades 9–12, highlighting modifications to content, scaffolding, and delivery to ensure developmental appropriateness and curricular alignment with secondary education standards. We describe the pedagogical framework guiding the redesign, pilot implementations strategies, and provide recommendations for integrating the modules into existing computer science pathways. By offering flexible, accessible, and engaging learning experiences, the adapted ALL modules aim to expand early exposure to inclusive computing practices and support a more diverse pipeline of future technologists. Complete project materials are openly available on our website: <https://all.rit.edu>

1 Introduction

Foundational STEM topics—including artificial intelligence, cybersecurity, and inclusive software development—remain largely absent from 9–12 computing curricula [9, 28]. A persistent barrier to their inclusion is the limited availability of high-quality, developmentally appropriate, and easily adoptable instructional resources [8, 10]. This gap is particularly prevalent in smaller or under-resourced schools, many of which serve populations historically under-represented in computing [3, 13, 34]. The inability to offer meaningful learning opportunities in these foundational areas contributes to the ongoing “diversity crisis” in U.S. STEM pathways [3].

To address this limitation of easy to adopt, high-quality experiential educational computing focused topics for the 9-12 audience, we have have modified and extended our Accessible Learning Labs [1] for usage within grades 9–12 environment. The ALL modules are inclusive, easily adoptable, and grounded in experiential learning principles, supporting instruction in critical STEM domains such as artificial intelligence, machine learning, and accessible software design while simultaneously advancing pedagogical knowledge. Specific alterations to these existing modules include altering the instructional material for the suitability of 9-12 students and tailoring the included instructor resources for 9-12 instructors (Section 2.1.2CR1).

Previous work has demonstrated the positive benefits of the ALL mod-

ules for the undergraduate environment [15, 30]. Research has shown that the ALL modules are accessible and easy to implement by adopters [16], enabling schools—including those with limited resources and those serving underrepresented students—to integrate essential STEM topics into their curricula [14, 36]. The objectives of this work is to: I) Describe the labs for the 9-12 audience (Section 2), and II) Present adoption scenarios that will support instructors for including these increasingly foundational computing topics (Section 3).

2 Overview: Accessible Learning Labs

The Accessible Learning Labs [1] have served thousands of students at numerous institutions throughout the United States. The labs are considered *accessible* for several reasons: (I) Their fully self-contained, web-based format enables effortless and cost-free adoption; (II) They employ inclusive technologies to ensure usability for all learners, including those who are blind, Deaf/HH, or otherwise disabled; (III) They present complex, contemporary computing topics in simplified forms that make them approachable to a broad range of instructors and learners; and (IV) A suite of instructor-facing resources supports implementation and facilitates participant monitoring within classroom settings.

Each lab addresses a specific STEM computing topic; Table 1 provides an overview of each lab’s components and sequence. Existing available on the project website include: [1]: ① Sound and Speech; ② Color Blindness; ③ Screen Readers (blindness); ④ Dexterity; ⑤ Cognitive Impairments; ⑥ Ethics of AI; ⑦ Foundations of AI Cybersecurity; ⑧ Algorithmic Biases; ⑨ MAPE-K; ⑩ Neural Networks; ⑪ Multi-Armed Bandits; ⑫ Reinforcement Learning; ⑬ Introduction to Autonomy; ⑭ Supervised Learning; ⑮ Unsupervised Learning.

2.1 Experiential Learning Format/Steps For Lab Completion

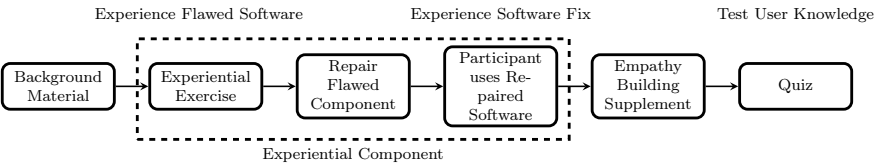


Figure 1: Experiential, empathy-building intervention method utilized.

Table 1: Components included in each of the labs.

Component	Description
Defined learning objectives	Learning objectives are defined before lab creation and inform each of the components below
STEM computing topic info	Background information on the topic being addressed. Includes compliance information and information about how users are affected
Lab instructions	Information how to complete the labs presented directly to the user
Lecture slides	Lecture slides that an instructor may use to present the material
Video presentation of lecture	ADA-compliant YouTube video of the lecture, supporting learners conducting the labs on their own or in a flipped classroom environment
Empathy-building supplement	ADA-compliant YouTube video from people impacted by the topic
Video of lab being conducted	ADA-compliant YouTube video of the lab being conducted
Lab activity	The lab activity to be conducted (Section 2.1)
Emulation component	Emulating the experience of non-inclusive software
Ethics and inclusivity	Ethical and legal importance of developing inclusive software
Instructor Resources	Customized student tracking, reporting, adoption information.
Quiz	An example quiz provided for participant evaluation at the conclusion

The labs are grounded in well-established *experiential learning principles* [18, 20, 24], following Kolb’s four-stage cycle of abstract conceptualization, active experimentation, concrete experience, and reflective observation [17]. This framework ensures that learners develop both conceptual understanding and practical, hands-on skills. By emphasizing personal engagement and the translation of experience into reflection, the labs create a rich and effective learning environment [29]. Each module incorporates interactive components that invite participants to manipulate core concepts directly and observe the consequences of their choices within a simulated system.

Figure 1 outlines the workflow for completing an individual lab. Each module contains several components designed to introduce, contextualize, and motivate engagement with the featured STEM topic:

1. **Background instructional material:** Establishes foundational knowledge relevant to the topic.

2. **Experiential activity:** Guides participants through a web-based, interactive sequence consisting of:
 - (a) **Initial software interaction:** Participants begin by completing a simple task within the system.
 - (b) **Exposure to a flawed system:** Learners encounter a deliberately “broken” or suboptimal element and experience the resulting negative effects.
 - (c) **Instruction on appropriate remediation:** Participants are given an explanation of correct practices and procedures that address the observed issue.
 - (d) **Implementation of the fix:** Using the provided guidance, learners modify the application to resolve the identified problem (Figure 2).
 - (e) **Evaluation of the repaired system:** Participants interact with the corrected system, observe the impact of their changes, and reflect on the improvements—an essential step in building confidence and deepening experiential learning.
3. **Empathy-building supplemental material:** Each ALL module contains one or more components focused on demonstrating the impact of inequitable/non-inclusive software and an informational component teaching participants how to create more inclusive software.
4. **Auto-graded quiz:** An optional web-based assessment allows learners to check their understanding. When signed in through Google, participants receive a personalized certificate that instructors may use for documentation or credit.

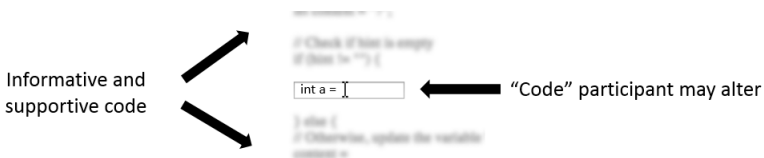


Figure 2: ALL modules enables participants to edit only intended portions of source code through a custom interface. The modified portions drive the simulations, enabling the participant to experientially learn.

2.1.1 Intentional Inclusivity

Despite regulations [2, 12, 33, 35], the development of inequitable software is still prevalent [5, 34]. Our foundational work has found that experiential, empathy-creating content can be beneficial for demonstrating the importance

of creating inclusive software [16, 19, 22, 31, 32]; this is crucial due to the magnified impact that their work has on the general population. Each ALL module contains one or more components focused on demonstrating the impact of inequitable/non-inclusive software and an informational component teaching participants how to create more inclusive software. *The objective of this inclusivity feature is for participants to experience the impact of creating inclusive software, not merely being told about the importance.* Consequently, each lab will contain supplementary awareness-creating materials:

1. **Experiential impact of inequitable issue:** Participants experience the adverse impact of non-inclusive software firsthand. Examples of such include AI/ML bias or inaccessible software (through an emulation feature such as those used in existing labs *i.e.*, a game in gray-tones simulating colorblindness).
2. **Real-world impact of inequitable issue:** The description of the provided issue will include its impact (both socially and financially), why it occurred, and how it could have been prevented.
3. **Descriptive impact from actual users who have been adversely impacted by inequitable software:** For each lab, at least one person who has been impacted by inequitable software relating to the examined STEM topic will contribute insight into their experiences with the discussed challenge, providing additional real-world context.
4. **Ethics/professional responsibility:** To address recent US initiatives [12], labs contain a component articulating the ethical and legal importance of developing inclusive software. For example, many accessibility-focused labs focus on Section 508 [33], its technical standards, and available validators.

2.1.2 Instructor Support Materials

Each lab provides a set of instructor resources to help educators deliver the content effectively and monitor student progress. Using learning groups, instructors can efficiently oversee participants' engagement with assigned modules. As students work through the labs, educators can view real-time progress, including completion of individual sections and quiz results. These resources are designed to complement lecture materials and supporting videos, providing a comprehensive toolkit for instruction and assessment.

Learning groups (Figure 3b) enable instructors to track and manage their students' progress on assigned labs. To create a group, simply choose a group name and select the labs you want your students to complete. Figure 3a illustrates how to create a group and assign labs. Once the group is set up, share the unique group code with your students. Students are then able to join your instructing group, as demonstrated in Figure 3. Finally, as students work

4. **Quick setup and integration:** Since no software installation or configuration is required, the labs can be seamlessly integrated even into short, introductory courses.
5. **Flexible completion time:** Each lab is designed to be completed in 30 minutes to 3 hours, allowing use à-la-carte, in-class or as homework.
6. **Supports diverse learning styles:** Students can benefit from the labs whether they are enrolled in a formal course or learning independently, accommodating a variety of learning preferences.
7. **Inclusive and accessible:** The labs are designed for all learners, including those with accessibility needs (*e.g.*, Deaf/HH, colorblindness, etc.).

These features collectively ensure that the ALL modules can be widely adopted across diverse educational contexts, including schools with limited resources and those serving underrepresented student populations, while maintaining a focus on accessibility and inclusivity.

3 Utilization Instructions

The Accessible Learning Labs are intentionally designed to support a wide range of learning contexts, enabling broad and equitable access to foundational computing experiences. Their modular structure, experiential learning approach, and minimal resource requirements make them adaptable to diverse implementation models. The following sections describe how the ALL modules can be effectively integrated into various 9-12 learning environments. These include individual learning pathways, traditional classroom instruction, and after-school enrichment programs.

3.1 Individual Learners

The Accessible Learning Labs are well suited for individual learners, providing a flexible, self-guided pathway for students to explore foundational topics in computing, artificial intelligence, and accessible software design. Each module is designed to be self-contained, enabling learners to engage with both conceptual materials and hands-on activities at their own pace. Because the labs incorporate scaffolded explanations, real-world examples, and incremental challenges, students can independently acquire new skills while building confidence in their ability to navigate technical content. This structure is particularly beneficial for learners who may not have access to advanced computing coursework in traditional school settings.

In the 9–12 context, the ALL modules can support differentiated learning by allowing students to select modules aligned with their interests or readiness levels. For example, a student interested in artificial intelligence, can complete

the introductory AI lab before progressing to more advanced modules, while a student drawn to design or accessibility can focus on labs related to inclusive software development. The modular format enables individualized learning pathways without requiring instructors to design or manage separate curricula for each student. This flexibility also supports learner in revisiting material, exploring alternative strategies, or repeating activities to deepen comprehension—an approach consistent with mastery-based learning.

The ALL modules additionally provide opportunities for enrichment and extension beyond the traditional classroom. Motivated learners can work through the modules as part of independent study, after-school programs, or participation in computing clubs. Because the labs rely on widely accessible technologies and step-by-step guidance, students can complete them at home or in community spaces without needing specialized equipment. This makes the ALL modules particularly valuable for learners in rural or under-resourced districts, where access to high-level computing coursework and teacher expertise may be limited. The experiential nature of the labs ensures that even students working independently can engage in meaningful, hands-on exploration rather than solely reading or watching instructional content.

3.2 Conventional Classroom Activities

The Accessible Learning Labs are designed to integrate seamlessly into traditional classroom environments, allowing teachers to introduce foundational computing concepts without requiring extensive curriculum development or specialized technical expertise. Each lab is structured around defined learning objectives, step-by-step guidance, and built-in scaffolding, making it straightforward for instructors to implement the activities as part of a unit, a multiweek module, or a short-term intervention. Because the ALL modules focus on experiential, hands-on learning, they support existing computer science or STEM courses by offering opportunities for students to apply key ideas in artificial intelligence, machine learning, and accessible software development through authentic problem-solving tasks.

In addition to their content flexibility, the ALL modules support diverse instructional models commonly used in 9–12 settings. Teachers can employ the labs during whole-class instruction, using them to anchor discussions, demonstrations, or collaborative explorations. Alternatively, the modules can be deployed in small-group rotations or station-based learning, enabling students to work at varying paces while the teacher provides targeted support. This adaptability makes the ALL modules a strong fit for classrooms with heterogeneous skill levels, where some students may be encountering computing concepts for the first time while others are ready for more advanced challenges.

The ALL modules also reduce logistical and resource barriers that fre-

quently limit the inclusion of advanced STEM topics in 9–12 classrooms. The materials are designed to run on readily available devices and rely on accessible tools, lowering the need for specialized hardware or software. The modular structure allows teachers to choose individual labs that align with their existing curriculum, local standards, or time constraints, rather than committing to a comprehensive redesign of the course. Furthermore, because the labs emphasize accessibility and inclusive design, teachers can incorporate them into broader conversations about equity, universal design, or responsible technology use, making them relevant across multiple subject areas.

Finally, the ALL modules support sustained classroom engagement by providing meaningful, project-based outcomes that connect directly to students' lived experiences. Many of the lab modules culminate in demonstrations, prototypes, or small-scale analyses that can be shared in class presentations or used as checkpoints for formative assessment. These artifacts allow teachers to monitor learning progression while giving students concrete evidence of their developing skills. By embedding experiential computing activities within a flexible, teacher-friendly framework, the ALL modules empower traditional classrooms to introduce emerging STEM topics in a manner that is accessible, engaging, and pedagogically sound.

3.3 After School Events

After-school programs provide an ideal environment for implementing the Accessible Learning Labs, offering students opportunities to explore advanced computing topics beyond the constraints of the traditional school schedule. Because after-school settings typically allow for more flexible pacing and informal learning structures, students can engage deeply with the experiential, project-based activities that characterize the ALL modules. This flexibility enables learners to follow their curiosity, revisit challenging concepts, and iterate on their work, all of which are central to the experiential learning model on which the labs are built.

The ALL modules also align well with the enrichment-oriented goals of many after-school programs, where clubs and activities often seek to spark interest in STEM fields and nurture early talent. Whether integrated into computer science clubs, STEM outreach efforts, robotics teams, or community technology centers, the labs provide ready-to-use, high-quality instructional materials that do not require specialized equipment or extensive teacher preparation. Adult mentors—including teachers, community volunteers, or college outreach coordinators—can facilitate the activities with minimal training, making the labs accessible to a broad range of programs, including those operating with limited funding or staffing.

In addition, after-school contexts support collaborative learning in ways

that complement the ALL modules' design. Small group formats, peer mentorship, and student-led exploration are common features of extracurricular STEM programs, and the labs are structured to encourage these forms of interaction. Students can work together to build models, test solutions, or refine accessible design ideas, creating a supportive environment where learners share knowledge and challenge one another.

3.4 University Supported 9-12 Outreach Events

College-supported 9–12 outreach programs, including initiatives like the NY STEP [26] program, New Jersey's NJSLS-CS [25] and Rhode Island's CS4RI [7], are well-suited to broaden participation in STEM by creating sustained, high-impact learning opportunities for students who may not otherwise have access to advanced computing content. The Accessible Learning Labs modules align directly with these goals by providing ready-to-use, experiential modules that can be delivered during campus visits, weekend workshops, or multi-week outreach sessions. Because the labs introduce foundational concepts in artificial intelligence, machine learning, and accessible technology through experiential, inclusive activities, they offer an interesting and useful entry point for students who are interesting in computing but may not yet see themselves represented in the field.

The ALL modules also enhance the effectiveness of college-supported outreach programs by offering structured, scalable content that can be facilitated by undergraduate mentors, graduate students, or outreach coordinators. Programs like NY STEP often integrate near-peer mentoring to support students academically and socially, and the ALL modules naturally complement this model: mentors can guide participants through the labs, share lived experiences about studying computing in college, and help demystify pathways into STEM majors. This creates an environment where students not only learn technical concepts but also build confidence, develop a sense of belonging, and gain exposure to authentic college-level learning experiences.

Additionally, the modular design of the ALL modules enables outreach programs to tailor learning experiences to different age groups, prior knowledge levels, or program goals. A shorter event might center on a single introductory module, while a multi-semester program could integrate several labs as part of a coherent curriculum. Because the materials require only commonly available technology and include detailed facilitator guidance, they are well suited to outreach contexts where logistical constraints or variable instructional experience might otherwise limit the scope of STEM offerings. This flexibility allows programs serving diverse populations—including students from historically marginalized groups, first-generation college aspirants, and learners from under-resourced districts—to provide high-quality computing education expe-

riences.

The ALL modules additionally support long-term engagement and pipeline development, which are central aims of many college-based outreach initiatives. Students can complete labs that result in tangible artifacts—such as accessible design evaluations or simple AI models—that can be shared with families, teachers, or future mentors. These products reinforce participants’ sense of achievement and provide meaningful checkpoints that outreach programs can use to track growth over time. By embedding experiential STEM learning within sustained mentorship and college-connected programming, the ALL modules help strengthen the bridge between 9–12 learners and higher education, ultimately contributing to a more diverse and well-prepared pool of future college applicants in STEM fields.

4 Conclusion

This work addresses a critical gap in 9–12 computing education by adapting a proven set of undergraduate-level modules to support foundational STEM learning in artificial intelligence, cybersecurity, inclusive software development, and related areas. Complete project materials are openly available on our website: <https://all.rit.edu>

Acknowledgements

This material is based upon work supported by the United States National Science Foundation under grants #2111152, #2145010, #2336941, #2524374.

References

- [1] *Accessible Learning Labs*. <https://all.rit.edu/>.
- [2] *Assistive Technology Act Law*. URL: %5Curl%7B%20http://www.ataporg.org/ATLaw%7D.
- [3] Johana Bhuiyan. *Diversity Crisis in AI, 2017 edition*. Aug. 2016.
- [4] Paul R Bohman. *Teaching accessibility and design-for-all in the information and communication technology curriculum: Three case studies of universities in the United States, England, and Austria*. Utah State University, 2012.
- [5] Ryan Browne and MacKenzie Sigalos. *A.I. has a discrimination problem. In banking, the consequences can be severe*. June 2023.
- [6] Sheryl E Burgstahler. *Universal design in higher education: From principles to practice*. ERIC, 2015.

- [7] *Computer Science for Rhode Island*. URL: <https://www.cs4ri.org/about-overview>.
- [8] Intel Corporation. *Teaching AI: Educator Resources for Artificial Intelligence in the Classroom*. 2022. URL: <https://www.intel.com/content/www/us/en/education/k12/teachers/teaching-ai.html>.
- [9] Ali Crawford and Cherry Wu. “Riding the AI Wave: What’s Happening in K-12 Education?” In: *Center for Security and Emerging Technology* (Apr. 2024). URL: <https://cset.georgetown.edu/article/riding-the-ai-wave-whats-happening-in-k-12-education/>.
- [10] Cybersecurity and Infrastructure Security Agency. *Cybersecurity for K-12 Education*. 2023. URL: <https://www.cisa.gov/K12Cybersecurity>.
- [11] Wenliang Du, Honghao Zeng, and Kyungrok Won. “SEED emulator: an internet emulator for research and education”. In: *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 2022, pp. 101–107.
- [12] *FACT SHEET: Biden-Harris Administration Takes New Steps to Advance Responsible Artificial Intelligence Research, Development, and Deployment*. May 2023.
- [13] Benjamin Herold. *Poor Students Face Digital Divide in How Teachers Learn to Use Tech*. <https://www.edweek.org/ew/articles/2017/06/14/poor-students-face-digital-divide-in-teacher-technology-training.html>. 2017.
- [14] *How K-12 Schools Can Minimize the Effects of Resource Gaps in Underserved Schools*. 2023.
- [15] Saad Khan et al. “Experiential Learning in Undergraduate Accessibility Education: Instructor Observations”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 54–68.
- [16] Saad Khan et al. “Experiential Learning in Undergraduate Accessibility Education: Instructor Observations”. In: *J. Comput. Sci. Coll.* 38.8 (June 2023), pp. 54–68. ISSN: 1937-4771.
- [17] David A Kolb. *Experiential learning: Experience as the source of learning and development*. FT press, 2014.
- [18] Stephan Krusche et al. “Interactive Learning: Increasing Student Participation Through Shorter Exercise Cycles”. In: *Proceedings of the Nineteenth Australasian Computing Education Conference. ACE '17*. Geelong, VIC, Australia: ACM, 2017, pp. 17–26. ISBN: 978-1-4503-4823-2. DOI: 10.1145/3013499.3013513. URL: <http://doi.acm.org.ezproxy.rit.edu/10.1145/3013499.3013513>.

- [19] Kyle L-Messerle, Samuel Malachowsky, and Daniel E. Krutz. “Open Questions for Empathy-Building Interventions for Inclusive Software Development”. In: *J. Comput. Sci. Coll.* 38.8 (June 2023), pp. 201–213. ISSN: 1937-4771.
- [20] Linda Laird and Ye Yang. “Engaging Software Estimation Education Using LEGOs: A Case Study”. In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ICSE ’16. Austin, Texas: ACM, 2016, pp. 511–517. ISBN: 978-1-4503-4205-6. DOI: 10.1145/2889160.2889205. URL: <http://doi.acm.org.ezproxy.rit.edu/10.1145/2889160.2889205>.
- [21] Jonathan Lazar. “Integrating accessibility into the information systems curriculum”. In: *Proceedings of the international association for computer information systems* (2002), pp. 373–379.
- [22] Yang Liu et al. “Do Users Act Equitably? Understanding User Bias Through a Large In-person Study”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 2023, pp. 83–95. DOI: 10.1109/ICSE-SEIS58686.2023.00014.
- [23] Yang Liu et al. “Using Accessibility Awareness Interventions to Improve Computing Education”. In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*. ICSE-SEET ’24. Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 66–71. ISBN: 9798400704987. DOI: 10.1145/3639474.3640821. URL: <https://doi.org/10.1145/3639474.3640821>.
- [24] Thomas Howard Morris. “Experiential learning—a systematic review and revision of Kolb’s model”. In: *Interactive Learning Environments* 28.8 (2020), pp. 1064–1077.
- [25] *New Jersey Student Learning Standards in Computer Science (NJSLS-CS)*.
- [26] New York State Education Department. *Science and Technology Entry Program (STEP)*. URL: <https://www.nysed.gov/postsecondary-services/science-and-technology-entry-program-step>.
- [27] G. Michael Poor et al. “No User Left Behind: Including Accessibility in Student Projects and the Impact on CS Students’ Attitudes”. In: *Trans. Comput. Educ.* 12.2 (Apr. 2012), 5:1–5:22. ISSN: 1946-6226. DOI: 10.1145/2160547.2160548. URL: <http://doi.acm.org/10.1145/2160547.2160548>.

- [28] National Girls Collaborative Project. *Guide to Inclusive Computer Science Education*. 2022. URL: <https://ngcproject.org/sites/default/files/downloadables/2022-04/Inclusive-Classrooms-Guide.pdf>.
- [29] Katherine Shaw and Julian Dermoudy. “Engendering an Empathy for Software Engineering”. In: *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*. ACE '05. Newcastle, New South Wales, Australia: Australian Computer Society, Inc., 2005, pp. 135–144. ISBN: 1-920682-24-4. URL: <http://dl.acm.org.ezproxy.rit.edu/citation.cfm?id=1082424.1082441>.
- [30] Weishi Shi et al. “ALL: Supporting Experiential Accessibility Education and Inclusive Software Development”. In: *ACM Trans. Softw. Eng. Methodol.* (Sept. 2023). ISSN: 1049-331X. DOI: 10.1145/3625292. URL: <https://doi.org/10.1145/3625292>.
- [31] Weishi Shi et al. “Experiential Learning in Computing Accessibility Education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 250–251. ISBN: 9781450371223. DOI: 10.1145/3377812.3390901. URL: <https://doi.org/10.1145/3377812.3390901>.
- [32] Weishi Shi et al. “Presenting and Evaluating the Impact of Experiential Learning in Computing Accessibility Education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. ICSE-SEET '20. Distinguished Paper Award. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 49–60. ISBN: 9781450371247. DOI: 10.1145/3377814.3381710. URL: <https://doi.org/10.1145/3377814.3381710>.
- [33] *The ADA and Section 508*. URL: <https://508compliantdocument.com/americans-with-disabilities-act/>.
- [34] Rachael Thomas. *Diversity Crisis in AI, 2017 edition*. Aug. 2017.
- [35] *Twenty-First Century Communications and Video Accessibility Act*. URL: <https://www.fcc.gov/general/twenty-first-century-communications-and-video-accessibility-act>.
- [36] “Unlocking Equity: Navigating Resource Allocation Challenges in K-12 US Public Schools”. In: (2022).

Teaching Computer Security Through Lab Demonstrations and Simulators Created as Senior Programming Projects*

Edwin Dauber
Department of Computer Science
Widener University
One University Place, Chester PA, 19013
egdauber@widener.edu

Abstract

As part of a computer science curriculum, it is typical to separate individual fields into separate courses. Specifically for security, this is difficult due to the breadth of the field and the amount of knowledge from other fields necessary for a well-rounded understanding. While many larger institutions are moving towards programs in this field, many small liberal arts colleges do not have the resources to support this. In this work, we examine the difficulties of teaching security in this setting and compare techniques for teaching this difficult topic.

1 Introduction

The typical structure for undergraduate computer science education is for students to spend their first two years building their foundational skills, and then their remaining years taking specialized upper level elective courses in distinct areas of the discipline. For many fields which have distinct skill sets and knowledge bases, a single course or a two course sequence is sufficient to introduce students to the field and prepare them for beginner level jobs or graduate study.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

For especially diverse fields, additional stand alone courses or a second orthogonal sequence may be necessary, but the model still works. Under this model, these electives typically have no prerequisites other than reaching a minimal point in the core curriculum, and for the second course of a sequence having completed the first course in the sequence. This model allows departments to offer electives in alternating years, maximizing the breadth of courses available within the constraints of a small department. Many of these departments do not offer concentrations, but instead offer elective choice and sequences.

While this model is effective for many topics in computer science, computer security is an exception. Security is concerned with preventing malicious parties from exploiting vulnerabilities in computing systems. These vulnerabilities may be in any component of the system. Therefore, students of security need knowledge from across computer science to gain even foundational expertise. Example domains of knowledge required include data structures and algorithms, operating systems, networking, and computer architecture. Additionally, attacks can occur at a variety of levels, requiring foundational knowledge in multiple programming languages such as C or C++, Python or Java, HTML or PHP, JavaScript, shell scripting, and assembly language.

Given this challenge, it is understandable that many larger institutions are moving beyond the traditional model for their security offerings. For example, both Hoag and Parrish et al. identified the need for and worked to develop entire curricular programs to teach computer security [8, 13]. A joint task force including the ACM, IEEE-CS, AIS SIGSEC, and IFIP WG 11.8 have released guidelines for the development of such programs [3]. However, for smaller schools without the funding to invest in such a programming, it is necessary to find creative ways to cover the necessary topics without assuming that students have all of the required foundational knowledge and skills.

In this experience paper, we extend our previous poster by adding detailed observations and analysis with additional data to examine methods to enhance undergraduate computer security education within the bounds of available resources and demand [4]. We investigated varying instructional and assessment methods and activities within our security course, as well as investigating the use of student projects outside of the course setting. We also examine how using student-developed simulator projects can offer educational opportunities for both the developing students and for students in the security course.

2 Institutional Needs and Constraints

While our institution is formally a university, it functions in many respects as a small liberal arts college. Our department regularly offers three courses in computer security. Of these, two have narrow focus and the last must cover

the remaining breadth of computer security. To create additional courses there must be sufficient audience to justify the expense of running the course, which has not been present in recent years. Because the narrowly focused courses are clearly defined, this work will focus on the general security course as well as the opportunity offered by senior programming projects.

At our institution, we have several computer labs and a computer science network. However, there is no network or lab dedicated for security, in which students may safely practice attacks and defenses. In addition to the financial considerations in establishing such a lab, we would also need cooperation from both the computer science network managers and our institution's Information Technology Services (ITS) team. This would require the development of a lab with an isolated network with exceptions to typical ITS security policies and computers on the standard network for Internet access.

2.1 Audience

The security course is an upper level elective course primarily taken by junior and senior students majoring within our department, but also admitting high-achieving sophomore students. This is a small selection of students, each of whom only requires a few upper level electives and has many other choices including courses traditionally considered by students to be both easier and more fun. The course must also serve the occasional criminal justice student pursuing an interdisciplinary minor related to computer security and the legal system. These students have even less of the necessary technical background than can be assumed from the sophomore and junior computer science students.

2.2 Course Requirements

Given the limitations of the audience, it is impossible to add prerequisites and is in fact desirable to remove prerequisites. The more strict the prerequisites, the less achievable the minor becomes and the fewer students who will pursue security as part of their upper level elective selection. Conversely, decreasing prerequisites both makes the minor more feasible and increases the audience among the computer science students.

However, the course is still an upper level course and must have sufficient technical content and academic rigor. This sole course must cover sufficient material to give students the background necessary for careers or graduate study in any subfield of computer security. Such subfields include software security, network security, Internet security, and operational security. Ideally the course would also include topics in hardware security, blockchain, and digital privacy, but covering all of these topics at any sufficient depth in a single semester would not be possible.

2.3 Senior Programming Project

Senior programming projects, also called senior design projects, are a common capstone experience where graduating seniors work in small groups to complete a significant programming project. Depending on the institution, these projects may be proposed by the students, the faculty, or some combination of the two. While security is typically a challenging area, there are several possible projects in the field which may be completed by senior students. These projects may include cybersecurity tools or sandboxed demonstrations and simulations of cybersecurity attacks and concepts.

3 Related Work

There are many models for undergraduate computer and network security education. Yang and Nguyen developed a network security lab to be used following a nine-step network security process [16]. This model requires a high startup cost to develop a suitable lab environment, and does not cover all of the topics. Du et al. developed the SEED labs, a set of labs developed to provide a comprehensive education across the fundamentals of computer security [5, 6]. This model covers a wide range of fundamental topics, but the labs provided are difficult for students without sufficient background, and it can be difficult for students to diagnose mistakes. Zeiheck proposed a model in which existing online cybersecurity competitions were used in a course structured around student-centered learning [17]. Although competitions are good learning opportunities and cover a wide range of topics, such a course requires a pace set according to the competition schedule and is challenging for grading.

Other models cover more specific topics in security. Rahman et al. proposed a model for courses focused on Internet security involving five projects about digital certificates [14]. Although these projects provide excellent depth of understanding of the topic, this model is too narrow in focus to suit our needs. Buchele proposed two models for security education focusing on cryptography and including risk analysis, malware, and passwords with one model emphasizing the underlying mathematics as a mathematics course and the other emphasizing practical skills as a computer science course [1]. While these models include many good components, we already have a dedicated cryptography course and the remaining topics do not cover sufficient breadth for our needs.

In addition, some research has produced tools and evaluation methods to aid in teaching security. Chun et al. introduced SLOB to organize trusted educational materials for student search [2]. Although this technology can prove to be a useful aide, we first need a course in place before adopting assistive technologies. Švábenský et al. compared two graph methods to evaluate student learning of low-level skills [15]. In the timeframe available, we are more

concerned that students are introduced to a variety of skills and issues rather than achieve mastery of these low-level skills.

Another thrust of research has been towards cybersecurity courses for non-majors. Ledford et al. developed a Denial of Service (DoS) lab to be used in a general education computer science course [9]. This is a great standalone instructional activity which can supplement but not replace other methods. Mountroudou et al. developed a first year experience course focused on cybersecurity [11]. While this is an interesting model, the level is not high enough for an upper level elective course. Gorka et al. developed a modular general education cybersecurity course with no computing pre-requisites focused on identifying risks and correctly utilizing security technology [7]. These are useful topics which can supplement our work, but we also need technical components. Page et al. propose a security course for high school students [12]. This course is interesting and has a good breadth of topics, but like the first year experience course the level is not high enough for our needs.

4 Course Evolution

Based on our evaluation of existing course models, we initially decided to base our course on the SEED labs by Du et al. [6]. This way, we would have a starting model that covers most of the necessary topics, which we could then extend with remaining topics and adjust to suit the necessary degree of difficulty. Thus far, we have run three iterations of the course, with one each in the Fall 2021, Fall 2023, and Fall 2025 semesters. In this section, we will discuss the variations of the course as well as the outcomes of those variations.

4.1 Course Structures

Table 1 lists the SEED labs used in the courses, in the order used in Fall 2021, with labs used only in later semesters at the end. In this table, entries which read “Homework” mean that students were assigned to complete the lab outside of class. In 2021, students had one week, although there were no penalties for late submissions. In 2023, students had two weeks, and the lab was demonstrated in class on or before the due date. In 2025, students had at least three weeks. Entries in the table which read “Home w/ Alt” mean that students were given an option between completing the SEED lab or an alternative activity on the subject with similar learning outcomes and difficulty. Entries which read “Demoed” mean the course instructor demonstrated the full lab in class, and students were expected to otherwise express what they learned through observation and following along. Entries which read “Discussed” mean that the topic was covered at high level in lecture and segments of the lab may have been demonstrated, but the whole lab was not used and mastery of low

level detail was not expected. Meanwhile, entries reading “Extra” mean that the topic was only provided as an extra credit opportunity and “N/A” means that it was not used in that version of the class.

Table 1: SEED Lab Usage

Lab	Fall 2021	Fall 2023	Fall 2025
Set-UID & Env. Var.	Homework	Homework	Demoed
Shellshock	Homework	Discussed	Demoed
Buffer Overflow	Homework	Demoed	Demoed
Return to libc	Homework	Demoed	Demoed
Format String	Homework	Demoed	Demoed
Race Conditions	Homework	Demoed	Demoed
Dirty COW	Discussed	Discussed	N/A
Meltdown & Spectre	Discussed	Discussed	N/A
CSRF	Homework	Home w/ Alt.	Demoed
XSS	Homework	Home w/ Alt.	Demoed
SQL Injection	Homework	Home w/ Alt.	Demoed
Packet Sniffing	Homework	Homework	Demoed
TCPIP Attacks	Homework	Discussed	Demoed
Firewall Exploration	Homework	Discussed	Demoed
Firewall Evasion	Homework	Discussed	Demoed
DNS	Homework	Discussed	Demoed
VPN	Homework	Discussed	N/A
Heartbleed	Homework	Discussed	Extra
Shellcode	N/A	N/A	Homework
ARP Cache Poisoning	N/A	N/A	Demoed
ICMP Redirect	N/A	N/A	Homework

In addition to these topics from the SEED labs, the course covers other essential security topics. Table 2 lists these topics and whether or not they were covered in each iteration of the course. By de-emphasizing many of the SEED labs, we were able to cover a wider selection of topics.

4.2 Student Created Simulations

We have worked with two senior project teams on constructing simulations for security concepts for use in the course. One of these projects was completed in the 2022-2023 academic year and was used in the course in both the Fall 2023 and Fall 2025 semesters. The other project was completed in the 2024-2025 academic year and was used in the Fall 2025 semester of the course.

Table 2: Additional Security Topics

Lab	Fall 2021	Fall 2023	Fall 2025
Security Goals	Covered	Covered	Covered
Security Properties	Covered	Covered	Covered
Side-channels	Covered	Covered	Not Covered
Redundancy & Backups	Covered	Covered	Covered
Blockchain	Covered	Not Covered	Not Covered
Mobile & IoT	Covered	Covered	Covered
Authentication	Covered	Covered	Covered
Access Control	Not Covered	Covered	Covered
Social Engineering	Not Covered	Covered	Covered
Security Audit & Policy	Not Covered	Covered	Not Covered
Password Attacks	Not Covered	Covered	Covered
Cross-Site Contamination	Not Covered	Covered	Not Covered
DoS & Web Crawlers	Not Covered	Covered	Not Covered
Zip, Fork, and Tar Bombs	Not Covered	Covered	Not Covered

4.2.1 Academic Year 2022-2023 Project

This project focused on software security issues, implementing Java simulations related to buffer overflows, return oriented programming, and race conditions.

For the buffer overflow simulation, a character array is used to represent program memory, with a simple GUI to display the contents of the array as a grid. The simulation then represents a simplified buffer of five characters followed by a five character return address and five character stack pointer as the last fifteen characters in the memory array. The simulation also includes a set of flags to determine whether certain conditions occurred. Specifically, it sets an overflow flag if the input is larger than five characters, a return address flag if the return address has been altered, a success flag if the return address is changed to a specific value, or a libc flag if it is changed to one of a collection of other values. If the overflow is too large, it causes the simulation to crash. This simulation removes a lot of the guesswork for students while still demonstrating the essential mechanism of the buffer overflow as well as possible consequences.

To simulate return oriented programming, the senior students relied heavily on abstraction. Instead of having functions stored at locations in memory and a stack with memory addresses artificially inserted, the abstraction contains an array of objects where the indices represent memory addresses and the objects include a string representing a function and an integer representing the next memory address on the stack, with too large of an index or an index equal to current index indicating that the simulated attack should stop. Students

interact with the simulation through simple method calls. First they construct the array by repeatedly calling a method with three parameters: the index at which to put the object, the string to put there, and the next index to visit. Once the array is sufficiently filled, they will call another method to run the simulation. This method may either be called with no parameter indicating that it should start from index 0 or with a single parameter indicating the index at which to start. The program then follows the order of the specified next indices in order to print a message which is not obvious by the order in which strings are stored in the memory array.

For the race condition simulation, the project focuses on the timing aspect. Instead of demonstrating the effect, such as an invalid read or write, the project produces five threads which cycles between three states at random timing and stops if ever all five threads are in the second state. In this way, the project allows students to visualize how processes may interleave in unpredictable ways, but if given sufficient time and built-in delays can produce adverse results.

4.2.2 Academic Year 2024-2025 Project

This project focused on network security issues. Specifically, the project creates a Python simulator for configuring firewalls. This is a lower level simulation, including a simulated command prompt. However, the commands are simplified from real command line commands, and the command prompt offers clear and immediate feedback to the user. The simulation allows students to practice configuring basic firewalls in a series of tasks inspired by the relevant SEED Lab. This allows students to practice with the concepts of firewalls without getting stuck in the specifics of the syntax.

4.3 Study Design

Our study focuses on student experience and engagement while learning security. We use classroom observations, anonymous standardized course evaluations, and in the Fall 2025 semester an additional anonymous survey to collect more detailed information about specific activities. The study was approved in an expedited review by our Institutional Review Board. Participation was opt-in for students in the Fall 2025 semester, while data from earlier course iterations were included under retrospective IRB approval, subject to the same scope and conditions as newly collected data. The survey was anonymous, and any non-anonymous student-focused classroom observations were usable once consent forms were released along with course evaluations following submission of final grades. Instructor reflections on pedagogical approaches and content delivery were also available. Data was stored in accordance with institutional requirements. Sample sizes were small, with six students in the Fall 2021 sec-

tion of the course and seven students in Fall 2023. In Fall 2025, seven students completed the anonymous survey and course evaluation and four signed consent forms for observation. In addition to the three runs of the course, both of the senior projects were completed by teams of three students.

4.3.1 Course Evaluations

The course uses a standardized course evaluation survey. The survey asks five questions. The first question is a Likert scale question with options “Very,” “Somewhat,” “A little,” “Not very,” and “Not at all” [10]. The question asks how interested the student was in learning the topics covered in the course. The second question asks students how many hours they spent outside of class on the course, with options of “None,” “1-3,” “4-6,” “7-9,” and “10 or more.” The remaining three questions were open ended and asked for the most important things they learned from the course, specific ways the instructor and course facilitated their learning, and suggestions they have for the instructor.

4.3.2 Survey Design

The survey consisted of three Likert scale questions with answers “Strongly Agree,” “Agree,” “Neither agree nor disagree,” “Disagree,” and “Strongly Disagree.” For each question, respondents were asked to react for each of the following topics: “Buffer Overflows,” “Return Oriented Programming,” “Format String Vulnerabilities,” “Packet Sniffing and Spoofing,” “Firewalls and VPNs,” “Network Protocol Vulnerabilities,” “Cross-Site Request Forgery,” “Cross-Site Scripting,” and “SQL Injection.” Respondents were permitted to skip any topic or question they wished. The questions were:

1. For each of the following topics, how would you react to the following statement: In-class and homework activities worked together to enhance my learning.
2. For each of the following topics, how would you react to the following statement: I had a good experience with the in-class activities related to this topic.
3. For each of the following topics, how would you react to the following statement: I had a good experience with the homework activities related to this topic.

4.4 Outcomes

4.4.1 Project Observations

The students involved in creating the simulation projects had not previously taken the security course. As such, they were learning the security content for the first time. However, these students had taken other relevant courses, and so were familiar with topics related to operating systems and networking. The students were generally self-directed, and worked together to learn the necessary concepts. By focusing on a small selection of topics, the students were able gain important insights outside of a classroom setting.

4.4.2 Course Observations

In the Fall 2021 semester, it was typical for two or three students to attend office hours to work on labs. There were many late submissions. Students self-reported as spending several hours per week on the labs. While the instructor was able to give an appropriate time to both theory and practice during lecture, during office hours the time was spent working through minor practical details.

In the Fall 2023 semester, office hours were not often used, but students admitted to searching for videos online for guidance with labs. Unlike in Fall 2021 where student struggled with lab details, in this run of the course the primary struggles were the the emulation technologies including virtual machines and containers. When given the choice between a SEED lab and an alternative activity, most students chose the alternative, but students were typically attentive during lab demonstration. The instructor noted that engagement was a challenge during the traditional lectures, but demonstrations were fairly successful, even if some labs were only partially completed.

In the Fall 2025 semester, the instructor noted some challenge with the SEED lab demonstrations. In particular, the labs went more slowly than the previous year, due to being the primary content delivery mechanism. Because of this, it was rare for the instructor to be able to give time for the labs to be interactive, instead of just a lecture through demonstration. The consenting students were noted to be highly engaged, asked questions, and worked both independently and collaboratively to understand lab tasks.

4.4.3 Student Course Evaluations

In the Fall 2021 semester, two students completed the course evaluation. Both reported high interest and an average between four and nine hours outside of class per week. Both highlighted the importance of learning the different attacks. One commented on the instructor's help with the labs, and one suggested using the SEED project slides instead of the whiteboard for the lecture.

In the Fall 2023 semester, six students completed the course evaluation. While one reported disinterest in the topics, most reported being either “Very” or “Somewhat” interested. Most reported spending between four and six hours per week outside of class, but some reported spending only one to three hours per week. Students highlighted the network and Internet security aspects of the class. Many recommended spending more time working through the labs in class and giving more alternative activities for the homework assignments. Multiple students suggested not using the slides during lecture.

Seven students completed the Fall 2025 evaluations. They reported being either “Very” or “Somewhat” interested and most reported spending one to three hours per week outside of class. They praised the selection of homework activities and course topics. Common recommendations related to the pace of the course and the challenge of exams.

4.4.4 Survey Results

We note that no students responded “Strongly Disagree” to any question. Table 3 provides the response count for question 1. These responses show that generally students found the assigned activities helpful with learning. We can also see that certain activities were seen as more or less helpful than others.

Table 3: Question 1: In-Class Activities and Homework Enhance Learning

Topic	Strongly Agree	Agree	Neither	Disagree
Buffer Overflow	0	6	0	1
ROP	1	3	2	1
Format Strings	1	4	1	1
Packet Sniffing	2	1	3	1
Firewalls & VPNs	3	1	3	0
Networks	0	3	3	1
CSRF	1	3	2	1
XSS	1	4	1	1
SQL Injection	1	4	2	0

Table 4 shows the responses from question 2. There was one omission. These responses demonstrate that overall students had positive or neutral experiences with the demonstrations. The format string demonstration was seen most neutrally, while the buffer overflow demonstration was overall positive, and the Internet security topics were least likely to be seen negatively.

Table 5 shows the responses from question 3. There was one omission and responses tended to be more neutral. However, we see that students still generally reported positive experiences for some topics.

Table 4: Question 2: Good Experience with In-Class Activities

Topic	Strongly Agree	Agree	Neither	Disagree
Buffer Overflow	0	5	2	0
ROP	0	4	2	1
Format Strings	1	1	4	1
Packet Sniffing	0	3	3	1
Firewalls & VPNs	1	3	2	1
Networks	1	2	3	1
CSRF	1	3	3	0
XSS	1	3	3	0
SQL Injection	1	3	2	0

Table 5: Question 3: Good Experience with Homework

Topic	Strongly Agree	Agree	Neither	Disagree
Buffer Overflow	0	4	2	1
ROP	0	2	4	1
Format Strings	0	0	5	1
Packet Sniffing	2	2	2	1
Firewalls & VPNs	2	1	3	1
Networks	1	1	4	1
CSRF	0	4	3	0
XSS	0	4	3	0
SQL Injection	0	4	3	0

5 Discussion

5.1 Outcome Analysis

We begin by noting our small sample size. With such small class sizes, clear generalizations are difficult to make, but still some trends can be observed. Our outcomes are consistent in a few important ways. Student learning is enhanced through use of the SEED labs, but students find many of them very difficult. By demonstrating them in class, the negative impacts of this difficulty on student experience can be mitigated. Students are generally satisfied with the Internet security labs, if they can successfully navigate the container configuration. Packet sniffing and spoofing and the firewall exploration and evasion portions are generally more successful than the attacks on specific networking protocols. The software security section is generally successful, but format

strings require more attention in both the experience of the lab demonstration and in development of an effective homework activity. Students were similarly positively receptive towards both the best developed student project activities (buffer overflows and firewalls) as towards Internet security homework activities. However, other topics require further development in this area. Maintaining engagement and momentum in any lecture is a challenge, but interactive components can improve it. Continuing in the direction of demonstrated labs paired with related homework activities is likely to continue to improve the student experience and increase engagement.

5.2 Framing Student Projects

Using student projects is highly desirable due to the learning opportunity for the developing students. It is also the case that, being developed by students for students, these projects often target specific areas of interest in ways which are approachable for students with similar educational backgrounds. However, students are not pedagogical experts. It is important that the instructor properly frame these activities, such as by making the abstraction into a metaphor or asking targeted questions encouraging students to think through the levels of abstraction and how the abstraction relates to the actual attack. By doing so, students can have a more digestible experience while still demonstrating insight into the attacks covered in class.

6 Future Work

Once the course is able to be run with sufficient class size, we will create a second semester course. This will allow many topics to be given greater depth, and additional topics to be added. We would also like to add additional alternate activities. While the labs themselves are excellent learning tools, students need opportunities to struggle in unsupervised settings. This study focused on student experience and engagement, with student performance out of scope. A follow up study including these features would be beneficial, as would increased sample size.

7 Conclusion

Computer security is exceptional among fields of computer science for the relationships to other fields and the challenge of teaching it at the undergraduate level. The SEED project by Du et al. is an effective and comprehensive attempt to provide a standardized set of hands-on educational activities. However, these activities are particularly challenging for undergraduate students

at small liberal arts colleges. By shifting them from individual assessment to collective learning activities and complementing them with simpler independent reinforcement, the educational value of these activities can be provided in a manner which decreases student stress and increases overall engagement. When senior students construct simulation tools for these reinforcements, they gain a valuable learning opportunity and provide tools which students will understand and appreciate. This in turn makes education in this essential area of computer science more approachable for a wider audience.

References

- [1] Suzanne Fox Buchele. “Two models of a cryptography and computer security class in a liberal arts context”. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE ’13. Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 543–548. ISBN: 9781450318686. DOI: 10.1145/2445196.2445360. URL: <https://doi.org/10.1145/2445196.2445360>.
- [2] Soon A. Chun et al. “Slob: security learning by ontology browsing: comprehensive cyber security learning resources in a web portal”. In: *J. Comput. Sci. Coll.* 31.5 (May 2016), pp. 95–101. ISSN: 1937-4771.
- [3] Joint Task Force on Cybersecurity Education. *Cybersecurity Curricula 2017: Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity*. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 9781450389198.
- [4] Edwin Dauber. “Examining Computer Security Education Activities for Liberal Arts Colleges”. In: *Journal of Computing Sciences in Colleges* 40.8 (2025), pp. 30–31.
- [5] Wenliang Du, Zhouxuan Teng, and Ronghua Wang. “SEED: a suite of instructional laboratories for computer SEcurity EDucation”. In: *Proceedings of the 38th SIGCSE technical symposium on Computer science education*. 2007, pp. 486–490.
- [6] Wenliang Du and Ronghua Wang. “SEED: A suite of instructional laboratories for computer security education”. In: *Journal on Educational Resources in Computing (JERIC)* 8.1 (2008), pp. 1–24.
- [7] Sandra Gorka et al. “A Cybersecurity Course for Everyone”. In: *Proceedings of the 21st Annual Conference on Information Technology Education*. SIGITE ’20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 409–412. ISBN: 9781450370455. DOI: 10.1145/3368308.3415399. URL: <https://doi.org/10.1145/3368308.3415399>.

- [8] Jim Hoag. “Evolution of a Cybersecurity curriculum”. In: *Proceedings of the 2013 on InfoSecCD’13: Information Security Curriculum Development Conference*. 2013, pp. 94–99.
- [9] Harrison Ledford, Xenia Mountroudou, and Xiangyang Li. “Denial of service lab for experiential cybersecurity learning in primarily undergraduate institutions”. In: *Journal of Computing Sciences in Colleges* 32.2 (2016), pp. 158–164.
- [10] Rensis Likert. “A technique for the measurement of attitudes.” In: *Archives of psychology* (1932).
- [11] Xenia Mountroudou, Xiangyang Li, and Quinn Burke. “Cybersecurity in liberal arts general education curriculum”. In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 2018, pp. 182–187.
- [12] Liberty D Page, Mehdi Mekni, and Elizabeth A Radday. “Incorporating cybersecurity concepts in connecticut’s high school stem education”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 173–187.
- [13] Allen Parrish et al. “Global perspectives on cybersecurity education for 2030: a case for a meta-discipline”. In: *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*. 2018, pp. 36–54.
- [14] Shamima Rahman, Tuan Anh Nguyen, and T Andrew Yang. “Developing certificate-based projects for web security classes”. In: *Journal of Computing Sciences in Colleges* 21.5 (2006), pp. 28–37.
- [15] Valdemar Švábenský et al. “Evaluating two approaches to assessing student progress in cybersecurity exercises”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 2022, pp. 787–793.
- [16] T Andrew Yang and Tuan Anh Nguyen. “Network security development process: a framework for teaching network security courses”. In: *Journal of Computing Sciences in Colleges* 21.4 (2006), pp. 203–209.
- [17] David Zeichick. “Successfully incorporating a cyber security competition into an intro to computer security course”. In: *J. Comput. Sci. Coll.* 38.1 (Nov. 2022), pp. 58–67. ISSN: 1937-4771.

Detection of Spinning Behavior with a Known Solution*

Mark Sherman¹, Mohit Chandarana², and Erica Hinkle¹

¹Emmanuel College

Boston, MA

{shermanm,hinklee}@emmanuel.edu

²Codio, Inc.

New York City, New York

mchandarana@codio.com

Abstract

In classroom and online learning environments, identifying which students need help at any moment is challenging. Students often enter a state of “spinning,” continuing to work without making progress, and would benefit from timely intervention. We are developing a real-time system to detect spinning using behavioral patterns from students’ programming editors. We collected fine-grained, often keystroke-level data from a Massively Open Online Course (MOOC) programming environment. In the first phase, we focus on assignments with known correct solutions, developing tools to measure students’ distance from the goal using Levenshtein and AST edit distances, revealing proximity to or struggle toward the correct answer. By segmenting work into active sessions, we map progress over time across 28,000 students and 70 exercises, revealing improvement, divergence, and sustained effort without progress. We find that spinning often involves ups and downs rather than stagnation. Behavioral features extracted from these episodes will train a machine learning model in Phase II to detect spinning when solutions are unknown, enabling smarter, more responsive learning tools for online and classroom orchestration. We present our analytical approach, findings on student behavior patterns, and hypotheses for future work.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Our goal is to develop a real-time algorithm or model capable of detecting when students enter a state we call “spinning.”

This refers to a form of nonproductive student effort characterized by continued work without progress. Students experiencing spinning often lack a clear strategy for how to proceed in an assignment yet continue attempting to edit or run code without improving it, thus not advancing toward a working solution. We propose that we can address this futile perseverance by creating an algorithm that flags “spinning” behavior. That algorithm could inform other mechanisms to deploy instructional interventions, such as an instructor dashboard indicating to help a particular student, or an automated tutor system offering a hint, or others. Because so much student work occurs outside class and instructors are not always present to intervene promptly, automated detection of spinning has the potential to provide timely support, build student confidence, and improve learning outcomes.

We approach this problem in two phases. Phase I focuses on exercises with a known set of correct solutions. Most CS1 exercises fall into this category. These programming tasks allow students to write and debug code within an editor while converging toward a well-defined goal. Where the goal state is known, we can track progress towards it over time. These progress trajectories allow us to algorithmically identify likely spinning episodes, which then serve as labels for the next phase, our machine learning model. The primary outcome of Phase I is a method for algorithmically identifying likely spinning movements within student sessions, enabling these moments to serve as labeled training instances for subsequent machine learning development.

This paper reports on the progress of this first phase and posits ideas and hypothesis as we go into the second phase: generalization.

Phase II will generalize spinning detection to assignments without known solutions. Using the labeled data from Phase I, we will extract features from edit histories and their corresponding likelihood-of-spinning metrics to train a machine learning model capable of inferring spinning solely from behavioral signals. The resulting system will estimate, in real time, whether a student needs intervention, regardless of exercise format.

2 Research Questions

Table 1: Research Questions

ID	Question
RQ 1.1	What features or patterns within edit data indicate spinning?
RQ 1.2	If identified, can these patterns/features be charted? Can likely spinning be assessed by a human looking at these charts?
RQ 1.3	How can detection of spinning be automated?
RQ 2.1	What features can be extracted from edit data to indicate spinning?
RQ 2.2	How well do these features, individually or clustered, predict spinning?
RQ 2.3	What machine learning architecture can be used to generalize estimation of spinning?
RQ 2.4	How can this ML model be verified?

3 Background

Previous studies have used fine-grain instrumentation through instrumentation of the IDE or student work environment to gain access to learner behavior. These studies varied widely, from the discovery of phases of development over a project [3], to analysis of off-task behavior in learning [2]. Piech et al. used IDE instrumentation to for identify struggling students [15]. All of them had commonalities: building models of learning by looking at fine-grain behavior data, observing small changes in work over time, filling in the stories that led up to the final artifacts. Examining the process of programming rather than the final products can uncover interesting behavior about student learning, and such patterns may have better predictive power than in-class exams [4].

3.1 Challenges At Scale

Prior work using fine-grained programming-process data has shown both the promise and the limitations of behavioral metrics for understanding student programming progress at scale.

Chandarana and Deitrick’s replication of EQ, Watwin, and RED across 16,000+ learners demonstrated that these metrics lose predictive power for grades when applied across diverse contexts, yet their variability appears to capture students’ struggle more sensitively than grades do, suggesting their value as real-time indicators of when learners need help rather than as outcome predictors [5]. Complementary large-scale analyses of compiler error data further show that students frequently spend substantial time resolving a small

set of difficult error types—especially syntax, name, logical, and runtime errors—which can induce extended periods of non-progress [7]. This aligns with observations that episodes of getting stuck are not simply idle time but involve repeated, often unsuccessful attempts to move forward. Prior exploratory work conducted by Codio focused on leveraging integrated development environments for learning analytics, but these early investigations were not oriented toward producing an operational real-time intelligent tutoring system, which distinguished the objectives of the present study. Similarly, recent large-scale work on time-on-task shows that while activity measures can predict outcomes in small, controlled settings, this breaks down across diverse courses. Segmenting time into “error-state” and “error-free” work also failed to improve predictions, because both types of activity often represent genuine effort. In other words, students’ progress cannot be inferred from time alone: even long periods spent fixing errors may reflect meaningful work rather than being stuck [6]. Together, these studies underscore the need to shift from predicting performance to identifying moments of struggle within students’ programming process.

3.2 Predicting Student Success

Prior work in identifying student risk of failure or attrition is related to the idea of identifying nonproductive perseverance, though the scope differs. Falkner & Falkner argue that early identification of students can help to catch at-risk students ahead of time [11]. Specifically, students that turned work in late or not at all were deemed to be more at-risk and could be identified at an earlier point, helping to prevent them from dropping out or disappearing. Similarly, Alice Armstrong used test and placement scores to prove that these were worthy diagnoses of a student’s success. Providing a framework, outlined by the foundation, understanding and sense of belonging in a community allowed the student to want to work in a more rigorous environment [1].

Pei and Xing’s work proposed a dynamic format to the classroom that allowed instructors to adapt to student progress week over week—and was easily adapted to in-person or online classes. Their data changed on a week to week basis and, critically, was accessible to the instructors. This allowed them to make active changes in their classroom to improve retention rates [14].

Finally, Falkner et al. utilized assignments and early identification to also improve retention rates, where it was the students’ reactions to the professor that provided the data [11]. This was quite different to the other studies that relied on strictly quantitative measures of student progress.

This shows there are many ways to assess and predict a student’s success. The goal of this project does differ from those above in a significant way: we want to assess a smaller goal (is the student spinning), and do so much faster,

moment-by-moment, rather than over weeks or semesters.

3.3 Machine Learning

Machine learning has also been used in assessing and predicting student success. Pei and Xing created a machine learning program that measured the success rates and completion data of the students on a week to week basis. Their algorithm collected data that was in constant motion based on the success or failures of a student that week. This allowed the professors and instructors to have a live insight into how the student was doing. While it did not directly address futile perseverance, there were commonalities among the intentions of the research in understanding student success or failure [14]. Barkam et al. proposed that machine learning could be used to identify students at risk of probation in computer science, but drew the conclusion that it was based on identity, such as race, income, generation status. The machine learning component provided an interesting first perspective into possible design ideas, but with the introduction of identity politics, the data became less objective [10]. Such a machine learning model would inherently be biased. We have no such demographic data in our study, providing a lower risk of one of the most common pitfalls of machine learning automation.

4 Methods

We used fine-grain data from a MOOC to analyze whether students working on an assignment are in need of help. We benefit from access to data from Codio, an online learning platform that was already recording data for a playback feature, where learners and instructors can replay the work in a given file as if it were a video.

To calculate these student code metrics, we collected data from thousands of learners around the world who used the Codio platform between October 2023 and April 2025, following IRB-approved collection and de-identification protocols. The data was generated from a MOOC-style, asynchronous course taught in Python. The system is also instrumented to log keystroke file histories for each file a user creates/adds/edits in the platform. These keystrokes are not particularly fine-grained, meaning not all key presses and latencies between key presses are logged, a design decision made to optimize storage costs. Roughly, user activity in a 1.5 second interval represents a keystroke event in the system. This would include typing, deleting, copying, pasting, etc. These edits are the “fine-grain” data. We analyzed these data using a learning-analytics approach, centering on interpreting log-level actions to infer learner behavior and identify patterns that can inform instructional design [3].

4.1 The Course

The course we are studying is a comprehensive “CS1” offering in Python. It is a fully self-paced, self-contained course, with all instructional materials and assessments built into a single, interface. The course is divided into modules, each on a large topic. Within a module are 3-9 assignments, which can be thought of like chapters of a textbook. With this learning platform, an assignment can have a mix of content, including didactic and instructional text, coding exercises (either from blank file or with starter code), and a variety of in-narrative interactive elements, including: turtle graphics [12], parsons problems [9], code visualizers, and traditional multiple-choice and fill-in-the-blank-style diagnostic questions. Each module then has a “lab” assignment for larger scenarios in which to apply the new ideas, and concludes with an “exercises” assignment, which most closely resembles a traditional homework assignment. **These “exercise” assignments are the sole subjects of our study**, as they are free-form, offering a blank file, instructions, and a button that will compile and run the code. They sometimes include starter code.

Content begins with simple skills in typing in code and printing text to console, and goes on to include everything expected in a CS1: Operators, Conditionals, Loops, Lists, Strings, and Functions. It further includes modules on file access, recursion, object-oriented programming, and other more advanced topics.

4.2 Early Machine Learning Attempts

In earlier stages of this project, we experimented with training a neural network to predict whether a student was “spinning.” This exploratory work yielded two important insights that ultimately shaped the direction of the present study. First, we found that the dataset we had available at the time was too small and lacked sufficient complexity for a model of this type. When trained, the network achieved nearly 100% accuracy on both the training set and the test set. Rather than indicating successful learning, this pattern suggested severe overfitting. The model was ultimately memorizing the limited and highly uniform examples rather than generalizing to underlying patterns in student behavior.

A representative example illustrates the issue. One of the early exercises asked students to print stars in increasing quantities across five lines, with a typical correct solution resembling:

```
for num in range(1, 6):
    for dots in range(5 - num, 0, -1):
        print(".", end="")
    print(num)
```

This task involves minimal abstraction and admits only a narrow range of syntactic variation. As a result, it provided the neural network with little structural diversity to learn from, and students themselves tended not to struggle with this type of problem. Compounding this limitation, we had only a little over 300 student submissions for this specific exercise (Exercise 5 in that module), further restricting the model’s ability to generalize.

These observations made clear that a neural-network based approach would require both substantially more data and data drawn from later, more conceptually challenging modules where student work exhibits greater complexity and variation. This realization directly motivated our decision to expand the dataset and to pursue the feature-engineering approach described in the following sections.

4.3 Extracting Features

We still need data to serve as input to the generalizable model, since in a production scenario we will not have the solution code from which to calculate distances. This means we need to extract meaningful signals directly from the students’ edit data. Our initial attempt simply fed the full code state—treated as a text blob—into an ML model. This approach failed for several reasons, the most important being the lack of sufficient data. More generally, this confirmed that placing large amounts of raw data into an AI system hoping to draw meaningful conclusions was unrealistic, even though it was worth trying in case it revealed early promise. Combined with the limited subset of exercises available at the time, this highlighted the need for explicit feature engineering rather than relying solely on unstructured code snapshots.

We returned to our plan of extracting explicit features that could be used either alongside or instead of the raw code. These features include granular edit operations (insertions and deletions), full code snapshots after each modification, timing-related variables, and distances derived from the reference solution, all of which provide behavioral signals independent of the final answer. The available data includes each student’s full edit history, information about when code was executed, and the responses returned by automated tests. Based on this, we developed a collection of features to extract, shown in Table 2.

Having identified the set of available features, the next step was to determine how these signals could be operationalized for model training. In our early modeling attempts, the extracted features were compiled into per-edit feature vectors, with each vector representing a student’s state at a specific moment in their editing process. These vectors incorporated indicators such as the type and number of characters inserted or deleted, the location of the modification within the file, the timestamp of the event, the computed edit distance from the

Table 2: Features Extracted from an Edit

Feature	Type	Description
Code State	str	Full state of the file at the conclusion of this edit
Time Stamp	num	Time since start of session
Position	int	Character position of this edit in the file
Parseable	bool	Can code be parsed
Time Delta	num	time period since the previous edit (in seconds)
Insert	bool	was this edit an insertion or a deletion
Insert/Delete Ratio	num	ratio of recent insertion vs. deletion
Number of Runs	int	how many times code has been run
Session Length*	num	Total time of the session the edit is from
Session Interval*	num	Time since previous session

*denotes feature is fixed for all edits in a given session

reference solution, and the full code snapshot at the moment. While this representation provided a well-structured and expressive input format, our labeling strategy introduced a critical limitation. We initially assigned spinning labels at the student level rather than at the edit level, effectively treating an entire trajectory as a single classification instance. This labeling obscured essential meaning and contributed to substantial overfitting, as the model memorized specific students rather than generalizable behavioral patterns. This revealed that our earlier labeling strategy, classifying entire students rather than specific moments or segments, was a fundamental flaw that prevented the model from learning meaningful behavioral patterns. Recognizing this flaw motivated our transition to session and segment-level labeling approaches described in the following sections.

5 Analysis

For each specific student code edit, we calculate the student’s distance from the known solution using both Levenshtein edit distance of the code text [13]. Plotting this distance over time creates a trajectory representing progress, regression, or stalling. Ideally, a student who is making progress exhibits a downward trend in the chart, moving progressively closer to zero. This distance-over-time curve becomes the data for the next step, which is determining when spinning occurred. This produces charts like Figure 1, which shows a student progressively working towards the solution, shown by their distance from the canonical solution decreasing over time.

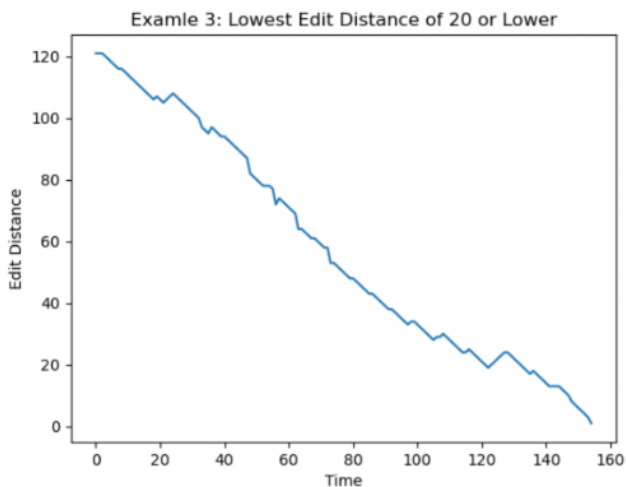


Figure 1: Edit Distance Over One Exercise

5.1 How Low is Low Enough?

A central question guiding our early analyses was: *What are the lowest (best) edit distances students achieve?* Minimum distance offers more than a simple measure of success, it indicates *how close* students came to constructing a viable solution at any point in their work. Students who reach a small distance, even briefly, typically demonstrate partial conceptual understanding, while persistently large distances suggest that they never formed a correct structural approach.

Analyzing minimum distances across roughly 28,000 students and 70 exercises revealed several important patterns. First, struggling students did not always remain at a fixed distance. Many cycled between scored, repeatedly moving closer to and then further from the solution. These oscillations showed that spinning can involve active reworking rather than simple stagnation, a nuance lost when looking only at correctness or final submissions.

Second, the distribution of minimum distances varied significantly by exercise. Tasks with narrow clusters of low distances tended to guide most students toward the intended solution structure, whereas tasks with wide or skewed distributions indicated broader conceptual barriers. This provided a way to identify exercises where spinning was more likely to be prolonged or widespread.

Finally, we found that reaching a low distance early in a session did not prevent later spinning; some students drifted into lengthy plateaus even af-

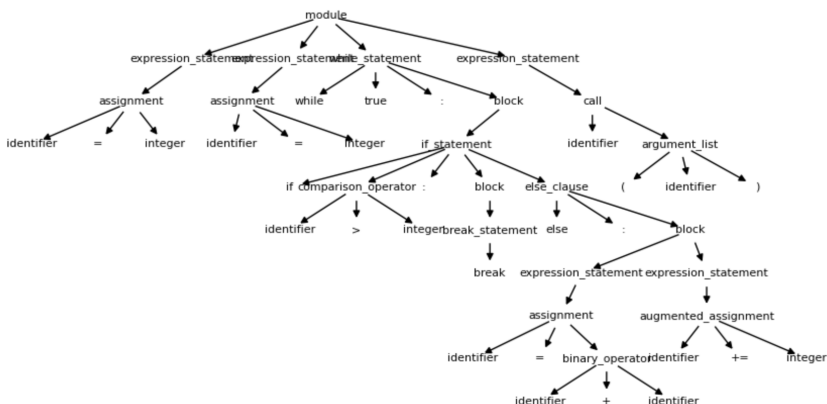


Figure 2: Abstract syntax tree of a solution

ter approaching the solution closely. This underscored the need for session- and segment-level labeling rather than student-level labeling, since a single attempt could contain both productive progress and extended nonproductive effort. Overall, minimum edit distance proved informative, not as a standalone indicator, but as a lens for interpreting the shape and timing of students' trajectories, insights that directly influenced our feature design and labeling strategy.

Later we added a similar analysis using abstract syntax trees of the code state. The edit distance between the AST of the student code and the known solution were calculated with the Zhang-Shasha algorithm [16]. These distances are a superior measure than text edit distance as they are immune to whitespace differences, variable names not matching, and other semantically irrelevant differences. A syntax tree of a solution can be seen in Figure 2.

5.2 Dividing into Sessions

Spinning is inferred from extended period of active editing where the distance remains unchanged or increases. For our specific goals, long periods of inactivity are excluded, as inactivity does not represent perseverance. In fact, long periods of inactivity proved to be problematic in our analysis, as many students will formally start an exercise, creating the file, but not start work for a period of time. As educators we do encourage the practice of taking a

break, but the variance in the breaks, along with initial idle time, skew the data. We also observed substantial variation in idle times, ranging from minutes to weeks, so we segmented work into sessions separated by more than three minutes of inactivity, a threshold supported by prior work [8]. Each session thus represents a contiguous period of active problem-solving, enabling clearer behavioral pattern identification. This segmentation also generalized conditions across sessions, as any given session could be from the beginning, middle, or end of an exercise, or the entirety of an exercise in a single sitting. This improved comparability among the data and allowing behavioral trends to be identified without effects from prolonged idle periods.

An individual session is a sequence of edits committed by a single student on an exercise, without any break periods. An individual student may have multiple sessions within a given exercise. Sessions are more comparable with each other, as they all have zero lead time from the start of the data until the first edit. With this new organization of the data, we could more accurately find behavioral patterns in the edits, allowing us to more easily create filters to detect when students were spinning. Sessions were a recent addition to the analysis, and future work will be based entirely on data organized into sessions.

6 Results

6.1 Edit Distances

An early question was how well students performed on the task overall. Recall that a score of zero indicates character-identical code to the canonical solution. It is possible to have a correct solution that is not identical to the canonical solution, as the text edit distance shows every white space, misspelling, or variable name choice as characters differing. Using the edit distances, we can extract the closest moments students got to a certain solution, as seen in Figure 3. Many students came quite close to zero, with only a few in this exercise having more than 30. In this specific exercise we chose 20 as the boundary of correct: everyone with a score under 20 had the right answer. This boundary varied by exercise. Ultimately this demonstrated that the distance-from-solution metric had power to determine student correctness.

6.2 Assessment of Spinning

We found that spinning could be assessed by humans looking at an edit distance chart with the corresponding code states. Over 60% of students analyzed showed a progressive downward trend, such as shown in Figure 1. They were not strictly monotonic, which, with support of looking at the code states over

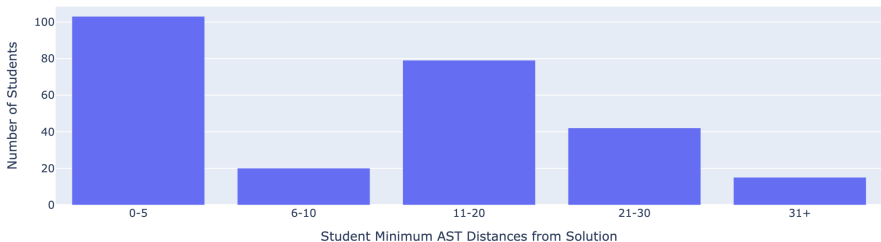


Figure 3: Histogram of students’ minimum AST distances from the solution for one exercise

time, showed that they were working authentically and naturally towards the solution. These students, in these exercises, were definitively *not* spinning.

Looking at the remaining students, researchers or other experienced programmers could often assert with confidence that a student was going in circles, making edits that were off-task, or in some other state of nonproductive perseverance.

From these observations we designed a simple algorithm that to categorize sessions as containing spinning or not by detecting oscillation within the distance score. This algorithm will be the core of the data labeling procedure we are developing as we move towards a generalizable machine learning model in Phase II.

7 Discussion

As we move into Phase II of the project, we are just starting to answer the research questions for Phase I. For RQ 1.1, we have a general set of features that we are currently testing for their power in predicting spinning. These tests include running component analyses for predictive power, and comparing predictions against human assessment. We know that RQ 1.2 is true: the likelihood of spinning can be assessed by a human, but only one student, exercise, or session at a time, and only with confidence if the code states show clear intent or lack of direction. There is a gray area that remains, which we hope to develop improved protocols to make clear. We have a simple, but working algorithm to detect spinning with reasonable confidence, making RQ 1.3 true: detection of spinning can be automated.

We currently have a wealth of near-keystroke-resolution student behavior data, collected from learners across the globe. That data has been successfully broken into comparable sessions, and has shown us that we can see progress

and failure to progress towards a solution when that solution is known. We can use that purely quantitative metric of progress to algorithmically label a session as spinning or not. This means we have all the components to create a machine learning training data set: the inputs (extracted features), and outputs (spinning detected). The pieces are not yet perfect, but we are making steady progress towards a generalizable model to assess spinning in real-time even when there is not a single canonical solution. We have experimented with a trivial-approach machine learning model, which didn't work, but demonstrated that we needed tremendously more data, that we have since gotten, and processed into a rich dataset.

Soon we will have a purpose-built neural network to detect the spinning state in arbitrary coding behavior, and we look forward to sharing those results with the community.

8 Acknowledgments

The authors wish to thank the student researchers who contributed so much to this project (in alphabetical order): Molly Hans, Alyssa Hogan, Samantha Marasca, Samantha Schumacher, and Nikholas Srnka. Thank you to the staff at Codio, and Elise Deitrick. Thanks to the Emmanuel College Faculty Development Committee for funding the research students time to work on this project.

References

- [1] Alice Armstrong. Successful first-year experience for at-risk students. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 45–50, 2017.
- [2] Ryan Shaun Baker, Albert T Corbett, Kenneth R Koedinger, and Angela Z Wagner. Off-task behavior in the cognitive tutor classroom: when students game the system. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390. ACM, 2004.
- [3] Matthew Berland, Taylor Martin, Tom Benton, Carmen Petrick Smith, and Don Davis. Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, 22(4):564–599, 2013.
- [4] Paulo Blikstein, Marcelo Worsley, Chris Piech, Mehran Sahami, Steven Cooper, and Daphne Koller. Programming pluralism: Using learning ana-

- lytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, 23(4):561–599, 2014.
- [5] Mohit Chandarana and Elise Deitrick. Challenges of scaling programming-based behavioral metrics. In *Proceedings of the Ninth ACM Conference on Learning @ Scale*, L@S '22, page 304–308, New York, NY, USA, 2022. Association for Computing Machinery.
 - [6] Mohit Chandarana and Elise Deitrick. Exploring error state in "time-on-task" calculations at scale. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2023, page 1336, New York, NY, USA, 2023. Association for Computing Machinery.
 - [7] Mohit Chandarana and Elise Deitrick. Most difficult errors for students to resolve across languages. In *Proceedings of the ACM Conference on Global Computing Education Vol 2*, CompEd 2023, page 192, New York, NY, USA, 2023. Association for Computing Machinery.
 - [8] John Edwards, Kaden Hart, and Christopher Warren. A practical model of student engagement while programming. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, SIGCSE 2022, page 558–564, New York, NY, USA, 2022. Association for Computing Machinery.
 - [9] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th koli calling international conference on computing education research*, pages 20–29, 2017.
 - [10] Hamza Errahmouni Barkam, Max Wang, Barbara Martinez Neda, and Sergio Gago-Masague. Testing machine learning models to identify computer science students at high-risk of probation. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, pages 1161–1161, 2022.
 - [11] Nickolas JG Falkner and Katrina E Falkner. A fast measure for identifying at-risk students in computer science. In *Proceedings of the ninth annual international conference on International computing education research*, pages 55–62, 2012.
 - [12] Ron Goldman, Scott Schaefer, and Tao Ju. Turtle geometry in computer graphics and computer-aided design. *Computer-Aided Design*, 36(14):1471–1482, 2004.

- [13] Heikki Hyyrö. Explaining and extending the bit-parallel approximate string matching algorithm of myers. Technical report, Citeseer, 2001.
- [14] Bo Pei and Wanli Xing. An interpretable pipeline for identifying at-risk students. *Journal of Educational Computing Research*, 60(2):380–405, 2022.
- [15] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 153–160, New York, NY, USA, 2012. ACM.
- [16] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

Teaching Algorithmic Bias*

Kevin McCullen
Computer Science Department
SUNY Plattsburgh
Plattsburgh, NY 12901
kmccu006@plattsburgh.edu

Abstract

Algorithmic Bias is the result of algorithms that have systematic errors that produce biased (unfair or discriminatory) results. Understanding Algorithmic Bias is of vital importance to both Computer Science majors and non-majors. Computer Science majors are poised to become the developers of systems that can be susceptible to bias, while non-majors (as well as majors) live in a world where many of their day to day interactions are mediated by machine learning applications that can exhibit bias. Aspects of our lives that are impacted by these algorithms include social media, marketing campaigns, government interactions, credit ratings, college admissions, and many more. Understanding the causes of algorithmic bias and the preventative measures necessary to produce systems that are equitable and fair, is an important ethical issue for future programming professionals.

1 Overview

Why teach algorithmic bias as the topic of a course? How is algorithmic bias important?

The ACM Curricula Recommendations [1] places a great deal of emphasis on ethics, and newer versions include specific recommendations with respect to Ethics and AI. "AI-SEP: Applications and Societal Impact" includes "Algorithmic Bias", "Evaluation Bias", and the assessment of the "societal implications"

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

of a selected application. One suggested learning outcome is "Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability."

Beyond the ACM's recommendations, the importance of this topic cannot be understated. In a world where the majority of interactions with companies are online (60% by 2020 [26]), how we interact with algorithms on a daily basis is vitally important. Are we treated equitably?

Governments are increasingly offering online services. As long ago as 2010, 81% of internet users were accessing government services online [8]. Are we treated equitably in those interactions?

Facial recognition technology is used in law enforcement [4], where it has been called "arguably one of the most controversial applications of machine learning", to exclude unwanted persons from malls [29], and for tasks as basic and common as building access [3].

Corporate and government databases contain massive amounts of data about individuals. As described by Roger McNamee, an early Facebook investor, companies like Google and Facebook use our data to build "data voodoo dolls" in order to make predictions about what we will like and what content will keep us engaged. These systems also carry an air of authority, because of assumptions that computers are impartial.

"Users believe Google and Facebook represent reality, but they are more like fun-house mirrors. Google's ability to deliver results in milliseconds provides an illusion of authority that users have misinterpreted. They confuse speed and comprehensiveness with accuracy, not realizing that Google results may be influenced by data voodoo dolls and advertiser needs, as well as search engine optimization and manipulation. Users mistakenly believe their ability to get an answer to any question means they themselves are now experts, no longer dependent on people who actually know what they are talking about." [19]

In *Algorithmic Authority: The Case of Bitcoin* [16], the authors define Algorithm Authority as "the trust in algorithms to direct human action and to verify information, in place of trusting or preferring human authority."

In explaining the significance of this issue to students, we can emphasize the ubiquitous nature of algorithm-mediated interactions, the extensive data about individuals used by those algorithms, and the default assumption that a computer is impartial, to make the case that this is an important topic for them to understand.

2 Setting the stage for students

First we need to define bias, algorithms, and algorithmic bias. When teaching a topic, it's critical to identify the subject being discussed. Ironically, students studying algorithmic bias may come with a "preconceived opinion" (one definition of bias) [11] of the meaning of that term. In a computer science context, algorithms may require no definition (although reminders are always helpful). But in a general education context, defining algorithms and engaging in some sort of exercise may be useful. There are a number of simple "algorithms are a set of instructions" exercises that can be used to demystify a computer science term for a general audience. Two that the author has found useful in the CS 0 context are sorting playing cards, and building Legos.

The card sorting exercise involves giving small groups of students a set of instructions and a set of playing cards. The students are instructed to shuffle the cards, then sort them by suit and number using their instructions. There are three different sets of instructions. One describes a radix sort by suit and number ($O(n)$). One describes bubble sort ($O(n^2)$). The third describes quicksort ($O(n \ln n)$). This introduces the concept of an algorithm being a set of instructions to accomplish a task, and that solving a task can be done with different degrees of complexity and time required.

The Lego exercise gives each small group of students a small set of Lego pieces. Each group gets the same pieces, and they are told to build something, anything. It's helpful to include different colors and some odd pieces like minifigs, or heads, wheels, roof parts, a small animal, etc. The group is to write down detailed instructions to build their object, and submit a picture. The next class session, shuffle the instructions and have groups use the instructions to try to replicate another group's original object. Then use the pictures to compare the results with the intended results.

Both of these short exercises are engaging and help clarify the meaning of the word algorithm.

Bias is a concept that is both clearly understood and simultaneously misunderstood. Virtually everyone believes that they know what "bias" or "biased" means. But in order to study the subject impartially, we need to establish a common definition.

Merriam-Webster defines bias as:

- (a) an inclination of temperament or outlook especially : a personal and sometimes unreasoned judgment : prejudice (b) an instance of such prejudice (c) bent, tendency (d) (1) deviation of the expected value of a statistical estimate from the quantity it estimates (2) systematic error introduced into sampling or testing by selecting or encouraging one outcome or answer over others [20]

Oxford Languages define bias as (def 3.c):

Tendency to favour or dislike a person or thing, especially as a result of a preconceived opinion; partiality, prejudice. Also: an instance of this; any preference or attitude that affects outlook or behaviour, esp. by inhibiting impartial consideration or judgement [11].

Can the results of an algorithm be mostly correct, but biased? If an algorithm is correct for a dominant selection of inputs, but incorrect for a small subgroup, is that a bug or is that bias? Over the range 1 to 500,000, approximately 8.3% of the numbers are prime [18]. In the United States, Census data indicates that approximately 8.2% of people do not have health care coverage [13]. Is an algorithm that produces incorrect results on prime numbers over that range biased? By definition **d2** from Merriam-Webster it might be. If it is incorrect in a systematic way, it could be said to be biased. Can we say that an algorithm that produces incorrect results for persons without health coverage has a strong indication of being biased? Again, looking at Merriam-Webster, there is an anthropomorphized aspect to definition **a**. We implicitly assume that bias is applied to persons, not numerical outcomes. In machine learning numerical outcomes are applied to persons. A numerical error (definitions **d1** and **d2**) becomes a bias (definition **a**).

3 What is Algorithmic Bias?

3.1 Sources of Algorithmic Bias

Algorithmic Bias can infect virtually any system, particularly those that are trained on data sets. The data sets themselves provide a source of bias. *Algorithmic Bias in autonomous systems* [10]. describes a taxonomy of sources of algorithmic bias. The authors identify "training set bias", "algorithmic force bias", "algorithmic processing bias", "transfer context bias", and "interpretation bias". In a general education seminar context, we focused on "training set bias" and the so-called "streetlight effect" ¹. This explanation points out that data is sometimes used for training because it is the most available data, not the most complete or accurate. IBM researchers found limitations in training data for facial recognition due to the fact that "while females and males are balanced in the training data, they are stereotypical females and males from the *celebrity* population." [21] In an online discussion with our seminar, one of the authors pointed to "paparazzi" image datasets as a source of images that were available without copyright restrictions. In her book [6] Dr. Buolamwini

¹a reference to a joke about an inebriated person looking for lost car keys under a streetlight, instead of where they lost the keys, because "the light is better here"

describes in detail her difficulties in finding data for facial recognition training that provided a breadth of genders, ages, and skin tones; while meeting standards for ethical use.

3.2 Examples of Algorithmic Bias

Algorithmic Bias examples are diverse enough to provide something of interest for virtually any level or type of course.

One of the major problems in approaching this topic is how many topics to cover, and in what depth? The instructor needs to choose between in-depth coverage of several topics or a survey of a large number of topics. Offering the course for CS majors in the future, the author would focus heavily on the topics described below: "analog bias", facial recognition, and LLM bias. Both facial recognition and LLMs allow for experimental exploration by computer science students. In a general education seminar, as was taught in the past, a broad survey of many topics is typically more suited to the student population.

Dr. Buolamwini's work [6] is focused primarily on facial recognition, with extensive discussion (noted earlier) of her attempts to find good training data, as well as her interactions with some of the companies whose systems she studied.

Large Language Models (LLMs) are susceptible to algorithmic bias, with a great deal of ongoing research into the topic, such as [23].

Looking for a broad list of topics, Cathy O'Neil's *Weapons of Math Destruction* [24] describes a set of specific examples, some of which are likely to be of personal interest to students:

1. College Admissions
2. Advertising
3. The Criminal Justice system
4. Hiring decisions
5. Job evaluations
6. Credit reporting
7. Insurance
8. "Civic Life" (with a focus on politics and social media, elections, and voting)

3.3 Analog Algorithmic Bias

We can start the story pre-internet with a non-computerized, analog example: "Shirley Cards". Kodak's photo printing machines (which were common in drugstores across the country) were calibrated using these cards. The printed color cards showed color blocks and the picture of a blond female (Shirley Page, a Kodak employee)[17]. An example can be seen in [32]. This calibration process, combined with the limited dynamic range of Kodak film at that time resulted in photographs that reproduced darker skin tones very poorly. Changes were not made until Kodak customers (described in the [17] as a chocolate company and a furniture company) complained, leading Kodak to develop film with a broader dynamic range (Gold Max), and eventually a multi-racial "Shirley Card".

3.4 Algorithmic Bias in Facial Recognition

Dr. Buolamwini's work is described in her book [6], the paper that described her research work [7], her dissertation [5], and the documentary film describing her work [15]. Dr. Buolamwini's work is described in ways that are accessible at all levels of courses, from general education and seminar classes, to upper-division courses with substantial programming and algorithmic content.

Briefly described, Dr. Buolamwini became involved in facial recognition research when a project she was developing for her Master's degree work was unable to detect her face. Due to her dark skin tone, her face was invisible to the software she was using. That led her to try to understand the causes of the software's failure, and to studying commercially available facial recognition software, finding that while published accuracy rates were high, performance on subsets of ages, genders, and skin tones could be very poor.

3.5 Algorithmic Bias in Large Language Models

This course was last offered as Large Language Models (LLMs) were beginning to become a dominant part of the discussion of AI, but they were addressed lightly. Future offerings of the course must address LLMs and their ability to be biased or to perpetuate biases. Two important sources of that bias are training data and deliberate choices made by the model provider.

It is possible that the way in which websites interact with LLM web crawlers can lead to biased training data. Assertions have been made that LLM web crawlers are more likely to be trained on misinformation than on "reputable news websites" [27]. Reputable sites (as defined in the paper by the authors) used robots.txt to block at least one AI agent 60% of the time, compared with 9% of sites that contain misinformation. Sites that require a paid subscription

to access are less accessible to LLMs than free sites supported exclusively by advertising.

In addition, LLMs utilize “system prompts” that give the model instructions on how to answer prompts, providing guardrails and directions [28] [25]. The model owners control these system prompts, and can use them to steer the outputs of the model. News reports show an apparent pattern of steering behavior with Elon Musk and the Grok LLM embedded in X [30].

Hallucination in LLMs is a well known phenomenon, but are the outputs biased? The model’s output can be free from hallucinations (i.e. factually true), but slanted in one direction or another. Or the model can offer opinions that are biased, either in subtle ways or in more overt ways.

A for-profit provider of an LLM could use the model’s output to steer users to products or vendors that have paid for placement. This is similar to how Google’s results show sponsored results, but in a more conversational way, and perhaps more persuasive than simple placement of sponsored links. Given the fact that LLMs are increasingly being seen as companions [2], there are substantial ethical considerations to be examined.

4 Class Content

4.1 Class Outline

This course has been offered twice as an honors seminar. The outline in table 1 shows the topics covered in the second iteration. The section on Social Science research is required under the general education category that covered the course offering.

Two books were assigned, Cathy O’Neil’s book [24] and *Technically Wrong* by Sara Wachter-Boettcher [31].

During a shortened week, the class watched the Coded Bias movie [15]. We also held an online session with researchers involved in the field of algorithmic bias with IBM corporation.

4.2 Assessments and Activities

Students developed an infographic on the subject of "What is Bias?", these were printed and posted in the honors center. In small groups, students created posters that were displayed during a university-wide diversity event. We held a debate, with a topic crafted by the students (with input from the instructor). The debate topic was "Resolved that algorithmic bias should be mitigated through regulation and transparency". Students were allowed to use AI as a research tool for the debate, with citation necessary. Preparation notes and a final debrief were required submissions. The importance of the debate is that

Week	Content (may change)
Week 1	What are Algorithms and how do they figure in our lives?
Week 2	What is Bias?
Week 3	How do we do Social Science research?
Week 4	Understanding ML, AI, and LLMs
Week 5	Aspects of Algorithmic Bias (Hiring and Employment)
Week 6	Aspects of Algorithmic Bias (Criminal Justice and College Admissions)
Week 7	Aspects of Algorithmic Bias (Medicine and Credit)
Week 8	Working on posters
Week 9	Ways to fix the problem? (Codes of Ethics, etc)
Week 10	Debate preparation
Week 11	Half week: Debate on Regulating AI and Big Data
Week 12	Future Shock! Where do you think we're going, what are the prospects?
Week 13	Guest speakers
Week 14	Half week: "Coded Bias" movie
Week 15	Working on Final presentations and Final thoughts
Week 16	Final Exam: Presentations

Table 1: Class Outline

it provides the students with some agency, looking at potential solutions rather than just focusing on the impacts.

The guest speakers in both iterations were from IBM Research. The instructor is acquainted a number of researchers at IBM, and was able to reach out for speakers. One of the student assignments was to write "three good questions" for the speakers, prior to the Zoom sessions.

In the first iteration of the course, the speaker talked about IBM's efforts to understand bias in facial recognition [21] (linked to Dr. Buolamwini's research, although that wasn't apparent until after Dr. Buolamwini's book was published). The second iteration included a talk on "Red Teaming" LLMs, from researchers who participated in events that attempted to elicit inappropriate behavior from the models [22].

The student final projects were highly varied, but one very promising project involved using Face++ [12] to experiment with facial recognition. The student and the class were very engaged with the results. The tool attempted

to determine gender and age from photos, with objectively poor results.

In a future iteration, the author would add experimentation with LLMs and comparisons of results from multiple LLMs.

The course evaluations, which are an imperfect measure, were uniformly positive, with an overall average of 4.59 / 5.00 (response n=8 out of 11 in the second iteration). Students rated their increased knowledge at 4.5 and having their interest in the topic stimulated at 4.38.

4.3 Modifications for a Computer Science audience

Were the course to be offered to computer science students, the author would investigate using tools such as Face++, as well as exploring algorithmic bias experiments using Python. IBM makes an "AI Fairness" toolkit available in Python using Jupyter and in R [14]. The toolkit includes tutorials such as a "credit fairness" test utilizing credit data from Germany. The example provides a method for altering the division of the data into privileged and non-privileged classes based on age, enabling the user to see how the outcomes change.

5 Conclusions

Algorithmic Bias has significant impacts in a wide variety of contexts. Algorithmic authority leads people to assume that computer algorithms are independent and unbiased. However, the selection of training data can lead to algorithms that favor or disfavor groups of individuals. Large language models, a form of artificial intelligence that is becoming dominant in many areas of life, may be trained on faulty data, or programmed to behave in biased ways.

Students, both computer science majors and non-majors, need to have an understanding of the causes, implications, and potential remedies for Algorithmic Bias. At the non-major level, a one semester course can introduce the concepts of algorithms and tie them to how machine learning tools make important decisions about our lives. Alternatively, selected content on this topic may be used in meeting new AI literacy requirements [9]. For Computer Science majors, adding experimentation with machine learning tools can provide a deeper understanding of the causes of bias and the need for careful selection of training data.

The material that is included in this course could be utilized in several different ways:

- As a stand-alone ethics seminar (how it has been taught).
- Portions extracted and used as sections within a broader computing ethics course.

- Applied within a course in Artificial Intelligence, Agentic Computing, or Large Language Models.

There are advantages to each of these. As a stand-alone course, the breadth of impacts can be explored. Within an ethics course or an AI course, the breadth and depth of topics would not be covered, but they can be presented in-context as part of the overall ethical landscape, or in conjunction with an understanding of how the models are trained and applied.

References

- [1] *ACM Curriculum Recommendations: Computing Curricula 2023*. 2020. URL: <https://www.acm.org/education/curricula-recommendations> (visited on 02/04/2025).
- [2] Marta Andersson. “Companionship in code: AI’s role in the future of human connection”. In: *Humanities and Social Sciences Communications* 12.1 (July 2025), p. 1177. ISSN: 2662-9992. DOI: 10.1057/s41599-025-05536-x. URL: <https://doi.org/10.1057/s41599-025-05536-x>.
- [3] Ginia Bellafante. “The Landlord Wants Facial Recognition in Its Rent-Stabilized Buildings. Why?” In: *New York Times* (Mar. 2019). URL: <https://www.nytimes.com/2019/03/28/nyregion/rent-stabilized-buildings-facial-recognition.html> (visited on 12/05/2025).
- [4] C. Borch and J.P. Pardo-Guerra. *The Oxford Handbook of the Sociology of Machine Learning*. Oxford Handbooks. Oxford University Press, 2025. ISBN: 978-0-19-765360-9. URL: <https://books.google.com/books?id=5QBHEQAAQBAJ>.
- [5] Joy Buolamwini. “Facing the Coded Gaze with Evocative Audits and Algorithmic Audits”. PhD thesis. Massachusetts Institute of Technology, 2022. URL: <https://dspace.mit.edu/handle/1721.1/143396> (visited on 12/05/2025).
- [6] Joy Buolamwini. *Unmasking AI: My Mission to Protect What Is Human in a World of Machines*. Penguin Random House, 2024.
- [7] Joy Buolamwini and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification”. In: *Proceedings of Machine Learning Research*. Vol. 81. 2018, pp. 1–15.
- [8] Pew Research Center. *How Americans Interact with Government Online*. Apr. 2010. URL: <https://www.pewresearch.org/2010/04/27/how-americans-interact-with-government-online/> (visited on 12/05/2025).

- [9] *Chancellor King Announces Improvements to SUNY Undergraduate General Education Curriculum, Including New Focus on Civic Discourse*. Tech. rep. State University of New York, Jan. 2025. URL: <https://www.suny.edu/suny-news/press-releases/1-25/1-7-25/general-education.html> (visited on 02/01/2026).
- [10] David Danks and Alex John London. “Algorithmic bias in autonomous systems”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. Melbourne, Australia: AAAI Press, 2017, pp. 4691–4697. ISBN: 978-0-9992411-0-3.
- [11] Oxford English Dictionary. *Bias*. URL: https://www.oed.com/dictionary/bias_n?tab=factsheet#21580744 (visited on 12/06/2025).
- [12] *Face++*. URL: <https://www.faceplusplus.com/> (visited on 12/05/2025).
- [13] US Government. *US Census Data*. URL: <https://data.census.gov> (visited on 12/05/2025).
- [14] IBM. *AI Fairness 360*. URL: <https://github.com/Trusted-AI/AIF360> (visited on 12/05/2025).
- [15] Shalini Kantayya. *Coded Bias*. English. Documentary. 2020.
- [16] Caitlin Lustig and Bonnie Nardi. “Algorithmic Authority: The Case of Bitcoin”. In: *2015 48th Hawaii International Conference on System Sciences*. ISSN: 1530-1605. Jan. 2015, pp. 743–752. DOI: 10.1109/HICSS.2015.95.
- [17] Roman Mars. *Shirley Cards*. Nov. 2021. URL: <https://99percentinvisible.org/episode/shirley-cards/> (visited on 12/05/2025).
- [18] Mathematica. *Mathematica*.
- [19] Roger McNamee. *Zucked: Waking Up to the Facebook Catastrophe*. Penguin Random House, 2020.
- [20] Merriam-Webster. *Bias*. URL: <https://www.merriam-webster.com/dictionary/bias> (visited on 12/06/2025).
- [21] Vidya Muthukumar et al. “Understanding Unequal Gender Classification Accuracy from Face Images”. In: *CoRR* abs/1812.00099 (2018). arXiv: 1812.00099. URL: <http://arxiv.org/abs/1812.00099>.
- [22] Manish Nagireddy, Bernat Guillén Peguerols, and Ioana Baldini. “DARE to Diversify: DATA Driven and Diverse LLM REd Teaming”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’24. event-place: Barcelona, Spain. New York, NY, USA: Association for Computing Machinery, 2024, pp. 6420–6421. ISBN: 979-8-4007-0490-1. DOI: 10.1145/3637528.3671444. URL: <https://doi.org/10.1145/3637528.3671444>.

- [23] Roberto Navigli, Simone Conia, and Björn Ross. “Biases in Large Language Models: Origins, Inventory, and Discussion”. In: *J. Data and Information Quality* 15.2 (June 2023). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 1936-1955. DOI: 10.1145/3597307. URL: <https://doi.org/10.1145/3597307>.
- [24] Cathy O’Neil. *Weapons of Math Destructions: How Big Data Increases Inequality and Threatens Democracy*. Penguin Random House, 2017.
- [25] Sunil Ramlochan. *System Prompts on Large Language Models*. Mar. 2024. URL: <https://promptengineering.org/system-prompts-in-large-language-models/> (visited on 12/05/2025).
- [26] Salesforce. *A Year of Change: 10 Salesforce Research Stats To Sum Up 2020*. Dec. 2020. URL: <https://www.salesforce.com/news/stories/a-year-of-change-10-salesforce-research-stats-to-sum-up-2020/> (visited on 12/05/2025).
- [27] Nicolas Steinacker-Olszlyn, Devashish Gosain, and Ha Dao. *Is Misinformation More Open? A Study of robots.txt Gatekeeping on the Web*. Oct. 2025.
- [28] SUSE. *Guiding the AI Model with System Prompts*. Nov. 2025. URL: <https://documentation.suse.com/suse-ai/1.0/html/AI-system-prompts/index.html> (visited on 12/06/2025).
- [29] Sarah Thamer. *New AI security measure at Mall of America raises concerns for some*. July 2024. URL: <https://www.mprnews.org/story/2024/07/09/new-ai-security-measure-at-mall-of-america-raises-concerns-for-some> (visited on 12/05/2025).
- [30] Stuart Thompson et al. “How Elon Musk Is Remaking Grok in His Image”. In: *New York Times* (Sept. 2025). URL: <https://www.nytimes.com/2025/09/02/technology/elon-musk-grok-conservative-chatbot.html> (visited on 12/06/2025).
- [31] Sara Wachter-Boettcher. *Technically Wrong: Sexist Apps, Biased Algorithms, and Other Threats of Toxic Tech*. W.W. Norton and Company, 2018.
- [32] Maggie Wessling. *The "Shirley Card" Legacy: Artists Correcting for Photography’s Racial Bias*. Apr. 2023. URL: <https://www.nga.gov/stories/articles/shirley-card-legacy-artists-correcting-photographys-racial-bias> (visited on 12/05/2025).

Reverse Engineering Student Misconceptions*

Anthony Notaro, Geoffrey Moores, and Maria R. Ebling
Department of Electrical Engineering & Computer Science
United States Military Academy
West Point, NY 10996

{anthony.notaro,geoffrey.moores,maria.ebling}@westpoint.edu

Abstract

We analyzed the effectiveness of generative AI in diagnosing student misconceptions in an undergraduate Operating Systems class. We asked students to assess whether the feedback provided by an LLM correctly diagnosed the mistaken belief underlying their error. We tested three models: ChatGPT, Claude, and Gemini. Gemini was the most consistent in terms of perceived correctness. ChatGPT received the highest student ratings. In a qualitative assessment, the LLMs correctly diagnosed the students' misconceptions 42% of the time. In a quantitative assessment, 63.5% of the LLM responses were perceived as exactly or almost exactly correct. We conclude that LLMs can provide valuable formative feedback but are not yet ready to be the sole source of insights.

1 Introduction

Generative AI (GenAI) has the potential not only to explain high-level concepts, but also to identify student mistakes. This study focused on the question of whether LLMs can identify student misconceptions on topics typically taught in an undergraduate Operating Systems (OS) course: CPU scheduling and virtual memory page replacement policies. We evaluated three well-known LLMs: ChatGPT, Claude, and Gemini. We asked students to rate each LLM's explanation of their mistakes.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

If successful, our overall goal for this work is to incorporate it into GraySim, a simulation system that supports students learning CPU scheduling policies [1] and virtual memory page replacement algorithms [4]. Currently, the simulation system has hand-coded tests for common errors so that the simulator can offer students formative feedback on their mistakes. This approach is laborious and, consequently, limited in scope. If one or more LLMs provide high-quality formative feedback, we can leverage it to enhance the simulation’s ability to support student learning.

Ultimately, our results were mixed. We found that Gemini was the most consistent of the LLMs. ChatGPT, on the other hand, received the highest ratings when it was able to answer within the 90-second time limit. LLMs correctly diagnosed students’ misconception just 42% of the time, yet gave poor explanations in fewer than 6% of cases.

This paper is organized as follows: Section 2 discusses prior work. Section 3 presents the details of our study, including the problems given to students, the prompts given to the LLMs, and the mechanics of the study. Section 4 summarizes the ratings and classifications of student comments. Section 5 highlights the implications of these results, comparing the performance of each LLM. Section 6 discusses areas in which the study could be improved and gives suggestions for follow-on studies. Section 7 concludes the paper.

2 Related Work

Automatically generating formative feedback to help students improve their work has long been an area of study [6]. Formative feedback, which provides information that can significantly improve the student’s understanding and future work, can take a variety of forms. Keuning *et al.* [6] designate five significant categories of formative feedback: knowledge about task constraints, knowledge about concepts, knowledge about mistakes, knowledge about how to proceed, and knowledge about meta-cognition. Determining which feedback from any of these categories is most appropriate is, in general, a difficult problem. In most cases, automated tooling for program feedback has made use of test failures as the primary feedback mechanism, or pre-defined next steps and hints. We note that while these forms of feedback provide the students with what went wrong, or a likely next step, they do not attempt to understand where the student went wrong or identify their incorrect thinking.

The theory of conceptual change [11, 3] in education refers to the idea that students enter the classroom with pre-existing, incorrect ideas, called misconceptions, that conflict with the material being taught. These misconceptions must be identified and addressed to allow true understanding and learning. In essence, students must restructure their existing knowledge rather than simply

adding new information. Our work seeks to identify student misconceptions so that these misconceptions can be actively addressed. Using the vocabulary of conceptual change theory, we seek to identify “anomalies” in student thinking to enable the student to find “dissatisfaction” with their current understanding and to make an “accommodation” and restructure their understanding.

FLA-RE and GraySim generate formative feedback by hand-coding a system to do so. FLA-RE provides formative feedback to students learning regular expressions [10]. Similarly, GraySim allows students to test their understanding of both CPU scheduling and virtual memory page replacement policies [1, 4]. Both FLA-RE and GraySim offer students formative feedback, but required custom programming by the system’s developers. In contrast to this work, we envision an enhancement to a system, such as FLA-RE or GraySim, that leverages an LLM to provide feedback without the developers having to anticipate every possible student error.

The In-Context Teaching model [12] uses a Bayesian framework fed by student responses to example problems to infer a student’s internal model of a concept, such as fractional arithmetic. This framework offers a rigorous method to classify student misconceptions, rather than their mistakes, but it is computationally complex and requires significant student data.

Another approach is to leverage custom artificial intelligence engines to generate formative feedback. Hoq *et al.* [5] describe a subtree-based neural network to detect logical errors within student programs. Students in an entry-level computer science course had their programming solutions parsed into an abstract syntax tree that was subsequently fed into a neural network. If the student response was incorrect, the network classified their error into one of five categories. The purpose of the study was to provide direction for educators through classifying error types. Our study shares similarities and differences. Both studies make use of artificial intelligence based on a neural network. Hoq’s however, makes use of a custom model, whereas we utilize GenAI. Our study also differs in the way we classify our errors. Instead of using AI to find the *type* of error, we use it to provide a detailed explanation of the student’s thought process. Furthermore, we validated this explanation by surveying the student on whether the explanation was correct.

In the past decade, natural language processing tools have grown significantly more capable, offering more adaptive and general approaches to automated interpretation and feedback generation. Shi *et al.* [13] studied automating the process of student error detection through use of a vector-embedding neural network, which maps (student response) text inputs to mathematical vectors. These vectors capture some semantic meanings via their relative proximities in various dimensions. The study focused on an undergraduate CS course and classified student error types. The goal of this was to help instruc-

tors understand trends within their courses so they could adjust their lectures accordingly. The model found 13 consistent clusters of misconceptions. Similarly, Demirezen *et al.* [2] confirmed that expert annotations align well with vector embeddings in undergraduate physics. Both studies used vector embeddings to classify student data; in contrast, we use GenAI. Instead of focusing on classifying types of misconceptions into groups, we focus on the individual student and determining whether GenAI can accurately explain their misconception. Our goal was to reverse-engineer their thought process and determine how effective different models are in finding the why behind their mistakes.

Weijers *et al.* [14] used GenAI to study undergraduate physics students taking a standardized exam. After taking the exam, students had a conversation with GenAI. The students went through their incorrect responses, having GenAI give them an explanation of the concept in question and how they got it wrong. Students generally found these interactions helpful. Similarly, Liu and M’hiri evaluated the responses of a virtual TA based on OpenAI’s GPT-3.5 in an introductory programming course and found that the virtual TA still required human oversight [7]. More recently, a study from Nguyen *et al.* compared the feedback from instructors versus OpenAI’s GPT-4 and determined that GPT provided feedback comparable to that of instructors [8]. We take a similar approach as these studies did in evaluating an AI’s performance in providing feedback for undergraduate technical work. We focus specifically on the question of whether LLMs can accurately identify what misconceptions the students had by leveraging immediate student self-assessments and their direct evaluations of the LLM output.

3 Method

In this section, we outline the method used in our study. We begin by describing each of three problems presented to students. We then discuss the student experience of the study. Next, we discuss the prompts used to capture feedback on the students’ performance from each LLM (ChatGPT, Claude, and Gemini) on each of the three problems. Finally, we summarize the way we collected data from students enrolled in an undergraduate OS course.

3.1 Problems

We presented students with three problems: CPU scheduling using round-robin (RR), CPU scheduling using a multi-level feedback queue (MLFQ), and virtual memory page replacement with the Clock algorithm (CLOCK). In both CPU scheduling problems, students used the specified algorithm to fill in the process execution timeline given each process’s start and service times. Figure 1 shows the initial screen for the RR problem (MLFQ is similar).



Figure 1: Round robin problem with a time quanta of 2.

In the Clock problem, we presented students with a page request stream and three frames representing physical memory. We asked them to use the Clock algorithm to place each requested page into a frame. We also asked them to identify which page requests caused page faults, by ticking the box in the bottom row. Figure 2 shows the initial screen for this problem.

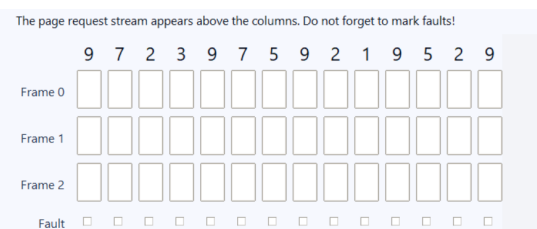


Figure 2: Page replacement problem with Clock algorithm.

3.2 LLMs

We used three general purpose LLMs for this study: ChatGPT 5¹, Gemini 2.5 Flash², and Claude Sonnet 4³. These models were chosen because they represent recent versions of each system and were expected to provide strong performance. Gemini 2.5 Flash represents a faster, but less powerful, version of Gemini 2.5. We selected this faster model based on timing issues with Gemini 2.5 during pre-trial tests (it was taking over ten times as long to return, on average, as Claude Sonnet 4).

We set a timeout of 90 seconds. If the LLM did not complete within this time, we allowed the student to continue. We set this timeout value based on twice the amount of time any of the LLMs had previously taken to respond to

¹<https://chatgpt.com/>

²<https://gemini.google.com/>

³<https://claude.ai/>

our queries the week before the study took place.

The order of presentation of the LLM responses to each student was randomly assigned, but remained the same for each problem presented to that student. Students were unaware of what LLM they were receiving a response from to eliminate bias.

3.3 Study

Each student began by downloading a Python script. The script presented each question in turn to the student, beginning with the round-robin CPU scheduling, continuing to the MLFQ problem, and finishing with the Clock problem. If the student answered the question correctly, the script moved on to a second question of the same type. If they also answered the second question correctly, the script moved them to the next question type. (Note that we only presented students with the second question if they responded correctly to the first question.) If the student answered a problem incorrectly, the script encoded their response and sent it to each of the three LLMs in the background. While the script awaited responses from the LLMs, it presented the student with their answer and the correct answer (Figure 3). It then asked them why they believed they had answered the problem incorrectly. The intent behind this interaction was to encourage students to reflect on the cause of their mistake before comparing it to any of the LLM responses.

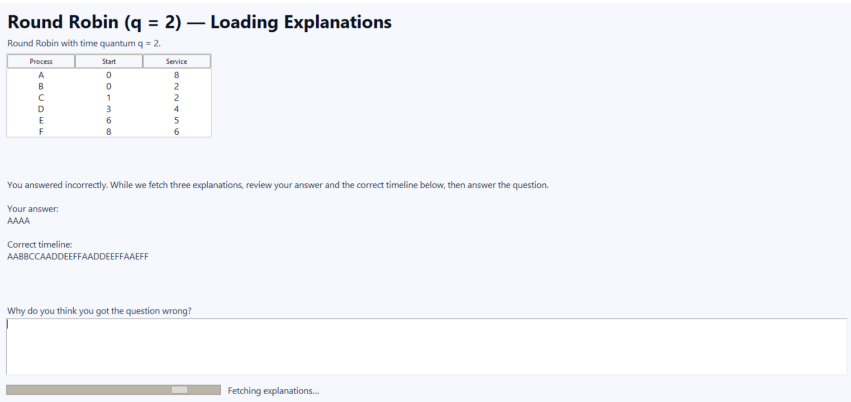


Figure 3: Loading screen after an incorrect response.

Once all LLMs finished (or timed out), the script showed each LLM’s response to the student in turn and asked them to rate the response’s accuracy on a scale of 1 (not close) to 7 (exactly right) and provide a justification for

this rating in a comment box (Figure 4). The student was not told which LLM was associated with each response. When the students completed the program, the script aggregated their responses into both a JSON file and a text file for analysis. They forwarded these files to one of the authors.

Your solution deviates from the approved one starting at time 2. At this point, process A completed its 2-unit quantum and should have been preempted, with process B (which arrived at t=0) scheduled next. Instead, you kept process A on the CPU for additional time units.

This error suggests a mistaken belief that a process continues to run until completion or indefinitely, rather than being strictly limited by the quantum in Round Robin scheduling. You appear to have misread the preemption mechanism that occurs when a process's quantum expires.

How close is this explanation to accurately identifying your misconception? (1=Not close, 7=Exactly right)

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

In your own words, what did this response get right or miss?

LLM Response 2

Feedback on your Round Robin (rr2) schedule:

Where your schedule deviates:

- At t=0 and t=0 you still run A. The approved schedule runs B at t=0-3, because A's 2-unit time slice ended at t=0 and A must be preempted.
- From t=4 onward your schedule is blank. The approved continuation is: C at t=4-5, A at 6-7, D at 8-9, E at 10-11, F at 12-13, A at 14-15, D at 16-17, E at 18-19, F at 20-21, A at 22-23, E at 24, F at 25-26.

Likely misconception:

- The slider's enforce the 2-unit quantum you appear to only reconsider the CPU on arrivals. In Round Robin, the quantum expiring triggers a context switch even if no new process arrives, and the running process is preempted behind any waiting processes.

How close is this explanation to accurately identifying your misconception? (1=Not close, 7=Exactly right)

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

In your own words, what did this response get right or miss?

Figure 4: LLM evaluation screen after an incorrect response.

3.4 Prompts

To create and refine our prompts, we began with a base prompt for each different problem type (MLFQ, RR, Clock). Each prompt generally introduces the problem, the correct solution, and the student solution, and then asks the LLM to explain the student's misconceptions and mistakes. We then used a sample incorrect student solution with each base prompt as input to multiple queries for each LLM platform (ChatGPT, Claude, and Gemini). After evaluating the batch of responses, we rated them between 1-5 each. Then we updated the base prompt in a direction we thought would cause the most improvement in responses, and repeated the process. At the end of the process, for each problem type, we selected the prompt with the highest average score across all responses from each LLM. We tailored the prompts to each problem type, but used the same prompt across all LLMs. The final prompts used for each of the problems can be found at [9].

One thing worth noting is how we encoded the student solution into the CPU scheduling prompts. We found that grouping the data into zipped tuples (correct process ID, student process ID, unit of time) resulted in faster and higher quality LLM responses. The Python script implemented this grouping for both the RR and MLFQ prompts.

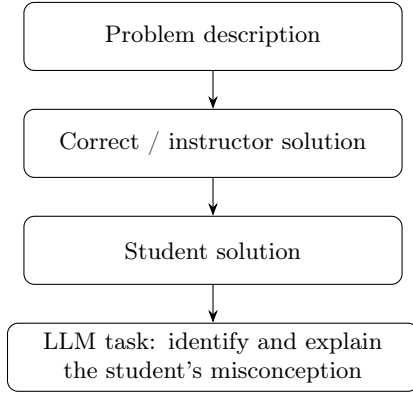


Figure 5: General structure of each prompt.

3.5 Student Trials

Students enrolled in our institution’s OS course were offered a 20 point bonus (in a 1000 point course) if they completed the exercises required for the study⁴, regardless of whether they permitted their responses to be used for analysis. Students completed the exercises during a regularly scheduled time for the class, but on a day when the class was not scheduled to meet or by appointment with one of the researchers. Upon completing the exercises, students emailed both the JSON and text files to one of the authors who was not teaching the course and indicated whether or not their responses could be used in the study. Of the 46 students enrolled in the class, 31 of them took this opportunity. All agreed to participate in the study. The full exercise took 49 minutes on average to complete with a minimum of 31 and a maximum of 97 minutes.

4 Results

In this section, we summarize the results. We begin by presenting the participants’ Likert ratings of the correctness of the LLMs’ explanations. We then present a summary of the qualitative feedback provided by the participants.

4.1 Correctness Ratings

Next, we report the ratings assigned by participants. Figure 6 shows the participants’ Likert ratings by problem type and model; we show each problem type in its own bar chart as well as a bar chart that combines the ratings

⁴Reviewed by our local IRB (Study #CA-2026-24)

across all three problem types. We omitted failures so the number of responses does not sum to the total number of incorrect student responses. In addition, the total number of responses does not necessarily sum to the total number of participants because some participants correctly answered both questions of the given problem type. Table 1 shows the participants’ Likert ratings by model and problem type for those LLMs that responded without failures. We highlight in bold each model’s overall performance across all problem types. The bootstrap mean of medians for ChatGPT is 6.5 with a 95% confidence interval of [6.0, 7.0]; similarly, Claude is 5.8 ([5.0, 6.0]) and Gemini is 6.0 ([6.0, 7.0]). No LLM dominates performance on all problems.

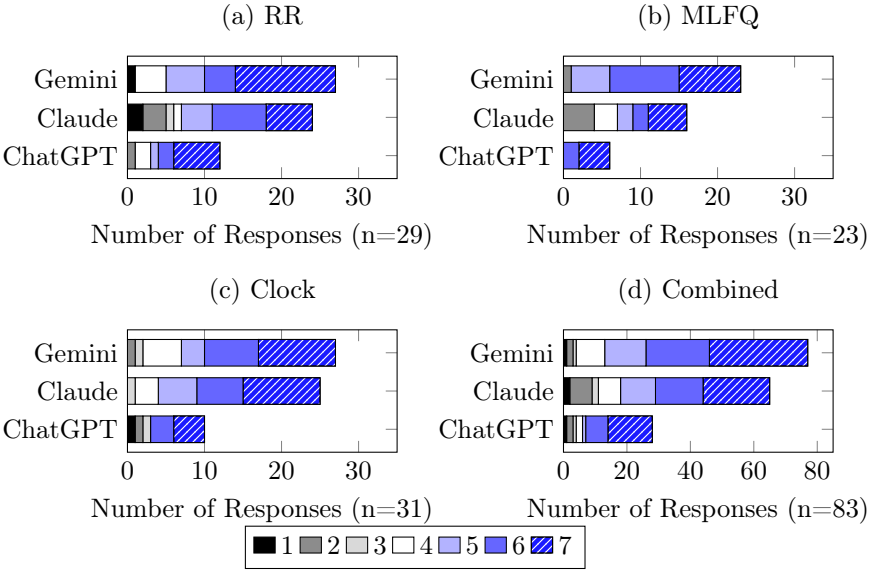


Figure 6: Ratings of Explanations (1=Not close; 7=Exactly right)

4.2 Qualitative Feedback

We classified the comments made by participants to the open-ended question that asked them to explain the correct and incorrect aspects of each LLM’s response. One researcher developed the classification scheme and coded each text response accordingly. That researcher shared the classification scheme with a second researcher who then marked each response independently. The two researchers compared their ratings and resolved any differences. Note that each participant’s text response could have more than one classification and students did not comment on all possible classification comments.

Out of 252 open-ended responses, 105 responses indicated that the LLMs’ responses were correct, 24 indicated that the LLM had both correct and incorrect aspects of its response, 16 indicated that the LLM was incorrect. Figure 7 shows these results. Note that not all students made a clear comment on correctness in their response to the open-ended question.

Many students reported that the LLM’s explanation was good (31), but students also reported problems with the LLM’s explanation. Students reported that, in some cases, an LLM’s explanation was confusing or difficult to understand (15), omitted an explanation (13), showed them *where* they were wrong, but did not explain *why* their approach was incorrect (12), was wordy (9), or ignored cascading errors (8). These results can be seen in Figure 8.

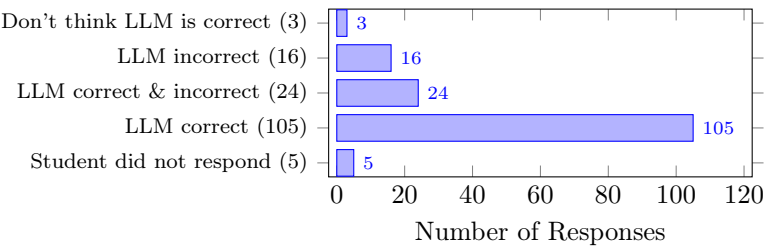


Figure 7: Correctness of LLM responses

LLM	Problem	1	2	3	4	5	6	7	Median	Failures	<i>n</i>
ChatGPT	RR	0	1	0	2	1	2	6	6.5	17	12
	MLFQ	0	0	0	0	0	2	4	7	17	6
	Clock	1	1	1	0	0	3	4	6	21	10
	Total	1	2	1	2	1	7	14	6.5	55	28
Claude	RR	2	3	1	1	4	7	6	6	5	24
	MLFQ	0	4	0	3	2	2	5	5	7	16
	Clock	0	0	1	3	5	6	10	6	6	25
	Total	2	7	2	7	11	15	21	6	18	65
Gemini	RR	1	0	0	4	5	4	13	6	2	27
	MLFQ	0	1	0	0	5	9	8	6	0	23
	Clock	0	1	1	5	3	7	10	6	4	27
	Total	1	2	1	9	13	20	31	6	6	77

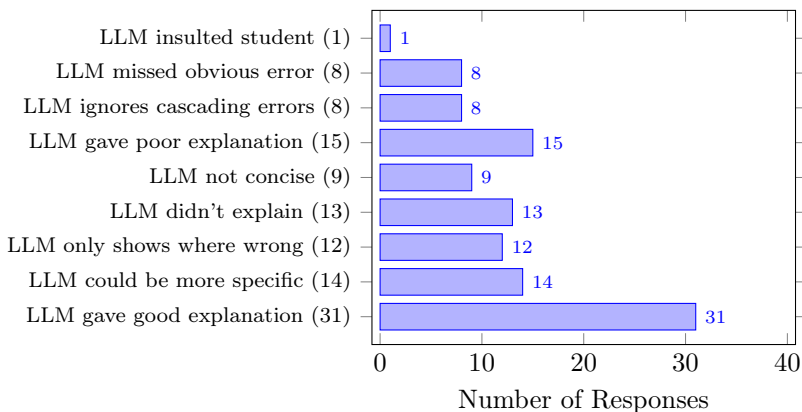


Figure 8: Quality of explanation.

5 Discussion

All three LLMs encountered failures, whether timeouts or server overload errors. Gemini encountered the fewest number of failures with just six timeouts across the entire study and no server errors. Claude experienced 18 server overloaded errors, but no timeouts. ChatGPT experienced 55 timeouts, but no server errors.

If we ignore cases where ChatGPT exceeded the time limit, students rated its explanations the highest. Most notably, its responses to the MLFQ errors received a median score of 7 (of 7) and its responses to the RR and Clock problems were 6.5 and 6 respectively. These ratings meet or exceed the ratings of the other two LLMs. However, the number of data points for ChatGPT is substantially lower than those for the other two LLMs and it is unclear if the trend would continue with additional data points.

The most common classification was that the LLM was correct in diagnosing the student's misconception, which appeared in 105 of the 252 responses. By this measure, the LLMs had only a 42% success rate. The next most common comments were that the LLM was partially correct in diagnosing their misconception or gave a good explanation, occurring on 10% and 12% of the comments respectively. If we consider the situation in which the LLM was at least partially correct, then the LLMs were rated as at least partially correct in 129 of the 252 responses, or 51%. In contrast, student responses to the Likert scale present a different picture with 63.5% of students reporting that the LLM responses were exactly (score of 7) or almost exactly correct (score of 6), with ChatGPT achieving 21/28, Claude achieving 36/65, and Gemini

achieving 51/77 for a total of 108/170. These two very different views of the data have a 21.5% discrepancy. Students may have focused on specific errors in their open-ended comments while expressing overall satisfaction in their Likert ratings. Further study would be required to understand this discrepancy.

Excluding comments about LLM correctness, the other responses to the open-ended question varied. Rarely students reported that an LLM missed an obvious error. Sometimes students reported an LLM's explanations as confusing, unorganized, or not specific. In summary, comments of that nature appeared approximately 15% of the time. Of note, students rarely reported that the LLMs missed cascading errors. This suggests that the LLMs have explanatory power beyond merely identifying an error and are able to avoid flagging subsequent errors that occurred due to the original mistake.

6 Limitations & Future Work

A limitation of this study is its generalizability. The problems given to each model were very specific to their subject and all were of similar format. This problem type differs substantially from other types of computer science problems such as programming or proofs. Future work could verify this result in other areas such as algorithm design, proofs for computer science theory, or complex system design.

A second limitation of this study was our approach to prompt engineering. We used a particular approach to generating a prompt that we then used across all three LLMs. Other approaches, including ones that are custom to specific LLMs or that apply different prompt engineering techniques, might produce better results. Future work could try customizing prompts to the specific LLMs or apply different prompt engineering techniques.

A third limitation of this study was our specific selection of LLMs. We focused on models that could respond within a time frame we believed students would be willing to tolerate. A different LLM selection or a different timeout value could produce very different results.

Another limitation of this study is the sample size. Our institution is a small, liberal arts college (SLAC) with fewer than 5000 students. There are fewer than 50 students enrolled in OS each year. This limitation could be addressed by repeating the study at a larger institution or otherwise making it available to a larger population.

The next limitation was a result of having the LLM timeout if the response took longer than 90 seconds. This timeout was done for reasons of psychological acceptability and was based on timings taken the week prior to the study. Unfortunately, we still experienced many timeouts (on ChatGPT and Gemini) or service-overloaded errors (on Claude), possibly due to fluctuations in server

usage and selecting (particularly for ChatGPT) a model that was too likely to time out. These timeouts greatly limited the number of ChatGPT and Claude responses that could be accurately rated by our study participants, but had less of an effect on Gemini’s results. Future research in this area should consider removing such timeouts, though that creates an issue about the length of the study and what happens in true failure conditions.

Kuening *et al.* report the results of a systematic review of automated feedback generation on programming exercises [6]. In this work, they classify the nature of the feedback generated, the techniques used to generate the feedback, as well as the quality and effectiveness of the feedback. Kuening’s systematic review pre-dates the release of GenAI so LLMs were not used in the generation of the feedback. Nevertheless, the classification system they used could offer valuable insights into the types of feedback being provided by LLMs and whether such feedback is substantively different from that provided by feedback generated through other mechanisms. Application of this work could be a fruitful area for further investigation.

7 Conclusions

We evaluated GenAI’s ability to diagnose student misconceptions in an OS class. Students were given round-robin, MLFQ, and Clock page replacement problems through a Python script. In the event they got a problem wrong, LLMs were prompted to explain the student’s thought process and why they were mistaken. Students then rated each response without knowledge of which LLM generated it.

After analysis, we found that Gemini was the most consistent of the LLMs. In cases where ChatGPT did not exceed the time limit, it received the highest ratings. As a whole, the LLMs performed best on MLFQ problems with ChatGPT receiving a median rating of 7 out of 7. Overall, the LLMs correctly diagnosed student misconceptions 42% of the time, and provided poor or confusing explanations in fewer than 20% of cases. We conclude that LLMs are not yet ready to serve as the sole source of formative feedback. However, when the custom feedback mechanism in a system such as GraySim yields no insights, LLM-generated feedback may offer useful supplementary feedback, provided students interpret that feedback with appropriate caution.

8 Acknowledgments

Outside of the clear uses of GenAI in this study, we also used GenAI to assist in developing the Python script, generating the TikZ bar charts, and editing the paper. The views and conclusions contained in this document are those of the

authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Military Academy, the U.S. Army, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

- [1] Sierra Zoe Bennett-Manke and Maria R. Ebling. “GraySim: An OS Scheduling Simulator”. In: *J. Comput. Sci. Coll.* 39.8 (Apr. 2024), pp. 55–69. ISSN: 1937-4771.
- [2] Mustafa Umut Demirezen, Ozgur Yilmaz, and Elif Ince. “New models developed for detection of misconceptions in physics with artificial intelligence”. In: *Neural Computing and Applications* 35.12 (2023), pp. 9225–9251.
- [3] R. Duit and D. F. Treagust. “Conceptual change: A powerful framework for improving science teaching and learning”. In: *International Journal of Science Education* 25.6 (2003), pp. 671–688. DOI: <https://doi.org/10.1080/09500690305016>.
- [4] Maria R. Ebling, Sierra Zoe Bennett-Manke, and Melinda Zhang. “GraySim: Simulations to Support Operating System Students”. In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2*. SIGSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, p. 1733. ISBN: 9798400705328. DOI: 10.1145/3641555.3705034. URL: <https://doi.org/10.1145/3641555.3705034>.
- [5] Muntasir Hoq et al. *Automated Identification of Logical Errors in Programs: Advancing Scalable Analysis of Student Misconceptions*. 2025. arXiv: 2505.10913 [cs.LG]. URL: <https://arxiv.org/abs/2505.10913>.
- [6] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. “Towards a Systematic Review of Automated Feedback Generation for Programming Exercises”. In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’16. (Extended version: <https://dspace.library.uu.nl/bitstream/handle/1874/346321/Extended.pdf>) Arequipa, Peru: Association for Computing Machinery, 2016, pp. 41–46. ISBN: 9781450342315. DOI: 10.1145/2899415.2899422. URL: <https://doi.org/10.1145/2899415.2899422>.

- [7] Mengqi Liu and Faten M'Hiri. "Beyond Traditional Teaching: Large Language Models as Simulated Teaching Assistants in Computer Science". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 743–749. ISBN: 9798400704239. DOI: 10.1145/3626252.3630789. URL: <https://doi.org/10.1145/3626252.3630789>.
- [8] Ha Nguyen, Nate Stott, and Vicki Allan. "Comparing Feedback from Large Language Models and Instructors: Teaching Computer Science at Scale". In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale*. L@S '24. Atlanta, GA, USA: Association for Computing Machinery, 2024, pp. 335–339. ISBN: 9798400706332. DOI: 10.1145/3657604.3664660. URL: <https://doi.org/10.1145/3657604.3664660>.
- [9] Anthony Notaro, Geoffrey Moores, and Maria R. Ebling. *LLM Prompts Details*. 2026. DOI: <https://doi.org/10.6084/m9.figshare.30820271>.
- [10] Olaperi Okuboyejo, Sigrid Ewert, and Ian Sanders. "Automatic Feedback Generation for the Learning of Regular Expressions". In: *ACM Trans. Comput. Educ.* 25.3 (Aug. 2025). DOI: 10.1145/3743682. URL: <https://doi.org/10.1145/3743682>.
- [11] George J. Posner et al. "Accommodation of a scientific conception: Toward a theory of conceptual change". In: *Science Education* 66.2 (Apr. 1982), pp. 211–227. DOI: <https://doi.org/10.1002/sce.3730660207>.
- [12] Alexis Ross and Jacob Andreas. *Toward In-Context Teaching: Adapting Examples to Students' Misconceptions*. 2024. arXiv: 2405.04495 [cs.CL]. URL: <https://arxiv.org/abs/2405.04495>.
- [13] Yang Shi et al. *Toward Semi-Automatic Misconception Discovery Using Code Embeddings*. Mar. 2021. DOI: 10.48550/arXiv.2103.04448.
- [14] Danique Weijers et al. "From Intuition to Understanding: Using AI Peers to Overcome Physics Misconceptions". In: *arXiv preprint arXiv:2504.00408* (2025). URL: <https://arxiv.org/abs/2504.00408>.

Integrating COIL into an Undergraduate Software Engineering Course: A Cross-Cultural Experience Report

Olivera Grujic
Computer Science
California State University, Stanislaus
Turlock, CA
ogrujic@csustan.edu

Abstract

This experience report describes a six-week Collaborative Online International Learning (COIL) module integrated into an undergraduate Software Engineering course, connecting students from the United States and Panama. The primary aim was to enhance student engagement and motivation through cross-cultural, project-based collaboration.

Students collaborated to complete software projects, including a stand-out game featuring campus-native animals symbolizing the partnership. Survey data from U.S. participants indicate increased motivation, improved project completion rates, and heightened interest in global collaboration.

Challenges such as time zone coordination and language barriers emerged but were effectively mitigated through the universal language of programming. The experience also fostered the development of soft skills, including enhanced communication and intercultural competence. This report offers insights into the pedagogical benefits, technological tools employed, and lessons learned.

*© 2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Background

The Collaborative Online International Learning (COIL) model, originally developed by the State University of New York (SUNY) in 2006, promotes project-based learning between classrooms in different countries through the use of digital tools and co-teaching strategies. It has been successfully implemented across disciplines to foster intercultural competence and global engagement [1].

This work was motivated by the following questions: Can a COIL-based international collaboration enhance student engagement and project quality in a software engineering course? What cultural, technical, and motivational outcomes emerge when students from different countries work on real-world software deliverables together? These questions guided the structure of the course and data collection.

While COIL has seen widespread adoption in the humanities, business, and communication disciplines, relatively few implementations have been documented in upper-division computing courses. This is a missed opportunity, particularly as modern video conferencing platforms like Zoom and asynchronous tools like GitHub and Discord make cross-border collaboration more feasible than ever [2], [3].

COIL enables instructors to pair their courses across institutions and countries, facilitating student engagement with authentic, international teamwork. The model has evolved over more than a decade and offers structured approaches to curricular alignment, co-teaching, and shared assessment. Research has shown that COIL projects enhance student motivation, intercultural awareness, and professional communication even when participants have never met in person or share a common first language [4]. In one study, students reported growth in confidence, reflection, and public speaking skills after participating in COIL-based assignments, even in asynchronous, English-medium contexts [4].

Typical COIL implementations last 4–8 weeks and involve joint activities, communication scaffolding, and shared deliverables. In computing, this format aligns naturally with project-based courses, particularly when student teams are asked to design, build, or deploy software artifacts. It also supports broader institutional efforts to internationalize the curriculum and foster global citizenship within disciplinary learning [5]. However, the integration of COIL into technical domains remains rare and under-reported in the literature. This paper contributes an experience report from a software engineering course that embedded COIL in a way that balanced institutional constraints with pedagogical goals.

2 COIL Implementation

In Spring 2025, a six-week COIL module was embedded into a Software Engineering course at a U.S.-based institution through a partnership with a faculty member from Panama. The project was developed through participation in a Faculty Learning Community (FLC), which provided guidance on best practices for intercultural course design, virtual engagement, and collaborative assessment. To ensure that our COIL-based approach effectively prepares students for real-world software engineering challenges, we aligned our course design with findings from Garousi et al. [6], who conducted a comprehensive review highlighting the necessity for educational programs to mirror industry practices.

Instructional responsibilities were shared between the two instructors. The Panamanian faculty member delivered three synchronous lectures focused on HTML, CSS, and Web Forms. The U.S.-based instructor followed with an interactive session on Servlets and JavaServer Pages (JSP), extending the technical concepts into backend development. In addition, the U.S.-based instructor reviewed all weekly student progress reports, provided cross-team coordination, and supported integration of frontend and backend deliverables. This instructional balance supported cohesion despite institutional differences in semester structure and course level.

Lectures were conducted synchronously via Zoom and recorded to accommodate schedule differences. Due to mismatched academic calendars and local timetables, the only overlapping time available for both institutions was a 75-minute session on Tuesdays. Lectures were delivered primarily in English, with Spanish used occasionally for clarification. COIL coordinators from both institutions also visited several joint sessions and class meetings to observe and support the collaboration. Students were organized into international teams and worked together using Discord for communication and GitHub for code sharing. Although the course did not formally adopt a specific process model, the six-week COIL module effectively operated as an Agile-style sprint with weekly deliverables, while documentation and integration checkpoints followed a more traditional waterfall structure, resulting in a practical hybrid approach.

All eight teams submitted working software by the end of the term. Panama students contributed frontend code only; U.S. teams were responsible for backend implementation and, in most cases, for integrating the final product. In several instances, integration could not be completed due to time constraints or mismatches in technical experience, so teams submitted separate front- and back-end deliverables. One standout project, a 2D tower defense game titled “Neques vs. Geese,” featured animal symbols representing each campus. The game demonstrated how creative freedom and intercultural collaboration can motivate students and produce technically complex, culturally meaningful

outcomes. The COIL implementation also highlighted real-world challenges in distributed collaboration: asynchronous teamwork, communication gaps, and coordination delays, which were addressed through intentional design, weekly reporting, and consistent instructional support.

3 Course and Context

The COIL module was implemented in Software Engineering, an upper-division course for undergraduate computer science majors at a U.S. institution that offers only a CS bachelor's program. The course introduces students to the full software development lifecycle with a focus on software design, documentation, teamwork, and deployment.

The curriculum emphasizes core backend development concepts. Technologies explicitly covered included Java, JSP, JDBC, and multithreading in Java. While the instructional focus was on backend systems, students were encouraged to explore other tools and platforms in their team projects. Notably, technologies such as RESTful APIs and Spring were not part of the required instruction.

The course was delivered in a hybrid format. Synchronous lectures were held via Zoom and recorded for asynchronous access. Students also had access to structured in-person support as needed. Students worked in stable teams throughout the semester and submitted a capstone-style software product as their final deliverable.

One important feature of the course was the weekly team report. This practice aligns with established research emphasizing the value of regular reflection in software engineering education. For instance, one study [7] found that weekly reflection surveys can effectively identify struggling teams early, enabling timely instructional interventions. Incorporating such reflective practices not only promotes self-awareness and metacognition but also enhances team collaboration and project outcomes. Each international team submitted a structured self-assessment that detailed goals, completed work, obstacles encountered, and next steps. These reports served both formative and summative purposes, allowing the instructor to monitor progress, offer asynchronous support, and intervene when team dynamics or technical coordination issues emerged.

The absence of a dedicated graphics course in the curriculum presented an opportunity. Students with an interest in game development were encouraged to pursue it as a final project. This flexibility helped students remain motivated and apply their technical knowledge in creative, culturally relevant ways, particularly within the framework of international collaboration.

A companion paper provides a comparative analysis of student experiences

Table 1: Summary of Student Projects

Project Title	Tech Stack	Key Features	Integrated?
Neques vs. Geese	Java, OOP, Custom GUI	Tower defense game, cultural mascots, grid logic	Yes
Social Chat App	Java, JSP, Web-Sockets	Real-time messaging	Yes
Medical Scheduler	JSP, MySQL, JDBC	Appointment system with role views	Yes
PDF to Audio	Java, Google TTS API	Accessibility document-to-audio tool	Partial
Finance Tracker	Java, JDBC	Expense tracking with export features	No
Do-Gether Task Manager	Servlets, Multi-threading	Task sharing with scheduling logic	Yes
Leaderboard System	Java + Go (Web-Sockets)	Realtime leaderboard and login	Partial
Second Chat App	Java, REST, JSP	Chat app with token auth	Yes

and learning across institutions, including professional, collaborative, and intercultural dimensions, as well as a more extensive discussion of instructional lessons and limitations [8].

4 Student Projects

Students collaborated asynchronously using GitHub for version control and Discord for team communication. Each team documented their progress through weekly reports and submitted a design document, presentation, and code repository at the end of the term. Most projects were jointly developed, with Panamanian students contributing front-end components using HTML and CSS, while U.S. students implemented server-side logic using Java, JSP, and JDBC. In several teams, these components were successfully integrated into a single functional application; in others, integration was attempted but ultimately submitted as coordinated front-end and back-end artifacts due to time constraints or differing levels of experience (see Table 1). Projects ranged from real-time chat platforms to task-sharing utilities, scheduling systems, accessibility tools, and an online leaderboard application.

All eight teams delivered working software by the end of the semester. Each team included approximately four students from Panama and five to six students from the U.S., spanning two course sections. One project stood out for its technical complexity, cultural creativity, and motivational impact: a 2D tower

defense game titled “Neques vs. Geese.” The game featured animals representing each campus: geese from the U.S. and neques from Panama, and reflected the students’ interpretation of cross-campus identity and playful competition. Built in Java with a custom GUI, grid logic, and object-oriented architecture, the game exemplified what students can produce when given autonomy within a supported design structure (see Figure 1).



Figure 1: Screenshot from the ‘Neques vs. Geese’ game project, showing tower placement and enemy movement.

5 Data and Findings

To evaluate the impact of the COIL module, both quantitative and qualitative data were collected through final exam surveys. The exam included seven COIL-related questions (see Table 2): six Likert-scale items (Q1–Q6) and one open-ended reflection (Q7). A total of 45 anonymized student responses were gathered across two sections. As the surveys are anonymized, and the activity was part of classroom evaluation, this study falls outside of the purview of the Institutional Review Board protocol.

The Likert-scale questions (Q1–Q6) asked students about their attitudes

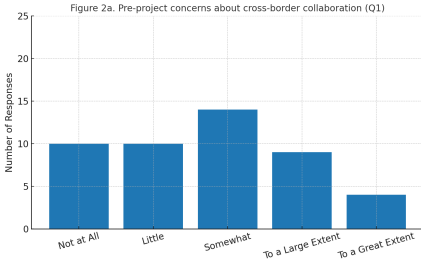


Figure 2a. Q1 – Pre-project concerns about cross-border collaboration.

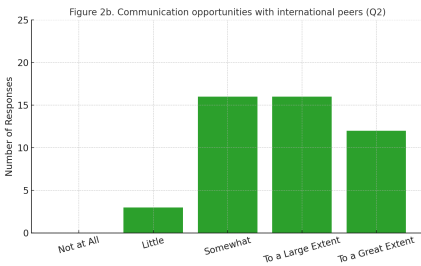


Figure 2b. Q2 – Communication opportunities with international peers.

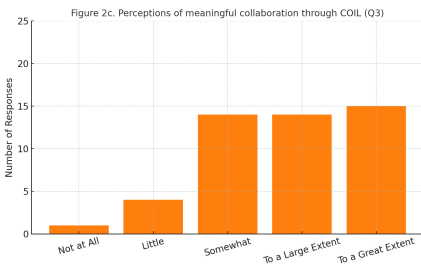


Figure 2c. Q3 – Perceptions of meaningful collaboration through COIL.

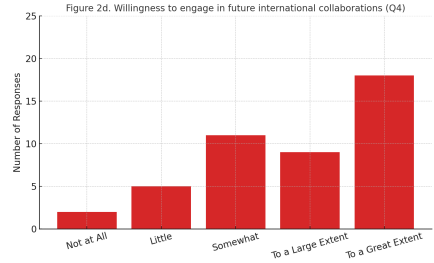


Figure 2d. Q4 – Willingness to engage in future international collaborations.



Figure 2e. Q5 – Interest in future courses with international integration.

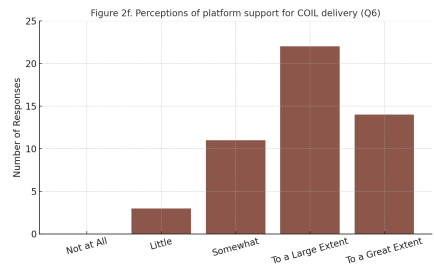


Figure 2f. Q6 – Perceptions of platform support for COIL delivery.

Figure 2: Survey response distributions for Q1–Q6 Likert-scale questions.

Table 2: COIL Survey Questions (Q1–Q7)

Q#	Survey Question
Q1	Before starting COIL project, I was concerned with my ability to collaborate with students from another country in a classroom setting.
Q2	The COIL project provided me with meaningful opportunity to communicate with students from another country.
Q3	The COIL project provided me with meaningful opportunity to collaborate on a project (assignment) with students from another country.
Q4	I am willing to participate in another international program or project.
Q5	I am willing to take another course that connects me with students from other countries.
Q6	The platforms (Canvas, Zoom, Panopto, etc.) used for the virtual classroom meetings were supportive.
Q7	COIL Reflection Journal: Please reflect (comment) on your experience with the COIL course. What did you think?

toward collaboration, communication, interest in future international experiences, and perceptions of the tools used. Each question used a five-point scale (e.g., “Not at All” to “To a Great Extent”). Survey response distributions for each item are visualized in Figure 2 (panels a–f).

Q1 asked whether students were initially concerned about collaborating across borders. Responses were mixed, with 10 students selecting “Not at All” and another 10 selecting “Little,” suggesting that nearly half of the students felt confident entering the international collaboration. However, 14 students responded “Somewhat,” and a combined 13 students selected “To a Large Extent” or “To a Great Extent,” indicating that apprehension was still present for a significant portion of the class (see Figure 2a).

In contrast, Q2 revealed a strong shift in tone. When asked whether the COIL project provided a meaningful opportunity to communicate with international peers, 16 students selected “To a Large Extent” and 12 selected “To a Great Extent.” Only 3 students expressed minimal benefit (“Little”), and no students selected “Not at All” (see Figure 2b). This indicates that despite initial concerns, students ultimately found the communication aspect of the collaboration highly successful.

Q3 followed with a similar trend: most students agreed that the collaboration itself was meaningful. 14 students each selected “Somewhat” and “To

a Large Extent,” and another 15 chose “To a Great Extent” with only five students selecting “Little” or “Not at All” combined (see Figure 2c). These results suggest that while students experienced some variation in how well the collaboration worked in practice, the vast majority saw value in the project experience and intercultural teamwork.

Q4 asked students whether they would be willing to engage in another international collaboration in the future. The results were strongly positive: 20 students selected “To a Great Extent” and 9 selected “To a Large Extent,” meaning nearly two-thirds of respondents were enthusiastic about repeating a similar experience. An additional 11 selected “Somewhat,” while only 7 students selected “Little” or “Not at All” (see Figure 2d). This suggests that the project not only met course objectives, but also sparked ongoing interest in global academic engagement.

Q5 further examined students’ openness to future international opportunities by asking whether they would take another course that connects them with students from other countries. Here again, responses were positive: More than 75% of students expressed high interest. 16 students selected “To a Great Extent,” 15 chose “To a Large Extent,” and 12 selected “Somewhat,” while only 4 students expressed minimal interest (see Figure 2e). These results reinforce the idea that structured COIL projects can promote long-term motivation to participate in global learning environments.

Q6 focused on the usability and support of digital platforms used during the project: Canvas, Zoom, Panopto, Discord, and GitHub among others. This item revealed the highest confidence overall, with 22 students selecting “To a Large Extent” and 14 selecting “To a Great Extent.” Only 3 students selected “Little,” and none responded “Not at All” (see Figure 2f). These results indicate that students felt well-supported by the chosen tools and platforms, which may have contributed to the high levels of satisfaction reported in Q2 through Q5.

When comparing Q1 with the remaining questions, a clear trajectory emerges: although 27 students reported at least some apprehension about cross-border collaboration at the outset (Q1), the subsequent items show that those concerns were largely overcome through the COIL experience. Responses to Q2 and Q3 demonstrate that students not only communicated effectively across cultures, but also found the collaboration itself to be meaningful. Their sustained interest in future international coursework (Q4–Q5) and confidence in the supporting infrastructure (Q6) provide compelling evidence that COIL positively impacted student attitudes toward global collaboration. The shift from concern to engagement suggests that thoughtfully designed international projects can meaningfully reduce barriers and foster enthusiasm for intercultural teamwork in computing education.

In addition to visualizing the Likert-scale distributions (Figures 2a–2f), we

Table 3: Descriptive statistics for Q1–Q6 survey items

Question	Mean	SD	Top 2 Box (%)
Q1	2.72	1.23	27.7
Q2	3.79	0.90	59.6
Q3	3.79	1.04	60.4
Q4	3.85	1.20	61.7
Q5	3.89	1.02	66.0
Q6	3.94	0.86	72.0

calculated descriptive statistics for each item to support comparative analysis (Table 3). Q1, which asked about initial apprehension toward international collaboration, had the lowest mean score (2.72) and the lowest top-2-box percentage (27.7%), reflecting mixed comfort levels at the outset. By contrast, Q2 and Q3, which addressed communication and collaboration experiences, showed higher mean scores of 3.79 and top-2-box percentages above 60%, suggesting that most students found the COIL structure both meaningful and manageable. The highest top-2-box score appeared in Q6 (72.0%), indicating strong confidence in the digital platforms used. These statistics reinforce the qualitative and visual findings: students gained confidence and engagement as the project progressed.

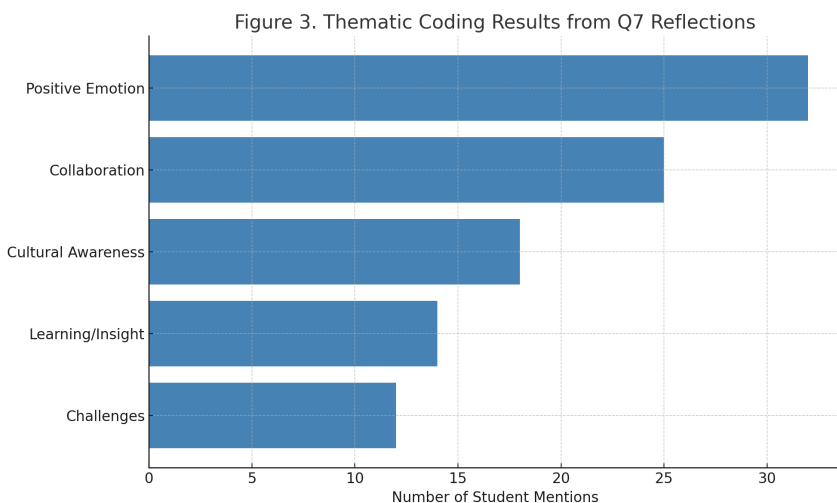


Figure 3: Thematic coding results from Q7 reflections. Bar chart shows number of students who referenced each theme in their open-ended feedback.

The open-ended Q7 prompted students to reflect on their COIL experience. All 45 responses were extracted and coded using thematic analysis. Results were charted for clarity, grouped thematically, summarized and visualized in Figure 3. Five dominant themes emerged: positive emotion (33 mentions), collaboration (25), culture (18), learning (14), and challenges (12). Notably, the most commonly cited theme, Positive Emotion, reflects a strong affective impact from participating in COIL, supporting claims that international collaboration enhances both technical and motivational engagement. This aligns with the survey findings from Q2 and Q3.

Sample reflections included: “I liked working with students from another country. It made the project feel more real,” and, “Even though we had issues merging the frontend, I still feel like we learned a lot about teamwork and patience.” These qualitative insights reinforced the trends seen in the quantitative results.

Together, the survey data and reflections suggest that this COIL collaboration met its learning goals. Students not only completed technical milestones but also gained experience in intercultural communication and asynchronous collaboration. The challenges they encountered did not detract from their overall positive evaluations.

6 Discussion

Integrating a COIL module into an upper-division software engineering course created valuable opportunities for technical, interpersonal, and cross-cultural growth. The experience demonstrated that student motivation and project quality can improve when learning is embedded in global, real-world collaboration despite institutional constraints and resource disparities.

The standout project was not only technically complex, but also thematically grounded in cultural representation. Featuring animals that symbolized each university, the game highlighted what students could achieve when given creative autonomy. It also exemplified the broader pedagogical goal of allowing students to channel their technical skills into personally and culturally meaningful work.

Across all eight teams, levels of collaboration and technical success varied. While some teams successfully integrated frontend and backend components from both institutions, others faced limitations in merging codebases due to differences in technical experience, scheduling mismatches, or time constraints. These real-world software engineering barriers provided authentic learning experiences. Students had to problem-solve across toolsets, communication styles, and institutional boundaries. Future implementations could address these integration challenges by scheduling dedicated joint debugging

sessions or leveraging shared continuous integration and continuous deployment (CI/CD) pipelines.

The data analysis confirmed strong student engagement. More than 75% of students expressed high interest in future international experiences. Over 30 responses mentioned the experience positively, and common themes included fun, challenge, collaboration, and cultural awareness. Several students commented on the reality of asynchronous team dynamics, noting that they felt better prepared for industry settings as a result. Others reported that it was their first time learning about Panama or engaging directly with international collaborators. These reflections suggest that the project succeeded not just in technical delivery, but in shifting students' perceptions of global teamwork and intercultural awareness.

Instructors also benefited from the COIL experience. The process of coordinating across institutions, reviewing weekly reports, and guiding students through real-world integration challenges offered insights into scalable collaboration. The shift in student attitudes, from moderate concern in Q1 to strong appreciation for communication and collaboration in Q2 and Q3, provides evidence of COIL's transformative potential. These results affirm the instructional value of embedding structured global teamwork into upper-division computing courses.

Several instructional lessons emerged from this COIL implementation, particularly regarding milestone structure, integration scaffolding, and coordination across academic calendars. These lessons, along with concrete recommendations for future implementations, are discussed in greater detail in a companion paper that analyzes both U.S. and Panama student perspectives [8].

While this study provides strong evidence of student engagement and perceived value, the findings reflect only the experiences of the U.S.-based participants. Due to differing academic calendars and the unexpected medical leave of the collaborating instructor, corresponding survey data from Panama students could not be collected during the study window. Future work will involve gathering and analyzing feedback from the Panama teams to capture a more complete picture of the cross-cultural collaboration.

7 Conclusion

The integration of a COIL module into an upper-division Software Engineering course enabled students to collaborate globally while applying core software development skills. Despite institutional and scheduling challenges, teams successfully delivered a range of web applications highlighting the motivational impact of creative autonomy. Additionally, open-ended reflections (Q7) revealed

a consistent emphasis on positive emotional experiences, suggesting that students valued the human connection as much as the technical accomplishment.

Students entered the project with moderate concerns about international collaboration, but survey results and reflections indicate that most found the experience highly meaningful. Their enthusiasm for future cross-border projects reinforces the value of structured, culturally embedded collaboration.

The project also provided instructors with insights into co-teaching, international coordination, and scalable global integration in technical coursework. These results affirm COIL's relevance for computing education and its capacity to support both technical learning and intercultural growth.

We hope this experience report can serve as a guide for other instructors seeking to implement COIL in computing courses, particularly those focused on software engineering or project-based learning. Among the key lessons learned were the importance of aligning calendars early, supporting integration with shared tools and checkpoints, and preparing students for cross-cultural collaboration in a technical context.

Limitations related to sample size, institutional context, and the absence of Panama survey data in this experience report are addressed more fully in the companion analysis [8].

References

- [1] Jon Rubin. Embedding collaborative online international learning (coil) across the curriculum. In *Internationalization of Higher Education Series*. Routledge, 2017.
- [2] Jon Rubin. The collaborative online international learning (coil) model: A decade of development. *International Education*, 41(2):1–10, 2017.
- [3] Adrian Kirkwood and Linda Price. Technology-enhanced learning and teaching in higher education: what is 'enhanced' and how do we know? a critical literature review. *Learning, Media and Technology*, 39(1):6–36, 2014.
- [4] Khaleda Al Mansoori. Research on the impact of coil projects in enhancing students' public speaking skills. *Journal of Teaching and Learning*, 12(2):45–58, 2024.
- [5] Hans De Wit and Betty Leask. Internationalization, the curriculum and the disciplines. *International Higher Education*, (83):10–12, 2015.
- [6] Vahid Garousi, Gursimran Giray, Eray Tüzün, Cagatay Çatal, and Michael Felderer. Closing the gap between software engineering education and industrial needs: A systematic literature review, 2018. arXiv:1812.01954.

- [7] Kathryn T. Stolee, Adit Shrivastava, Jordan Harriott, Tareq Alqaimi, and Robert Pienta. Identifying struggling teams in software engineering courses through weekly reflection surveys. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE '22)*, pages 1234–1240, 2022.
- [8] Olivera Grujic and Paulo Picota. Cross-cultural challenges and learning in a coil-based software engineering collaboration. *Journal of Computing Sciences in Colleges*, 41(6), 2026.

Integrating Smart Learning Content with Project-Based Introductory Programming Course at Community Colleges*

Mohammad Hassany², Can Kultur¹,
Rafaella Sampaio de Alencar², Kamil Akhuseyinoglu^{1,2},
Christopher Bogart¹, Jaromir Savelka¹, Peter Brusilovsky²
¹Carnegie Mellon University, Pittsburgh, PA 15213

{ckultur,kakhusey,jsavelka}@cs.cmu.edu

{cbogart}@cmu.edu

²University of Pittsburgh, Pittsburgh, PA 15260

{moh70,ras555,peterb}@pitt.edu

Abstract

This paper is a case-study on integrating Smart Learning Content (SLC) into a project-based undergraduate Python programming course delivered through an online learning platform to community college students. We showcase how practice-oriented SLC can be integrated in a project-based course and examine the connection between SLC usage and student learning outcomes. There are large publicly available repositories of SLC materials, yet their integration into a specific course and evaluation of its success are often not straightforward. By providing access to several SLC types, such as program construction examples, code animations, and Parson's puzzles, we aimed to bridge the gap between static conceptual reading and programming projects already present in the course. To evaluate the integration of SLC, we released the augmented course in the Spring 2025 semester to 252 students at 20 community colleges in the US and analyzed the interaction logs collected.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction and Motivation

Smart learning content (SLC), a broad set of interactive learning tools, offers an important resource for learners in introductory programming classes to develop a deeper understanding of complex coding concepts and competencies [5]. Popular SLC examples developed by the Computing education community include program visualizations [23], algorithm animations [20], interactive worked examples [15], Parson’s programming puzzles [17, 11], and coding exercises with self-assessment [6]. With increased adoption of learning management systems, interactive programming textbooks [7, 19], online practice systems [4, 9], and personalized learning systems [25, 14], multiple types of SLC can be combined, helping students to have greater and varied opportunities to master programming skills.

Although SLC in computing education can provide powerful tools that complement the guidance and feedback provided by instructors, and several studies in different computing courses demonstrated their learning benefits [8, 10, 21, 1], their practical impact has been much lower than expected due to its limited use outside their home sites [5]. This is unfortunate. If carefully designed to align with the specific needs and contexts of learners [12], and used in a *practice mode* to complement rather than replace classroom instruction, the growing volume of SLC could greatly benefit learners in under-resourced institutions, such as community colleges (CCs), which now serve a large number of learners from underrepresented communities in STEM [13]. These learners often have fewer opportunities for hands-on experiences to support their learning than their peers in 4-year degree programs [26, 27]. However, building early momentum in introductory programming courses is crucial for learners’ retention and success. Without developing a deep understanding of foundational concepts and skills, learners may encounter barriers to advance their career opportunities in computing and transfer to four-year degree programs [24].

Despite the demonstrated ability of SLC to help less-prepared learners, currently it is predominantly used in upper-level universities and colleges where the presence of faculty familiar with SLC facilitates its adoption. In this paper, we present our work on integration of four types of SLC into a regular learning process at several CCs. We start the paper by explaining our approach to integrating SLC in the CCs context, follow up with reporting the results of our semester-long study, and conclude with a summary of lessons learned and plans for future work.

2 The Context: SLC Integration into an Existing Course

We integrated the SLCs in an introductory programming course named “Practical Programming with Python (PPP)”. PPP has a project-based learning approach where each unit contains modules offering hands-on auto-graded projects. This course has been delivered through the Sail() platform and has been used by 950+ students in 40+ offerings at 30+ CCs since Fall 2022 after its launch with the support of a National Science Foundation grant addressing the critical need for accessible, high-quality computer science education in resource-limited educational settings.

The Sail() platform is a learning and research platform specifically designed to deliver project-based learning (PBL) experiences in computing education [3]. Students interact with real-world scenarios working on scaffolded tasks. They submit their solution for auto-grading to get actionable feedback in minutes, so that they can iteratively learn through coding-testing-refinement cycles. Auto-graded projects are at the center of the Sail() methodology, various pre-project modules (e.g., conceptual reading modules, quizzes, and primers) and post-project modules (e.g., reflections and online discussions) are used to support learning. While students interact with the platform, it collects comprehensive learning analytics such as navigation patterns, submission content with timings, assignment scores with detailed rubric breakdowns, and forum participation.

The student population is highly diverse, and many part-time students balance educational pursuits with full-time employment or childcare responsibilities [3]. In recognition of such unique challenges facing CC students, Sail() supports synchronous and asynchronous learning modalities. Institutions adapted the course delivery to their specific needs and student populations [3, 16]. Sail()’s modular structure lets institutions customize the provided curriculum by skipping some units or by delivering the course in two semesters to adjust their scope.

The PPP course is designed as an introductory programming experience for learners with minimal or no prior programming background, to build fundamental computational skills through hands-on applications. The overall goal is to equip learners with the competencies needed to write small- to medium-scale Python programs, preparing them for more advanced programming challenges and professional development opportunities in the technology sector. The course contains nine units, each having larger programming projects on which the learners can work individually. Although these large projects are subdivided into tasks for scaffolding purposes, they still require sustained effort that often extends over several days (and sometimes over a whole week), depending on the proficiency of the learner [18]. The first five units of the course curriculum encompass core programming concepts to provide students

with a comprehensive foundation in Python programming (see Table 1 for details). The following four units (not considered in this work) focus on software development and data processing in different contexts such as web-scraping, databases, and data analysis including visualization.

CS1 courses are generally considered challenging [2]. The inclusion of larger projects with real-life scenarios, while valuable, has the potential to make a course even more challenging. Promoting learners' self-efficacy and enabling their successful transition to real development environments is crucial in such settings. Each unit of PPP provides pre-project components (i) to develop knowledge and understanding through conceptual reading modules (with embedded MCQs and quizzes), and (ii) to improve the required project-specific skills through primer modules. Despite efforts such as embedding an interactive programming sandbox within the conceptual reading modules, the PPP course appeared to lack support for learners who may need more practice and feedback in the transition from conceptual understanding to working on hands-on projects. Unlike project-specific primers, SLCs seemed to be a good fit to fill this gap.

2.1 Selection of Smart Learning Content (SLC)

To explore the value of SLC in the context of PPP delivered at CCs, we selected four different SLC types to be integrated into the Sail() platform. We specifically focused on openly available and popular types of SLCs, which demonstrated their educational value in earlier studies [4, 10]. The selected SLC types can be easily accessed from SPLICE ACOS server (<https://acos.cs.vt.edu/>), a hosting platform that distributes reusable browser-based SLCs in an interoperable way [22].

Jsvee animations are Python code-tracing examples visualized at the expression level using the Jsvee library [21]. Each example presents a complete meaningful program that students can visualize step by step (Figure 1a) to see the execution of the program.

PCEX-examples, a type of PCEX (Program Construction EXample) activities [10], present worked code-construction examples. Each example consists of a problem statement and a complete solution explained line-by-line (Figure 1b). Using the “Explain the Program” button or clicking on the question mark buttons of each line, students see the explanations and can get more detailed explanations with the “Additional Details” button. On average, each PCEX-example has 10.93 lines annotated with explanations ($SD=7.05$, $range=2-35$).

PCEX-challenges [10] are simple code-completion problems. Like PCEX examples, they include a problem statement and complete solution, however one or more lines of code are removed. The students' goal is to complete the

Table 1: Distribution of the integrated SLCs

	PCEX example	PCEX challenge	Jsvee	JSParsons	JSParsons & Jsvee	Total
PPP Course: First Five Units						
1: Python Basics and Functions	8	9	7	8	2	34
2: Control Flow, Strings, Basic I/O	26	38	–	5	–	69
3: Python Data Structures	10	12	–	4	–	26
4: Working with Files and Modules	2	2	–	–	–	4
5: Object-Oriented Programming	5	5	–	2	1	13
Total	51	66	7	19	3	146

code by selecting the correct lines from a set of distractors. Students get immediate feedback and could try again or see a hint for incorrect lines. On average each PCEX-activity contains 1.20 PCEX-challenges (SD=0.45, range=1–3) and each challenge has 1.80 (SD=0.85, range=1–4) masked lines.

JSParsons problems [11] is an implementation of popular Parson’s puzzles [17]—a type of code construction problem in which students must assemble correct program solution (in this case, a Python program) by selecting, ordering, and indenting randomly shuffled lines of a correct solution. Students can check their answer by clicking the “Check” button or click “Reset” to try again; feedback on the student’s submission will also be provided.

The four selected SLC types offer a balance of worked examples and problems while supporting the acquisition of both program-tracing and program-construction skills. For each SLC type, SPLICE catalog offers a large number of activities developed by different research teams covering the key concepts of introductory Python classes. These activities were used in multiple courses beyond their home universities.

2.2 Integrating Smart Learning Content

To integrate the selected SLC activities with the “Practical Programming with Python (PPP),” we began by reviewing the course to identify topics to be covered and the target difficulty level. After several iterations of the review by the authors, we selected 146 activities distributed to the first five units of the PPP course as shown in Table 1.

4: Variables

5: --- Practice Time: Basics ---

6: --- Parsons Puzzles: Basics ---

7: Functions

8: Variable Scope and Stack Trace

9: --- Practice Time: Functions ---

10: --- Parsons Puzzles: Functions ---

11: Reading and Understanding a Program

12: Use of Functions and Variables to Improve Code Quality

13: --- Parsons Puzzles: Overview ---

14: References

Wrap Up

Conclusion

```
def calculate_area(width, height):
    print("Dimensions are:", width, height)
    result = width * height
    return height

def main():
    side1 = 12
    side2 = 10
    area = calculate_area(side1, side2)
    print("Area is:", area)
    print("Area is:", calculate_area(20, 7))

main()
```

Stack

Stack frame

side1 12

side2 10

calculate_area 12 10

Stack frame

main

Text console

Moved one step backwards.

Built-in Functions

Example: Input Manipulation

Construct a program that takes a user's inputs and multiplies the inputs. Then print the type of the result, convert the result to float and print its float absolute value.

Challenge Me!

1 # Step 1: Import math library

2 import math

3 # Step 2: Take the first input from the user

4 input_a = input("Enter the first integer:")

5 a = int(input_a)

6 # Step 3: Take the second input from the user

7 input_b = input("Enter the second integer:")

8 b = int(input_b)

9 # Step 3: Multiply the two variables

10 c = a*b

11 print(type(c))

Explanations

We need to read and process the input value that the user enters. To read the input value from the user, we need to use the input() function. This statement prompts the user to enter the first value, then reads the text entered by the user and places it in the variable input_a. The value that will be stored in variable input_a is a string.

Figure 1: SLCs within the Sail() conceptual reading modules – (a) A code animation (Jsvee) is shown on the top and (b) a worked example (PCEX-example) on the bottom.

The next decision was about how to integrate SLCs into the course content. After some trials and discussions, we decided to add SLCs as sets of activities that can be completed after specific chapters, akin to traditional textbook end-of-chapter exercises. Since SLCs may use different sets of concepts, or artifacts together, this approach has been found to be more feasible to align existing explanations and SLCs effectively. In this way, we explicitly encourage practicing with the help of specific naming conventions like “Practice time: ...” or “Parsons Puzzles: ...”.

Collecting learning logs from SLC was important. Since ACOS server supports “pluggable” logging approaches, we implemented an ACOS protocol, which uses the Sail() JavaScript logging library to collect SLC logs on the Sail() platform. The Sail() JS-logging library tagged the logs with contextual information about their role within Sail(), identifying the user, course, and location within the Sail() platform. A local instance of ACOS with the selected SLCs was hosted on the Sail() infrastructure to ensure privacy of students’ data and comply with the IRB requirements. From the consolidated Sail() and

196

SLC logs, we removed data from students who did not consent to research and anonymized the remaining records to preserve student anonymity.

3 Experiences and Results

3.1 Demographics and Students’ Activity

SLCs were inserted into the PPP course, used by seven CCs (referred to as CC-A to CC-G) in six states in the US. The total number of students enrolled in sections for which SLC activities were offered was 252. The demographic breakdown of students reflects the ratios typical of the domain (Figure 2a) and the schools’ regions (Figure 2b), i.e., roughly 2/3 male, typically in their 20’s and 30’s (Figure 2c) and with a strongly bimodal distribution of weekly work hours (Figure 2d).

The instructors used the course materials in different modalities: online, face-to-face, or hybrid (some online, some in person). The CC-A section (148 students) was taught entirely online. The CC-B section (45 students) was taught online, but with synchronized deadlines and zoom lectures by the instructor. Sections at CC-C, E, and G (30 students) were taught face-to-face, and CC-D and CC-F (29 students) were a mixture of online, face-to-face, and hybrid, depending on the course section.

In addition to general analysis of the learning data, we decided to investigate the CC-A section by itself. The number of students in CC-A (148) was greater than the total of all other sections of CC-B to CC-G. In addition to the varying student counts, this gave us a chance to interpret data from a specific modality perspective (online self-paced). Considering that these SLC activities were optional, while students have to complete challenging project assignments, we can expect a decrease of their use. So, understanding how these SLC activities are used would provide a good point to focus on to improve the learning experience of students in general and in specific modalities.

Figure 3 shows a retention curve for a CC-A section, showing the percentage of students attempting the main Sail() projects in each of the subsequent

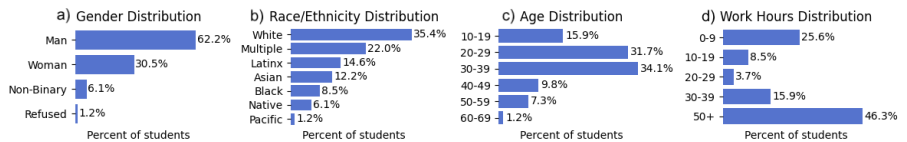


Figure 2: Distribution of characteristics for students who chose to answer the optional background survey (about half of total number).

Table 2: Summary of Log events by type.

Event type	Events	Students	Student-usage*
PCEX-examples			
Explanation views	2,006	59	6.30 ± 4.43 (6.00)
PCEX-challenges			
Result checks	3,230	60	3.15 ± 1.74 (3.00)
Hint requests	553	44	1.78 ± 0.98 (1.63)
Explanation views**	595	36	5.62 ± 3.19 (5.46)
Jsvees			
Tracing steps	20,901	49	63.97 ± 50.79 (53.14)
Line views	1,809	48	5.81 ± 1.96 (6.00)
JSParsons			
Result checks	2,896	68	5.50 ± 3.75 (4.39)
Tile moves	15,067	68	28.10 ± 15.65 (24.16)

*Student-usage reports Mean $\pm$ SD (Median) of a_i over students with $E_i \geq 1$; $a_i = \frac{1}{E_i} \sum_{e=1}^{E_i} c_{ie}$, where c_{ie} is the event count in item e and E_i is the number of items student i used. **Solving PCEX-challenges unlock their line-by-line explanations, like PCEX-examples.

range=1–73) used the integrated SLCs. Most (n=43) used a combination of all four SLC types, 23 students used only PCEX-examples, 12 used a combination of PCEX-examples/challenges and Parson’s puzzles, and the other 25 students used different combinations of the SLC types (Figure 4). A noticeable portion of students—46 students (18.25% of 252)—showed no active engagement in the PPP course; with few exceptions, they neither submitted tasks nor interacted with any SLCs. Table 2 provides descriptive statistics of important log events for the used SLC types.

Each SLC type was used by students in multiple sections. PCEX-examples were the most widely adopted, used in all 20 sections. PCEX-challenges and Jsvee animations were both used in 11 sections, while JSParsons problems were used in 13 sections; all of the SLCs were used by at least one student, except for 5 (7.58%) of the PCEX-challenges. See Table 3 for summary descriptive statistics of integrated SLC usage.

3.3 Student Performance and SLC Usage

The total scores for students who submitted at least one assignment were slightly different between non-users (n=91, mean=358.08, SD=239.28, Mdn=338.00) and SLC-users (n=102, mean=405.17, SD= 208.98, Mdn=375.50); however this difference is not statistically significant (p=0.146). Figure 5 shows

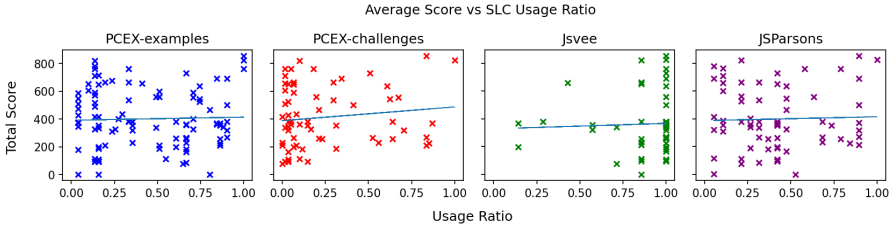


Figure 5: SLC Types’ Usage Ratio vs Students’ Total Score; plots include linear regression line.

the correlation of SLC types usage with the students’ total score (in all colleges). Across all SLC types, increased usage is associated with higher average scores; however, the correlations are weak and statistically not significant. A similar correlation can be observed between students’ average success rates for PCEX-challenges and their total scores ($r=0.0859$, $p=0.5140$), however, as seen in Figure 6 for JSParsons problems the correlation is moderate and statistically significant ($r=0.3982$, $p=0.0031$). No significant correlation has been observed between SLC use and student self-efficacy reported.

We examined the score differences among SLC and non-SLC users in one of the colleges with the highest enrollment (CC-A) to further reduce the dis-

Table 3: Summary of SLC usage by type.

Measure	Students	Mean $\pm$ SD (Median)
PCEX-examples used	103	22.00 $\pm$ 15.66 (18.00)
Lines viewed	59	25.46 $\pm$ 24.82 (20.00)
Time spent (mins)*	50	1:51 $\pm$ 2:41 (0:58)
PCEX-challenges used	65	14.83 $\pm$ 17.32 (6.00)
Attempts	65	2.64 $\pm$ 1.77 (2.38)
Time spent (mins)*	61	2:38 $\pm$ 2:03 (2:08)
Success rate**	60	49.48% $\pm$ 22.34% (50.00%)
Jsvees used	49	6.16 $\pm$ 1.55 (7.00)
Lines viewed	49	4.50 $\pm$ 1.20 (5.14)
JSParsons used	68	7.90 $\pm$ 5.06 (8.00)
Attempts	68	5.50 $\pm$ 3.75 (4.39)
Success rate**	68	45.25% $\pm$ 25.90% (50.13%)
Total SLCs used:	103	39.50 $\pm$ 35.05 (28.00)

*Only 5secs–30mins sessions are included. **Success rate includes all attempts up to and including the 1st correct one, with future attempts assumed to be exploratory.

Table 4: Comparisons of Sail() project scores, over units 1-3, for students who attempted a Sail() project (+Sail), and SLC’s effect on score. Analysis is limited to sections where these units overlapped the data collection period.

	Num. Students			Mean Score		
School	Total	+SLC	-SLC	+SLC	-SLC	p-value
CC-A	110	70	40	220.30	164.75	0.0011
CC-B	35	11	24	253.09	232.04	0.3347
CC-C	12	6	6	229.50	192.17	0.2873
CC-E	6	1	5	77.00	238.40	1.0000

crepancies between colleges. We included students who completed at least one task (n=119). Wilcoxon rank-sum (Mann–Whitney) test finds no statistically significant differences in average score (mean) between SLC users and non-SLC users (non-SLC: n=46; SLC: n=73; $z = -1.32$; $p = 0.186$). While SLC users have a slightly higher rank sum (suggesting somewhat higher scores), the evidence is not strong enough to conclude their average scores differ from non-SLC users. Table 4 shows the comparison for several colleges with comparable subsets of projects assigned.

4 Discussion

The correlation between SLC usage and improved performance on Sail() projects suggests that providing students with an opportunity to practice can meaningfully enhance their learning outcomes. However, the data also indicate the importance of sufficient engagement with SLC to obtain these benefits and the

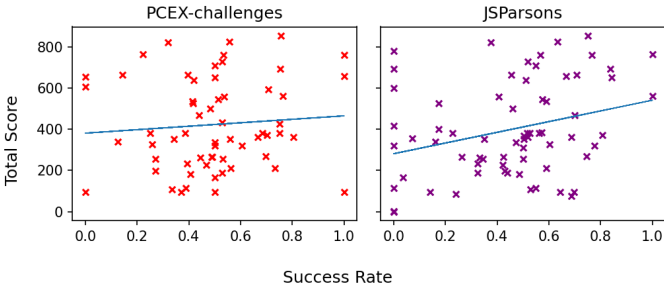


Figure 6: Students Total Score vs their Average Success Rate in PCEX-challenges and JSParsons problems; plots include linear regression line.

engagement logs revealed differences in SLC usage among the CCs. We are trying to explore several options to increase student engagement with SLC, such as progress indicators and other tools that support self-regulated learning [4]. Interestingly, CC-A offering the course remotely (self-paced) had higher active student ratios. We hypothesize that students used to self-paced courses might have developed more robust self-regulated learning skills. This in turn suggests that we may need to consider strategies on how to engage students whose skills are not that developed.

Based on our findings, we recommend that institutions integrate multiple types of SLC, as our data show that students benefit from diverse learning supports. Second, our analysis indicates that successful SLC deployment requires careful attention to institutional context. The variation in usage ratios across institutions suggests that one-size-fits-all approaches are unlikely to be effective. Third, the significant relationship between SLC usage and project performance may suggest that these tools should be integrated into the core curriculum rather than offered as optional supplements.

We need to note that our study relies on log data to determine SLC usage which may not fully capture the quality of students' engagements. Additionally, the correlation between SLC usage and performance does not establish causality, as higher-performing students may be more likely to use SLC. The institutional variation in our dataset poses difficulties in interpreting the results. Different institutions may have varying levels of technological infrastructure, student support systems, and pedagogical approaches that could influence SLC effectiveness.

The lessons learned from our integration efforts may help instructors, instructional designers, and developers in similar future attempts. When integrating SLCs from a library into existing curricula, people may naturally focus on SLC types, or they may have a tendency to start by matching the topics and learning objectives between two resources. However, it is equally important to first examine the underlying assumptions and approaches of both systems. Similarly to making minor content revisions, the need for code adjustments (like dynamical height adjustment) in both platforms can be signals of mismatching approaches or underlying assumptions. To avoid unexpected consequences, these signals should be interpreted and addressed early in the integration process.

5 Conclusions and Future Work

Our study demonstrates that SLC usage may correlate with improved student performance in programming education. Our findings suggest that successful SLC implementation requires institutional-specific strategies rather than uni-

versal approaches, and that diverse SLC types provide complementary learning benefits. Future work should explore causal relationships between SLC usage and performance and investigate strategies for engaging students with limited self-regulated learning skills.

Acknowledgments

This material is partially based upon work supported by the National Science Foundation under Grants No. 2111305 and No. 2213789, and by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W519TC-23-C-0045. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred.

References

- [1] Kamil Akhuseyinoglu et al. “A Study of Worked Examples for SQL Programming”. In: *2022 Annual Conference on Innovation and Technology in Computer Science Education*. 2022, pp. 82–88.
- [2] Jens Bennedsen and Michael E Caspersen. “Failure rates in introductory programming: 12 years later”. In: *ACM inroads* 10.2 (2019), pp. 30–36.
- [3] Christopher Bogart et al. “What Factors Influence Persistence in Project-based Programming Courses at Community Colleges?” In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. Portland, OR, USA: ACM, 2024, pp. 116–122. DOI: 10.1145/3626252.3630965.
- [4] Peter Brusilovsky et al. “An integrated practice system for learning programming in Python: design and evaluation”. In: *Research and Practice in Technology Enhanced Learning* 13.18 (2018), pp. 18.1–18.40.
- [5] Peter Brusilovsky et al. “Increasing Adoption of Smart Learning Content for Computer Science Education”. In: *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. 2014, pp. 31–57. ISBN: 978-1-4503-3406-8.
- [6] Stephen H. Edwards and Krishnan Panamalai Murali. “CodeWorkout: Short Programming Exercises with Built-in Data Collection”. In: *2017 Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE’17*. ACM Press, 2017, pp. 188–193. DOI: 10.1145/3059009.3059055.

- [7] Barbara Ericson et al. “An eBook for Teachers Learning CS Principles”. In: *ACM Inroads* 6.4 (2015), pp. 84–86.
- [8] Tommy Färnqvist et al. “Supporting Active Learning by Introducing an Interactive Teaching Tool in a Data Structures and Algorithms Course”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. Memphis, Tennessee, USA: ACM, 2016, pp. 663–668.
- [9] Adam M. Gaweda and Collin F. Lynch. “Student Practice Sessions Modeled as ICAP Activity Silos”. In: *14th International Conference on Educational Data Mining*. 2021.
- [10] Roya Hosseini et al. “Improving Engagement in Program Construction Examples for Learning Python Programming”. en. In: *International Journal of Artificial Intelligence in Education* 30.2 (June 2020), pp. 299–336. ISSN: 1560-4306. DOI: 10.1007/s40593-020-00197-0.
- [11] Petri Ihantola and Ville Karavirta. “Two-dimensional parson’s puzzles: The concept, tools, and first observations”. In: *Journal of Information Technology Education. Innovations in Practice* 10 (2011), p. 119.
- [12] Shanna Smith Jaggars, Nikki Edgecombe, and Georgia West Stacey. “What We Know about Online Course Outcomes. Research Overview.” In: *Community College Research Center, Columbia University* (2013).
- [13] Beethika Khan, Carol Robbins, and Abigail Okrent. “Science and engineering indicators 2020: The state of US science and engineering”. In: *Science & Engineering Indicators, NSB-2020-1* (2020).
- [14] Aleksandra Klačnjak-Milićević et al. “Enhancing E-Learning Systems with Personalized Recommendation Based on Collaborative Tagging Techniques”. In: *Applied Intelligence* 48.6 (June 2018), pp. 1519–1535.
- [15] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. “A Review of Worked Examples in Programming Activities”. In: *ACM Transactions on Computing Education* 23.1 (Dec. 2022), 13:1–13:35. DOI: 10.1145/3560266.
- [16] Huy A. Nguyen et al. “Examining the Trade-Offs Between Simplified and Realistic Coding Environments in an Introductory Python Programming Class”. In: *Proceedings of 19th European Conference on Technology Enhanced Learning, EC-TEL 2024, September 16–20, 2024*. Krams, Austria: Springer-Verlag, 2024, pp. 315–329.
- [17] Dale Parsons and Patricia Haden. “Parson’s programming puzzles: a fun and effective learning tool for first programming courses”. In: *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 2006, pp. 157–163.

- [18] Jaromir Savelka et al. “Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses”. In: *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*. ICER '23. Chicago, IL, USA: ACM, 2023, pp. 78–92. ISBN: 9781450399760.
- [19] Clifford Shaffer. “OpenDSA: An Interactive eTextbook for Computer Science Courses”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 2016, pp. 5–5.
- [20] Clifford A Shaffer, Matthew Cooper, and Stephen H Edwards. “Algorithm visualization: a report on the state of the field”. In: *Proceedings of the 38th SIGCSE technical symposium on Computer science education*. 2007, pp. 150–154.
- [21] Teemu Sirkiä. “Jsvee & Kelmu: Creating and tailoring program animations for computing education”. In: *Journal of Software: Evolution and Process* 30.2 (2018). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1924>.
- [22] Teemu Sirkiä and Lassi Haaranen. “Improving online learning activity interoperability with acos server”. In: *Software: Practice and Experience* 47.11 (2017), pp. 1657–1676.
- [23] Juha Sorva, Ville Karavirta, and Lauri Malmi. “A Review of Generic Program Visualization Systems for Introductory Programming Education”. In: *ACM Transactions on Computing Education* 13.4 (2013).
- [24] Michelle Van Noy and Matthew Zeidenberg. “Community college pathways to the STEM workforce: What are they, who follows them, and how?” In: *New Directions for Community Colleges* 2017.178 (2017), pp. 9–21.
- [25] Boban Vesin et al. “Adaptive Assessment and Content Recommendation in Online Programming Courses: On the Use of Elo-rating”. In: *ACM Transactions on Computing Education* 22.3 (2022), Article 33.
- [26] Xueli Wang. “Course-taking patterns of community college students beginning in STEM: Using data mining techniques to reveal viable STEM transfer pathways”. In: *Research in Higher Education* 57 (2016), pp. 544–569.
- [27] Yi Leaf Zhang, Maria Adamuti-Trache, and John Connolly. “From community college attendants to baccalaureate recipients: A planned behavior model for transfer students in STEM fields of study”. In: *The Journal of Higher Education* 90.3 (2019), pp. 373–401.

AI-Enhanced Guided-Inquiry Platform for Collaborative Learning in Introductory Programming*

Bo Kim¹ and James Skon²

¹Department of Computer Science
Southern New Hampshire University
Manchester, NH 03106

`b.kim@snhu.edu`

²Kenyon College
Gambier, OH 43022

`skonjp@kenyon.edu`

Abstract

This paper presents a digital learning framework, coLearn-AI (hereafter referred to as the system), that combines guided inquiry pedagogy with intelligent, adaptive feedback mechanisms to support collaborative learning in introductory programming courses. The system enables instructors to author structured inquiry activities, manage student groups, and provide embedded AI guidance that scaffolds learning in real time. Students work in groups and engage with progressively complex problems through exploration, collaboration, and reflection. The system integrates synchronous collaborative workspaces, turn-taking mechanisms, and analytics dashboards that help instructors monitor student progress at scale. A key innovation is the integration of AI-driven formative feedback (OpenAI API is used), where instructors embed AI guidance directives within activity files. These directives, visible only to instructors, enable the AI module to provide context-aware feedback that aligns with

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

pedagogical goals. By bridging structured human facilitation with adaptive, intelligent tutoring capabilities, the system promotes engagement, accountability, and deeper conceptual understanding. The platform has been successfully deployed in multiple university courses for in-class work since September 2025; instructors monitor progress through dashboards. We report implementation status and early classroom use. This paper describes the pedagogical foundations, system architecture, activity model, AI feedback pipeline, and group-learning mechanisms and outlines directions for formal efficiency evaluation and refinement.

1 Introduction

Introductory programming courses often face challenges, including uneven engagement, limited opportunities for individualized guidance, and difficulty sustaining collaborative learning at scale [3]. These challenges are amplified by the wide diversity of students' prior experiences, technical backgrounds, and cultural or social contexts. For many learners, particularly those with little or no programming experience, courses can feel intimidating when peers appear substantially more prepared, discouraging participation and help-seeking and exacerbating feelings of exclusion or impostor syndrome. Traditional lecture-based formats often fail to provide sufficient feedback or opportunities for students to articulate reasoning or collaborate with peers. Prior work in computing education has consistently shown that structured, collaborative, inquiry-based learning can reduce these barriers while improving conceptual understanding, retention, and problem-solving performance. One goal of our approach is to create a supportive environment where students can work together, leverage diverse perspectives, and engage with course material in a way that feels enjoyable rather than threatening.

At the same time, the emergence of generative artificial intelligence presents both opportunities and challenges. AI tools have the potential to provide immediate feedback, help students identify misconceptions, and scaffold their reasoning. However, unmanaged AI support may bypass learning objectives, provide over-assistance, or conflict with pedagogical intent. As a result, there is a need for learning environments where AI is tightly aligned with instructor goals, supports guided reasoning, and integrates naturally into collaborative activity structures.

To address these challenges, we developed an AI-enhanced guided inquiry platform that blends the pedagogical strengths of Process-Oriented Guided Inquiry Learning (POGIL) with a robust technical architecture for collaborative programming instruction. The system enables instructors to author guided-inquiry activities using a flexible markup language or Google Docs, embed hidden AI directives that shape the nature of automated feedback, and deploy

these activities to student groups working synchronously in class.

Within the system, students work in small rotating groups, adopting structured roles (e.g., facilitator, spokesperson, quality controller, process analyst) that promote equitable participation, metacognitive reflection, and collaborative sensemaking. The system enforces turn-taking, tracks active participation, and synchronizes workspaces in real-time, allowing groups to collaborate through exploration, concept invention, application, and challenge cycles.

To ground the discussion that follows, **Figure 1** presents the system architecture and data flow, mapping instructor-facing services, storage, and AI mediation, as well as the student collaboration workspace with its feedback loop.

A distinguishing feature of the proposed system is its instructor-aligned AI feedback layer. Rather than providing unconstrained assistance, AI responses are shaped by hidden instructor-authored directives describing the depth, tone, and structure of feedback. This preserves the integrity of guided inquiry by encouraging students to explain their reasoning and evaluate alternatives without shortcutting the intellectual work of conceptual construction.

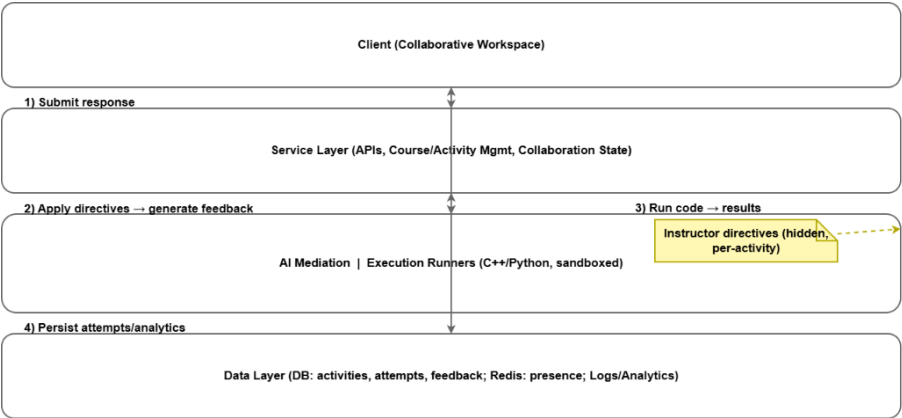


Figure 1: System Architecture and Data Flow. *A layered diagram depicting client, service layer, AI mediation, execution runners, and data stores; arrows show request/response paths for code execution and AI feedback.*

The system incorporates a secure execution environment for C++ and Python programming tasks, supports file I/O simulation, and logs student attempts for analytics purposes. Its backend architecture supports multi-course management, activity deployment, group monitoring, and data storage. Together, these pedagogical and technical components enable instructors to scale collaborative, inquiry-based instruction across many programming courses.

To preview how guided inquiry integrates with AI-mediated feedback in the proposed system, **Figure 2** illustrates the activity cycle, highlighting the AI touchpoints where responses are evaluated, nudges are generated, and analytics are logged.

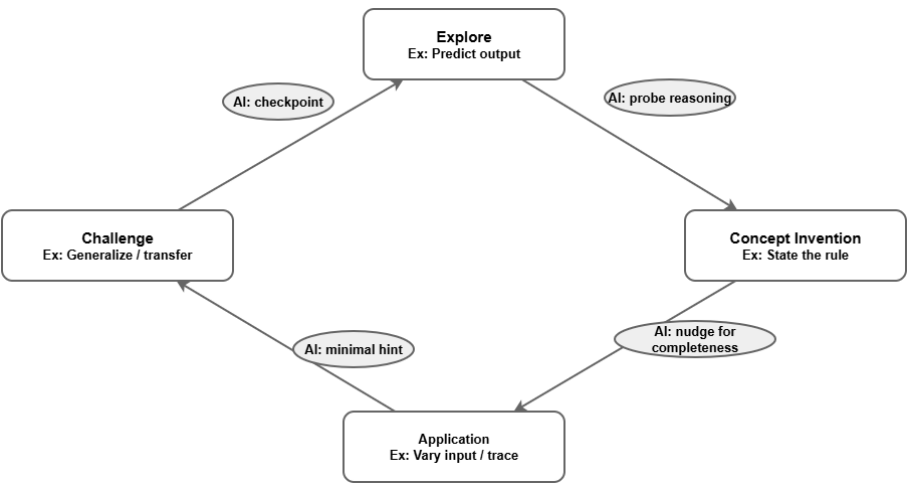


Figure 2: Guided Inquiry Cycle and AI Touchpoints. *The four inquiry phases (Explore → Concept Invention → Application → Challenge) are overlaid with AI interventions: directive-aware evaluation of submissions, generation of formative prompts, and logging/analytics for reflection and instructor monitoring.*

The remainder of this paper presents a comprehensive description of the proposed system, including its pedagogical foundations, system architecture, activity model, collaborative learning mechanisms, AI feedback pipeline, and ongoing research and development plans.

2 Related Work and Pedagogical Foundations

Research in computer science education consistently emphasizes active learning and structured student engagement as key factors in achieving improved outcomes. Guided-inquiry approaches, particularly those inspired by POGIL, have demonstrated effectiveness in fostering collaboration, conceptual understanding, and student agency in introductory courses [3, 2, 7]. These methods emphasize structured team roles, shared responsibility, and cycles of exploration, concept invention, application, and challenge.

Collaborative learning has been shown to enhance students’ reasoning processes by promoting explanation, peer instruction, and negotiation of meaning;

these effects are especially relevant for programming tasks that require articulation of strategy and justification of choices [4]. Inductive pedagogies (including inquiry) provide a general framework for sequencing experiences, allowing students to infer principles from evidence rather than receive them through exposition [8].

In programming education specifically, inquiry structures have been used to organize multi-stage reasoning and sustain engagement across problem sequences [1]. At the same time, work on AI in education and intelligent tutoring systems (ITS) underscores both the promise and the constraints of automation: systems can provide hints, formative feedback, and assessment at scale, but must be aligned to pedagogical goals to avoid counterproductive over-assistance [5, 9].

These strands motivate our design, which integrates guided inquiry, role-aware collaboration, and instructor-aligned AI within a single system.

2.1 Active, Inductive, and Collaborative Learning

Active learning and cooperative structures are associated with improved outcomes and equitable participation [3, 4]. Inductive pedagogies (inquiry-, problem-, and discovery-oriented) sequence experiences so learners infer principles from evidence rather than receive them solely by exposition [8]. For programming, this translates into tasks that require learners to externalize strategy, predict program behavior, test hypotheses, and justify revisions. We adopt this stance for code-centric tasks where articulating strategy, justifying predictions, and reconciling conflicting interpretations are essential.

2.2 Guided Inquiry / POGIL Lineage and Adaptation

POGIL supplies a well-studied template for structuring small-group work via explicit roles and phased activities [2, 7]. Our adaptation follows a four-phase sequence (Explore → Concept Invention → Application → Challenge) implemented in a digital environment with lightweight role rotation and visible turn histories. Teams work in groups of three or four; rotation is enforced gently to foster equitable participation while maintaining a natural feel in synchronous settings. Reflection prompts require students to cite program behavior as evidence, reinforcing metacognitive habits noted in prior work [7, 4]. (See **Figure 2**.)

2.3 Instructor-Aligned AI Support

The AI in the system is not intended to function as an autonomous tutor, but as a pedagogically constrained assistant. Its behavior is governed by hidden

instructor-authored directives that specify the intended tone, level of scaffolding, and criteria for evaluating student work; prompts occur at defined touchpoints in the inquiry cycle (see **Figure 2**). In each step, students first attempt a response; the AI then issues targeted nudges, requesting a missing condition, probing for evidence, or re-anchoring to the prompt, after which teams revise. The goal is not simply to reach a “correct” answer, but to support students in collectively constructing well-reasoned explanations and solutions that reflect deepening understanding.

This approach ensures that

- feedback remains aligned with activity objectives,
- solutions are not revealed prematurely,
- misconceptions are surfaced and addressed productively, and
- guidance strengthens reasoning rather than encouraging answer-seeking.

This aligns with the findings of Lee and Moore [5], who emphasize the importance of instructor oversight and scaffolding in AI-mediated formative assessment.

2.4 Integrating Programming Work with Inquiry

Because many introductory concepts are best grasped by observing live behavior, the system incorporates secure execution for C++ and Python. Students explore, run, and revise code within the same collaborative activity where they justify predictions and reconcile differences in reasoning. The coupling of executable artifacts with conceptual prompts helps students connect rules and results, supporting debugging as a form of sense-making rather than trial and error [1]. Reflection prompts close the loop by asking learners to name the idea they now trust and how they would recognize or apply it in future work.

2.5 Gaps Addressed

Despite substantial progress in intelligent tutoring and auto-grading, most systems still optimize either individualized feedback or scalable assessment workflows, not the combination of structured collaboration, guided inquiry, and instructor-aligned AI. Systems that offer code execution often treat collaboration as peripheral (shared editors without pedagogical roles) or treat pedagogy as peripheral (auto-grading without inquiry). Likewise, AI feedback is often generic, providing hints that may be technically correct but pedagogically misaligned. Our system here addresses these gaps by binding three elements that are rarely integrated and creates an interactive, web-based learning environment: (i) a role-aware collaboration model that makes equitable participation

a first-class goal, (ii) a guided-inquiry activity structure that sequences exploration, generalization, and transfer, and (iii) an AI feedback layer that is explicitly authored and constrained by the instructor [3, 7, 8, 5, 9].

Importantly, the proposed system is not intended to operate as an independent or self-contained tutor. Its effectiveness depends on knowledgeable instructors who design high-quality guided-inquiry activities and understand how to shape the learning process. The system amplifies, not replaces, expert pedagogy: AI guidance functions as an extension of the instructor’s intentions, enabling scalable implementation of established learning practices rather than attempting to automatically teach arbitrary content. In this way, the system should be understood as an instructor-centered tool that leverages AI to increase the efficacy and reach of well-designed learning experiences.

3 System Architecture

The system architecture translates the foregoing pedagogy into a responsive, multi-service platform capable of supporting many sections and courses simultaneously. As previewed in **Figure 1**, the client renders activities and synchronizes group work in real-time, while the server orchestrates activity delivery, AI mediation, execution jobs, and data storage. A modern web front-end coordinates collaborative editing, presence, and turn-taking; a service layer exposes APIs for course management, activity lifecycle, and analytics; and a relational database persists activities, rosters, attempts, and feedback. A collaboration service maintains low-latency shared state for group interactions. When a prompt involves code, the request is routed to containerized runners (C++/Python) that compile or execute programs with resource/time limits, capture outputs, and return structured results linked to the group’s current step.

The AI mediation service sits alongside these components. When students submit a response, the server packages the response, the local context (e.g., earlier attempts, code output), and the instructor’s embedded directives. The AI service interprets those directives to shape its response, deciding whether to probe, nudge, or request clarification, and whether to ask a follow-up or accept the response, allowing the group to move forward. All exchanges, such as student drafts, code runs, AI prompts, and final submissions, are logged for later instructor review and research analysis. Instructor dashboards also surface basic learning analytics indicators to support timely intervention and early warning [6].

3.1 Activity Model and AI Feedback Layer

Activities are the unit of design that binds pedagogy to infrastructure. Each activity is parsed into a structured representation of sections, question groups, prompts, expected answer features, optional rubrics, code blocks, and instructor notes. This representation enables the client to render rich interactions while giving the server sufficient structure to route code runs, validate formats, and coordinate the AI. (See **Figure 1** for the end-to-end pipeline; **Figure 2** shows where AI engages within the inquiry cycle.)

The AI feedback layer is invoked whenever a step carries an embedded directive. Rather than answering the question, the AI examines the student's response against the author's intentions. If a response appears partial, the AI might request a missing condition; if it seems correct but brittle, it might propose a counterexample; if it is off topic, it might re-anchor the group to the prompt. In a code step, the AI may point out an observed discrepancy between the expected and actual output and request an explanation, rather than suggesting a wholesale fix. These behaviors are bounded by instructor-authored directives, enumerated in **Table 1**, so that feedback should remain formative, concise, and aligned with the author's intent. To avoid stalls, instructors can enable a controlled progress-with-feedback path (e.g., "Checker errors should not block"), allowing teams to advance when only minor issues remain while logging signals for later review. This preserves momentum without compromising the focus of the inquiry.

When appropriate, the AI can also honor rubric language by flagging evidence of specific criteria and recording that evidence for the instructor's analytics [5, 9].

Activities authored in a lightweight markup or imported from Google Docs are normalized into sections, prompts, expected features, rubrics, code blocks, and instructor notes, allowing the system to route execution, apply rubric logic, and coordinate AI (see **Table 1**).

3.2 AI Mediation: Model and Inference Specification

The platform uses a large language model (ChatGPT) as a formative-feedback assistant. The base model is trained and maintained by the provider on licensed, public, human-generated sources. We do not fine-tune; instead, we steer behavior through prompt conditioning grounded in instructor-authored directives. Course artifacts are stored in our database and supplied only at inference time; they are not used to train the base model.

At runtime, each feedback turn is built from a directive-aware prompt. A concise system role defines the assistant's function ("a teaching aide constrained by instructor directives and pedagogy"). Hidden directives authored

Table 1: Examples of AI Directives and Pedagogical Intents. *Directives are hidden in activity files and shape feedback scope, tone, and gating during each step of the inquiry cycle.*

Directive (hidden)	Operational Effect on Feedback	Pedagogical Intent
Follow-ups: gibberish-only	Provide guidance only when responses are empty, off-target, or incoherent; allow productive struggle otherwise	Minimize interruptions; respect productive struggle
Requirements-only	Highlight missing conditions; avoid giving strategies	Preserve inquiry; focus on completeness
Match sample output exactly	Enforce exact labeling/ordering on final checkpoint	Support assessment reliability
Checker errors do not block	Provide hints but allow progression after minor issues	Maintain momentum; avoid over-scaffolding
Concise, single-sentence feedback	Compress responses for fast rhythm in live collaboration	Reduce cognitive load; keep teams talking

by the instructor (such as Follow-ups: gibberish-only, Requirements-only, Concise feedback, Fail-open) specify tone, scaffolding bounds, and follow-up policy. The context window contains only the minimum needed to reason about the current step — activity and item identifiers, the prompt text, brief rubric cues, paraphrases of tests or exemplars (not answer keys), and small, quoted excerpts of the student’s code or output. For open-practice steps, we use low-temperature sampling (temperature = 0.2) to produce succinct, targeted nudges. For stricter checkpoints, an even lower temperature can be used. The service expects a structured reply: a one-to-three-sentence rationale that cites concrete evidence (e.g., line numbers or tokens) and, where configured, a compact set of feature flags indicating missing conditions or misconceptions. All responses are schema-validated and truncated server-side before storage or display.

Inference calls are issued by the Activity service against the OpenAI chat completions endpoint with per-request credentials, rate limiting, and latency/-cost logging. The web client (single-page application) renders feedback inline during practice and suppresses hints during locked checkpoints while still col-

lecting student justifications. All prompts and responses are recorded with step metadata to support reproducibility and instructor oversight.

Privacy safeguards are enforced server-side. Prompts exclude personally identifiable information (PII), including names, email addresses, IDs, and identifiable paths or comments. Only minimal excerpts needed for the current turn are transmitted. Vendor training on our prompts and responses is disabled where supported, and audit logs retain redacted prompts, model outputs, and user actions with timestamps for later review.

This mediation layer makes the AI component a documented, reproducible part of the system rather than a black box: model and interface are specified, inputs are minimized and redacted, outputs are constrained by instructor intent and validated for form and brevity. In combination with the activity model, it yields feedback that is predictable in tone, bounded in scope, and inspectable after the fact, and amendable to the evaluation metrics reported later, while preserving the pedagogical primacy of guided inquiry.

3.3 Collaborative Group Model

The collaborative layer is designed to make equitable participation the default rather than an aspiration. **Figure 3** illustrates the collaborative group dynamics, including rotating roles, turn-taking, and participation signals used during each activity step. Each group's workspace surfaces the "active student" and a transparent history of turns, so that orchestration is visible and contestable: students see when it is time to drive, and when to review. This visibility matters because the benefits of collaboration only appear when conversational obligations are distributed and when reasoning is externalized.

From the instructor's perspective, a session is a moving field of small teams. The dashboard provides a real-time overview of all groups in class, showing which students are logged in, who is currently the active participant within each group, and how far each team has progressed through the assigned activity. Instructors can immediately identify stalled or struggling groups based on progress indicators, common errors, or repeated AI guidance triggers, and determine which teams may require additional support or intervention.

Because AI feedback is constrained by embedded directives, its prompts are predictable in tone and scope, allowing instructors to rely on it as a partner rather than an uncontrollable variable. Over multiple offerings of a course, instructors can compare runs of the same activity to see how different cohorts engaged, which items were consistently sticky, and how revisions to wording or exemplars affected learning dynamics [3, 1].

Instructors can adopt the system incrementally, beginning with inquiry activities and reflections, then adding live execution, and later enabling rubric checkpoints or summative assessments. Reliability features (checkpointing,

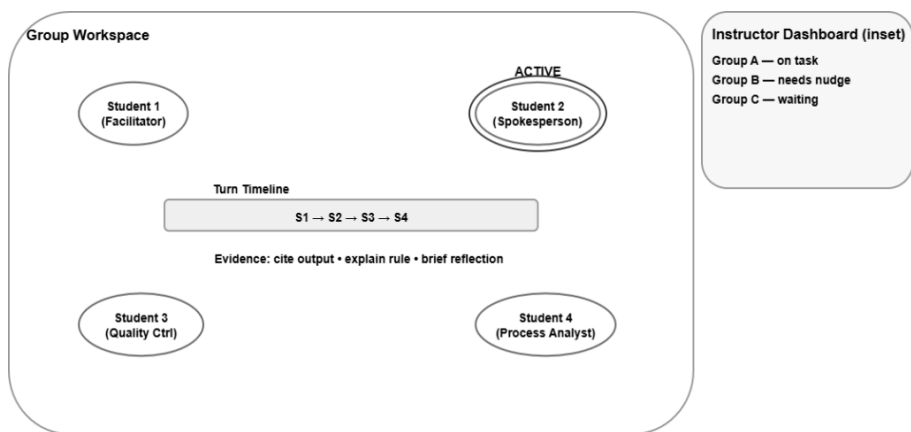


Figure 3: Group Collaboration Model. *An annotated sketch showing rotating “active” role, shared workspace, visible turn history, and instructor overview panel.*

execution logs, and autoscaling runners) and basic analytics support timely interventions and early detection of at-risk patterns [6].

4 Evaluation Plan and Early Observations

A mixed-methods evaluation is planned, combining learning outcome measures with process analytics and qualitative accounts of classroom experience. The quantitative strand examines performance on near- and far-transfer items, changes in error profiles on representative code tasks, and retention across the term. The qualitative strand analyzes group transcripts, reflection responses, and instructor field notes to understand how guidance, role rotation, and AI prompts shape talk, explanation, and consensus-building [7, 4]. Preliminary observations suggest that visible turn-taking smooths participation and reduces dominance by a single strong student, while directive-aligned AI prompts are effective at eliciting missing conditions and reframing off-track reasoning without revealing complete solutions. These impressions align with prior findings that inquiry structures and group roles can enhance equity and conceptual engagement in programming contexts [1] and that AI feedback is most effective when its behavior is guided by pedagogy rather than left generic [5].

The system has been implemented and is in active use in multiple university course sections. The following summarizes the current status of the system implementation:

- Activity authoring (markup + Google Docs import) — implemented
- Grouping/turn-taking, real-time sync — implemented
- C++/Python runners (containerized, resource caps) — implemented
- AI mediation (ChatGPT via OpenAI API) — implemented
- Assessment mode (single-learner lock, timer, hint suppression) — implemented
- Instructor review with audit trail, publish/finalize — implemented
- Analytics warehouse & exports — partially implemented
- Optional LMS sync — planned
- Versioned item bank & regrade workflow — planned

We will also track how revisions to AI directives (e.g., shifting from “requirements-only” to “default” follow-ups, see **Table 1**) affect student behavior and solution quality across iterations. Dashboards will be augmented with early-warning indicators to support timely intervention [6].

It is essential to note that the proposed system does not eliminate the need for active instructor involvement. As with any group-oriented, collaborative activity, instructors should monitor engagement, observe how well students enact their assigned roles, and intervene when groups struggle with collaboration. Effective use of the system requires instructors to circulate, review group dashboards, and encourage equitable participation and serious engagement with each question.

5 Limitations and Future Work

Several limitations remain. First, alignment of AI feedback with instructor goals depends on the quality of directives; poorly specified guidelines can yield prompts that are too terse, too leading, or insufficiently sensitive to the activity’s aims. Second, even with role rotation, groups can drift into asymmetric participation; additional nudging mechanisms and analytics may be needed to maintain balanced contributions [4]. Third, while the current runners support C++ and Python well, broadening language coverage and improving multi-file project ergonomics will be important for upper-division use. Finally, we have only preliminary classroom experience; results to date are observational rather than causal. These limitations motivate a more rigorous evaluation and continued refinement of directive templates, authoring linting, and operational safeguards (e.g., autoscaling runners, offline fallbacks). Future work will therefore focus on (i) directive authoring tools with previews and “safety checks,” (ii) lightweight collaborative debugging views that make reasoning about state a shared activity, (iii) adaptive group-formation and rotation policies informed by process data, and (iv) an evaluation study that assesses the efficiency of this pedagogical approach, such as learning achieved per unit of student and

instructor time, comparing the system against strong baselines. The study will investigate whether directive-aligned guided inquiry yields faster mastery with equal or reduced instructional effort, while monitoring collaboration equity and hint dependence as process indicators [3, 4, 1].

6 Conclusion

The proposed system combines guided inquiry, equitable collaboration, and instructor-aligned AI into a single platform, making high-quality, discussion-rich programming instruction feasible at scale. AI, under explicit directives, becomes a reliable contributor to the learning process, capable of asking the questions an expert tutor would ask. By treating AI as a controlled pedagogical tool and by making participation structures visible and lightweight, the system supports reasoning, reflection, and transfer rather than simply providing answers. Early deployments suggest that the approach reduces instructional bottlenecks and deepens conceptual engagement, while the architecture provides a foundation for credible evaluation and iterative improvement. As institutions seek sustainable ways to blend human facilitation with intelligent tools, the system offers a model where pedagogy and automation reinforce each other rather than compete.

References

- [1] Shamsa Ahmed, Syed Ali Hassan Bukhari, and Amir Ahmad. “Inquiry-based learning in software engineering education: Exploring students’ multiple inquiry levels in a programming course”. In: *Frontiers in Education* 10 (2025), Article 1503996.
- [2] James J. Farrell, Richard S. Moog, and James N. Spencer. “A guided-inquiry general chemistry course”. In: *Journal of Chemical Education* 76.4 (1999), pp. 570–574.
- [3] Scott Freeman et al. “Active learning increases student performance in science, engineering, and mathematics”. In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415.
- [4] David W. Johnson, Roger T. Johnson, and Karl A. Smith. “Cooperative learning: Improving university instruction by basing practice on validated theory”. In: *Journal on Excellence in College Teaching* 25.4 (2014), pp. 85–118.
- [5] Sang Soo Lee and Rhonda L. Moore. “Harnessing generative AI (GenAI) for automated feedback in higher education: A systematic review”. In: *Online Learning* 28.3 (2024), pp. 82–104.

- [6] Owen H. T. Lu et al. “Applying learning analytics for the early prediction of students’ academic performance in blended learning”. In: *Educational Technology & Society* 21.2 (2018), pp. 220–232.
- [7] Richard S. Moog and James N. Spencer. *Process-Oriented Guided Inquiry Learning (POGIL)*. Oxford, U.K.: Oxford University Press, 2008.
- [8] Michael J. Prince and Richard M. Felder. “Inductive teaching and learning methods: Definitions, comparisons, and research bases”. In: *Journal of Engineering Education* 95.2 (2006), pp. 123–138.
- [9] Ido Roll and Ryan Wylie. “Evolution and revolution in artificial intelligence in education”. In: *International Journal of Artificial Intelligence in Education* 26.2 (2016), pp. 582–599.

Teaching Programming at a Small College in the Era of GenAI*

Christine F. Reilly
Computer Science Department
Skidmore College
Saratoga Springs, NY 12866
`creilly@skidmore.edu`

Abstract

The era of GenAI began with the release of ChatGPT 3.5 in November 2022. GenAI systems that can generate passable code for undergraduate-level programming-focused courses require a shift in how student learning is assessed and potentially in the content of these courses. Rather than trying to create assignments that the current GenAI systems cannot easily generate code for, this paper suggests approaching the GenAI era by updating assessments of student learning and incorporating GenAI into the coursework. Small class sizes at small colleges provide an excellent environment for experimenting with new pedagogical approaches.

1 Introduction

This paper serves as one example of how a programming-focused course was adapted in light of GenAI systems that can produce passable code for undergraduate level assignments. The November 2022 release of ChatGPT 3.5, its later improvements, and a variety of similar systems caused shockwaves across academia. These GenAI systems generate essays and computer code, provide summaries of readings, and stand in for human tutors.

The subject of this paper is a 200-level computer science course focused on software design and object oriented programming in Java. The course is offered

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

at a small liberal arts college in the Northeast United States. This college has approximately 2500 undergraduate students and the computer science major graduates 20 to 40 students per year. The class described in this paper has a capacity of 16 students per lab section with one or two lab sections combining for a capacity of 16 or 32 students in the lecture class. In the semester discussed in this paper, the class was particularly small with 6 students enrolled in the lecture and lab class. The small class size enabled the professor to experiment with new approaches in the classroom, including projects that benefit from frequent check-ins and feedback while students are working on them. The lessons learned during this semester can be applied to a future semester with a larger class and to other colleges.

The goal of this paper is to explore pedagogical approaches that are appropriate for a programming-focused course in the era of GenAI. The pedagogical approaches used for the course should be robust enough to be reused for many years. The course design should be able to easily adapt as the code generation capabilities of GenAI systems continue to improve. A second motivation for this robust course design is so the faculty member does not need to redesign the course every time they are assigned to teach it. At this college, the 100 and 200-level courses rotate among the computer science faculty based on a variety of course scheduling factors. Sometimes a particular faculty member teaches the same course multiple semesters in a row, and other times one or two academic years may pass before a particular faculty member teaches the course again.

This paper begins by reviewing related work about teaching computer science during the era of GenAI. It then reflects on recent events that have impacted computer science pedagogy including the COVID-19 pandemic and the wide availability of GenAI systems. Then the paper presents how the particular 200-level software design course was adapted for the era of GenAI. Finally the paper summarizes the lessons learned and looks forward to ongoing improvements in the pedagogical approaches for this course.

2 Related Work

Because GenAI systems are relatively new and their capabilities are frequently improving, the review of related work mostly focused on papers published within the past year (since mid-2024), with a strong focus on the papers published in the SIGCSE 2025 proceedings. Due to the lag between when a paper is submitted and when it is published, these related papers reflect GenAI as it was in 2023 and 2024.

A literature review of papers published before June 2024 combined with a survey of instructors and industry professionals in July 2024 provides a sum-

mary of the state of teaching computer science in the early era of GenAI [8]. This study found students make the best use of GenAI when the instructor provided guidance through a tool with built-in guardrails or instructions on how to use the technology. Many of the surveyed faculty believe the presence of GenAI changes the balance of the skills needed for software development. There is now a stronger focus on the ability to understand and explain code and less need for memorization of syntax. Some of the surveyed faculty predict a branching of current computer science into two fields: “computer scientists and conversational programmers”. However, as of mid-2024 there was not much information about how to alter computer science classes to reflect this change in the balance of skills. The main changes the surveyed faculty made to their classes was to change their assessment practices to put less weight on code written outside of class. Some faculty increased the weight of in-class exams or have in-class proctored assignments, while others shifted assessments to focus on the process of software development. Many faculty have concerns about the privacy of students’ data in GenAI systems.

Another literature review surveyed papers published between January 2019 and June 2024 [9]. This survey found that students generally had positive experiences with GenAI, although they expressed some challenges with creating prompts that generated the desired output and with GenAI responses that were not relevant or not accurate. Instructors found that students who used GenAI had lower grades. These instructors expressed concerns about academic integrity, whether students were learning the concepts, and whether students were relying too heavily on the use of GenAI tools.

As computer science educators face changing how programming is taught, the research of the top-down versus bottom-up approaches to teaching reading and mathematics can be drawn upon for reference [12]. There is some evidence that using LLMs for a top-down approach for learning programming leads students to have an inflated sense of their competence and results in students not understanding fundamental concepts. However, the authors acknowledge the social context of learning programming and suggest that the use of LLMs may help keep students engaged. They conclude by suggesting a balanced approach to teaching programming that combines top-down and bottom-up approaches. Much of the research on top-down versus bottom-up approaches to teaching reading and mathematics has been conducted on young learners in elementary schools. It will be important to continue this exploration in the context of college-level learners. Another interesting avenue for exploration is if there is a difference between students who have little to no prior programming experience, such as in a CS0 or CS1 class, and those learning more advanced programming skills or learning a second, third, or more programming language.

Computer programming combines two types of work: writing instructions

for a machine to perform a series of operations, and creating a computational solution for a problem. GenAI is relatively good at the task of writing instructions. It is now becoming more important for humans to focus on the creative aspects of designing solutions [5].

Some faculty aim to design programming assignments that cannot be completed by GenAI [3]. However, the GenAI systems are continuously evolving. The assignments that GenAI systems could not complete in one academic year may be easily completed by the GenAI systems available in the next academic year. Attempting to create GenAI-proof assignments results in a never-ending race of professor against GenAI.

Another approach is to allow students to use GenAI. A study reported on the use of GenAI by students in a database course [10]. This study examined students' use of GenAI for a database concepts problem that did not involve SQL or other coding. The authors conclude that the use of ChatGPT led to higher grades on the one problem but lower grades on the overall assignment and course. They suggest students need guidance on how to use GenAI to support their learning. Another factor that may have impacted the results of this study is students self-selected which GenAI use group they enrolled in. It is possible that students who would have had lower course grades regardless of using GenAI may have self-selected into the group that used ChatGPT. The authors acknowledge that the individuals who self-selected into the group that used a variety of other GenAI tools may have been more exploratory students who are higher performers.

Industry is using GenAI for coding. In order to prepare students for a career in software engineering, colleges will likely need to incorporate the use of GenAI into computer science classes [1]. In 2025 there was a slowing down of hiring recent graduates with computer science bachelors degrees due to correcting the over-hiring that occurred in 2021 and 2022 combined with companies hoping that GenAI can do some of the entry-level programming work [11]. This slowdown of hiring by tech companies is balanced by overall tech hiring due to the increasing reliance on technology in many other industries. On the other hand, the code produced by GenAI often contains errors. Perhaps we are currently in an era of GenAI hype that will soon pass [4].

3 Recent Events Impacting CS Pedagogy

The five years from 2020 through 2025 have been ones of continuous change in how computer science courses are delivered. Prior to the Spring 2020 semester, it was common for computer science courses to have heavier weight on exams than on programming assignments. Ever since good web search tools and fast, ubiquitous internet connections appeared in the 2000s, professors have known

that a subset of students submit programming assignments that heavily draw upon the work of others.

And then came March 2020. Suddenly just about everyone in the United States shifted to online learning, including online exams students took using their own computer. At this college, computer science classes were also fully online throughout the 2020-2021 academic year. Because of the challenges of giving and proctoring online exams, the course grade put more weight on programming assignments and exams were modified to include coding problems.

Computer science classes at this college were back on campus in the 2021-2022 academic year but not back to normal. A student could be sent to isolation for 10 days if they had close contact with someone who tested positive for the COVID-19 virus. If many students in a class were in isolation at the same time or if the professor was in isolation the course pivoted back to online. Exams had to be flexible to accommodate one or more students taking it remotely. The Fall 2021 final exams at this campus were all remote because the college decided to close the campus when Thanksgiving break started rather than risking spreading the virus students acquired during their holiday travel. The heavy weight on programming assignments and take-home coding exams continued.

There was hope that the 2022-2023 academic year would settle into a new post-pandemic normal. Then the release of ChatGPT 3.5 in November 2022 introduced another disruption to programming, and many other, courses. It was a like a super-charged web search that handed students the code. The chatbot had some odd output making it somewhat apparent when students handed in the chatbot's work instead of their own. Over time the chatbot improved and became integrated into the popular development environments. Distinguishing between a student's work and the chatbot's output became more challenging. Programming assignments decrease in value when the chatbot's output can earn a passing grade.

Professors have been trying to figure out how to adapt assignments and assessments to this GenAI era. Some quickly embraced the new technology while others ignored it and hoped it would go away. Three years into this era it seems that GenAI technology is here to stay. The challenge now is to use pedagogical approaches that will stand the test of time as GenAI systems continue to improve, and to adapt course content to ensure students have the experiences and skills they will need after graduation.

4 Adapting a Software Design Course to the GenAI Era

This section focuses on adapting a software design and object oriented programming in Java class to the GenAI era. In the semester this paper focuses

on, the class had six students. Quantitative results are not possible with this small of a class. In general, gathering an adequate amount of data for quantitative results from classes at this college would require compiling results from multiple semesters. Because of the small class size in the semester studied, qualitative results were not gathered because it would be challenging to maintain anonymity. The benefit of the small class size is it enables trying different ideas, including ones that involve a significant amount of faculty time and attention given to individual students. The experiences gained from this course can then be reflected upon and modified to be deployed in future classes and can be studied with larger numbers of students.

The weighting of various course activities in the students' grades in this course can be roughly divided into three eras: the pre-2020 era, the COVID era, and the GenAI era. Table 1 shows the contribution of different activities toward the course grade in three different academic years representing these eras.

Students have been able to easily find solutions and get outside assistance on assignments since the web became widely used in the 2000s. Prior to the COVID-19 pandemic, it was common for Computer Science courses at this college to have one or two midterm exams and a final exam count toward a large portion of the course grade. For example, in the 2017-2018 academic year, a similar course had 60% of its points from in-class exams.

The COVID-19 pandemic led to remote instruction in Spring 2020, and this Computer Science department had remote instruction for the entire 2020-2021 academic year. In the 2021-2022 academic year, protocols to slow the transmission of the virus resulted in a remote final exam period for Fall 2021 and the need throughout the academic year to accommodate students remotely taking exams while in mandatory isolation. These remote exam conditions combined with a desire to accommodate students' heightened anxiety around high-stakes assessments resulted in lower weight on exams and higher weight on programming assignments. The COVID pandemic provided useful lessons on assessing student learning. Continuing some of the flexibility in types of assessments and in how and when assessments are given continued beyond the crisis of the pandemic because faculty found these adjustments better address the diverse needs of learners.

Currently, in 2025, the availability of GenAI systems that can often produce passable code for assignments is pushing assessment back to timed, in-class tests. This section describes how lessons from the COVID era are incorporated into the shift back to higher weight on in-class assessments and how this course is adapting to the reality of the GenAI era. In-class tests are discussed first as they are the largest component of the course grade. Next, the use of programming projects that incorporate student choice is discussed. The last parts of

Table 1: Components of Course Grade in Various Academic Years

	Pre-2020 Era AY17-18	COVID Era AY22-23	GenAI Era AY25-26
Timed Tests	Exam 1 17% Exam 2 17% Final Exam 26%	Midterm Exam 20% Final Exam 20%	Lower Exam 10% Higher Exam 15% Final Exam 20% Quizzes 15%
Total	60%	40%	60%
Outside Class	Homework 30%	Homework 30% Projects 20%	Homework 15% Projects 20%
Total	30%	50%	35%
In Class	Lab Problems 10%	Lab Problems 5% Participation 5%	Participation 5%
Total	10%	10%	5%

this section address what is perhaps the more important long term adaptation: an explicit recognition of the availability of GenAI and the potential impacts of GenAI on student’s learning and future career.

4.1 In-class Tests

As in-class tests become a larger component of a student’s course grade in the GenAI era, the question arises of how to maintain the flexibility of assessment used during the COVID era while ensuring the work submitted by students is not unduly influenced by GenAI. This course utilized two approaches for in-class assessments.

First was the traditional approach of two exams during the semester and a final exam. A bit of flexibility was incorporated into the semester exams by having each student’s lower exam grade count toward 10% of the course grade and their higher exam grade count toward 15% of the course grade. This provides some accommodation for a situation where the exam falls in a challenging week for a student.

Second, on most weeks there was a ten to twenty minute quiz at the beginning of the lab period. Weeks that had an exam or project due did not have a quiz. This resulted ten quizzes during the semester. The quiz must be taken at the scheduled time. To provide students the ability to miss some class days or accommodate when a student has a challenging week, each student’s lowest two quiz grades are dropped when calculating their course grade. The goals of the quizzes are to have an in-class assessment of the out-of-class work that was recently submitted, and to provide a lower-stakes opportunity for students

to gain practice on the types of questions and with the environment for the exams.

A major reason why short quizzes had not been used in the past was due to uncertainties about how to handle the timed quiz for students have disability accommodations for timed tests including extended time, distraction reduced environment, and/or use of a laptop for testing. At this college, approximately 25% of the student population has testing accommodations. In some semesters none or few of the students in a class have accommodations while in other semesters around one third of the students in a class have testing accommodations. Students with accommodations may take their tests at the college testing center, or the professor may make other arrangements to provide accommodations. Discussions with colleagues in other departments revealed that other professors at this college successfully used short in-class quizzes. These colleagues suggested making the quiz problems relatively straightforward and giving more than enough time for students to complete the quiz. The quiz procedures were discussed with the class at the start of the semester and students who have testing accommodations were invited to have a discussion with the professor about how to best meet these accommodations with the quizzes. None of the students in the class discussed in this paper had testing accommodations. In future semesters, an idea for providing accommodations is to have the student take the quiz in a conference room nearby the classroom and start early enough so their extended time ends at the same time as the class is finishing the quiz.

4.2 Programming Projects

Each of the two programming projects counted toward 10% of the course grade. The first project is fairly structured and is an adaptation of an assignment from the EngageCSEdu resource [7]. Using an abstract class that represents a generic pet, students create a class for a specific pet of their choosing. Their pet class must implement the abstract methods from the generic pet class and additionally have some properties and/or actions that are specific to the particular type of pet. This student-chosen aspect of the assignment is an attempt to make it less prone to being easily completed using GenAI and to motivate student engagement with the assignment.

Because of the small class size, it was decided to have an open choice for the topic of the second programming project. Students were provided with a list of required Java components to ensure their project utilized a variety of topics from the semester. Over the course of the month spent working on the second project, there were multiple checkpoints where the professor provided feedback to each student about the progress they were making on the project.

For grading the projects, in addition to reviewing and running each stu-

dent's code the professor had an individual meeting with each student to discuss their project. While related work discusses ideas for structuring code interviews in larger classes [2], the small size of this class allowed for less structured and more individualized discussions. Prior to the meeting the professor reviewed the student's project and made notes about aspects of the project to discuss with the student. The goal of these individual meetings was to have an opportunity to see how well the students knew their code with the hope that this may reveal use of GenAI. For the most part, students appeared to use GenAI within the acceptable boundaries. Students reported limited use of GenAI for the projects and lacked knowledge of how to best utilize GenAI for the class. When asked about their opinions of GenAI students expressed hesitation about using it for a class where they wanted to gain the knowledge. It is acknowledged that students may not fully report their use of GenAI directly to their professor.

4.3 Discussing GenAI

During the first week of the semester and near the end of the semester the class took a pause from the regular topics to consider the use of GenAI in our class and in programming more generally. Students read assigned readings, watched videos, and suggested additional readings or videos.

Three readings were assigned in the first week of the semester [4, 5, 11]. These readings were selected after reviewing articles published within the previous 6 months in *Communications of the ACM*. Each of these is a short article written for an audience with a general understanding of computer science and programming; in other words, accessible to students in a 200-level computer science course. For each article, students were asked to write a 1 to 3 sentence response to the following two questions:

- What did you find most interesting about this article?
- Based on this article, is there something you have a question about or do not agree with?

Students were also asked to provide a link to something they have read, watched, or listened to that is related to GenAI and their education (with the restriction that the media is a source available to the professor and classmates without needing to create an account, other than the college library account). Students then wrote a 3 to 5 sentence summary of that media item including why they found it interesting.

Additionally, students were asked to write a 1 to 3 sentence response to two summary questions:

- What are you challenged by or worried about regarding studying computer science in the presence of Generative AI tools?

- What are your thoughts about how this class can help you with studying computer science in the presence of Generative AI tools?

Near the end of the semester, the class revisited the topic of programming with GenAI. Based on discussions with individual students, they seem to not be using GenAI to the extent the professor anticipated. While this is certainly a positive outcome showing that students are engaged with learning the material rather than taking shortcuts, the professor felt it would be beneficial to provide students with more guidance on using GenAI tools for programming and selected a YouTube video produced by GitHub with a demonstration of how to use the GitHub Copilot tool for programming with GenAI¹. Additionally, the class watched a video called "Learning Software Engineering During the Era of AI"², a recent TEDx talk that presents an optimistic view of software engineering evolving to focus more on design than on coding.

4.4 Guidance on use of GenAI

On programming assignments, students were provided with guidance on acceptable use of GenAI. Two categories of recommended use were provided. As detailed in Table 2 the guidelines for the category of “None or Minimal” use of GenAI suggests that students use examples from class and the textbook to help them with writing the code, and consult GenAI with specific questions about Java syntax. On the other hand, as detailed in Table 3, the “Moderate” category of GenAI use allows students to use GenAI for assistance with writing repetitive code, unit tests, and Javadoc style comments.

Table 2: Guidance for None or Minimal Use of GenAI	
	• Use the example code from class and the textbook to help you write the code for this assignment. • You may consult GenAI with specific questions about Java syntax. • In the header comment in your solution file: include a statement about how you used (or did not use) GenAI tools to help you with this assignment.

Informal feedback from students indicated appreciation of this guidance. The students in the class discussed their desire to learn how to program and their understanding that in order to be successful computer science students

¹<https://youtu.be/b5xcWdzAB5c>
²<https://youtu.be/w4rG5GY9IIA>

Table 3: Guidance for Moderate Use of GenAI

- Use the example code from class and the textbook to help you write the code for this assignment.
 - You may consult GenAI with specific questions about Java syntax.
 - You may use GenAI for assistance with writing repetitive code (such as accessor and mutator methods), help with writing Javadoc style comments, and assistance with writing unit test methods.
 - In the header comments in your solution files or in the comments for a method: include a statement about how you used (or did not use) GenAI tools to help you with that part of the assignment.
-

and to become successful professionals they need to understand how to write and read code. These two categories of guidance provided students with information about when to focus on writing code without the assistance of GenAI and when GenAI assistance is appropriate. This guidance specifically identifies the types of repetitive tasks where GenAI can complete the task without taking away from student learning.

5 Conclusions

At the time this paper was written, GenAI has been available for three years and has continuously improved. GenAI cannot be ignored; courses and pedagogical practices need to figure out how to adapt to it. The most useful adaptations are those that will stand the test of time as GenAI systems improve and change. In this era of GenAI, colleges with small classes have an advantage both for the students and the faculty. As always, students benefit from directly interacting with faculty in small classes. Through the relationship students have with their professor they can discuss the impact GenAI may have on their future career and can explore this and other new technologies. Professors have the benefit of getting frequent informal feedback from students in a small class and can experiment with pedagogical approaches that require interaction with individual students and detailed grading of programming assignments. The pedagogical approaches explored in a small class can be reflected upon and adapted to work with larger class sizes.

5.1 Next Steps

At the end of the academic year the professor will reflect on what worked well with these new pedagogical approaches and what needs improvement. The end of semester course evaluations completed by students will provide additional information about students' perspectives of the course.

One aspect of this course that needs improvement is the open-ended programming project. While this project was manageable with 6 students, in its current form it would be difficult to scale to the maximum size of this class at up to 32 students divided across two lab sections. One idea is to transform this into a semi-structured project where students choose from a list of topics. Some rough structure could be provided for each of the topics. Students could also be required to work in pairs or small groups. The project discussions need to be streamlined to take less time per student. One idea is to use questions that are more targeted at specific parts of the student's code. Another idea is for students to give presentations in class, then for the professor to have a shorter, more targeted discussion with each student.

The course will continue to evolve. Other types of assessments such as oral exams [6] could be deployed in future semesters. It is possible that GenAI will lead to a change in the skills and abilities companies expect from those completing a Bachelor's degree. In this case, programming education may evolve to focus more on design and testing and less on memorizing syntax. Small classes at small colleges are well positioned to continue to adapt pedagogical approaches to these ongoing changes.

References

- [1] Maggie Johnson. "Generative AI and CS Education". In: *Commun. ACM* 67.4 (Mar. 2024), pp. 23–24. URL: <https://doi.org/10.1145/3632523>.
- [2] Suhas Kannam et al. "Code Interviews: Design and Evaluation of a More Authentic Assessment for Introductory Programming Assignments". In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 554–560. URL: <https://doi.org/10.1145/3641554.3701806>.
- [3] Bradley McDanel and Ed Novak. "Designing LLM-Resistant Programming Assignments: Insights and Strategies for CS Educators". In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 756–762. URL: <https://doi.org/10.1145/3641554.3701872>.

- [4] George V. Neville-Neil. “The Drunken Plagiarists”. In: *Commun. ACM* 68.4 (Mar. 2025), pp. 22–23. URL: <https://doi.org/10.1145/3715926>.
- [5] Sebastian Nicolajsen and Claus Brabrand. “What Is Programming?” In: *Commun. ACM* 68.6 (June 2025), pp. 28–30. URL: <https://doi.org/10.1145/3713068>.
- [6] Peter Ohmann and Ed Novak. “A Multi-Institutional Assessment of Oral Exams in Software Courses”. In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 882–888. URL: <https://doi.org/10.1145/3641554.3701848>.
- [7] Jan Pearce. *Virtual Pets and ADTs*. <https://www.engage-csedu.org/find-resources/virtual-pets-and-adts>.
- [8] James Prather et al. “Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools”. In: *2024 Working Group Reports on Innovation and Technology in Computer Science Education*. ITiCSE 2024. Milan, Italy: Association for Computing Machinery, 2025, pp. 300–338. URL: <https://doi.org/10.1145/3689187.3709614>.
- [9] Nishat Raihan et al. “Large Language Models in Computer Science Education: A Systematic Literature Review”. In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 938–944. URL: <https://doi.org/10.1145/3641554.3701863>.
- [10] Valeria Ramirez Osorio et al. “Understanding the Impact of Using Generative AI Tools in a Database Course”. In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 959–965. URL: <https://doi.org/10.1145/3641554.3701785>.
- [11] Esther Shein. “The Outlook for Programmers”. In: *Commun. ACM* 68.5 (Apr. 2025), pp. 12–14. URL: <https://doi.org/10.1145/3715692>.
- [12] Sverrir Thorgeirsson, Tracy Ewen, and Zhendong Su. “What Can Computer Science Educators Learn From the Failures of Top-Down Pedagogy?” In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 1127–1133. URL: <https://doi.org/10.1145/3641554.3701873>.

ConnectPlus: A Demonstration of a Full-Stack Collaboration Tool for Academic and Team Projects*

Makayla Coleman, and Unnati Shah
Computer Science
Utica University
Utica, NY 13495
`{macolema,unshah}@utica.edu`

Abstract

This paper presents ConnectPlus, a full-stack project management application designed to unify multiple collaboration features within a single platform, including a dashboard, project workspace, messaging, task tracking, alerts, and user settings. The system addresses a common challenge in group work: communication often becomes scattered across multiple tools, leading to missed updates and fragmented information. ConnectPlus provides a centralized environment where students, instructors, and coworkers can manage tasks, share updates, and maintain project information in one dependable location. The application is implemented using the MERN stack, MongoDB, Express.js, React.js, and Node.js, with supporting middleware such as CORS, JSON parsing, and environment configuration to ensure reliable server behavior. A key feature is the integrated alerts system, which enables users to send immediate notifications about needs, locations, or urgent updates, helping prevent important information from being lost. Interface testing and iterative development were conducted to ensure usability and confirm improvements over basic prototypes. ConnectPlus serves as both a practical collaboration tool and a demonstration of full-stack development,

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

backend design, and scalable communication features. Additionally, the project emphasizes educational growth, highlighting the learning process involved in mastering React, Node.js, server-side rendering, security practices such as AWS and password policies, and asynchronous communication. This paper focuses on both technical implementation and the development of full-stack software skills, demonstrating the value of independent learning and hands-on project work.

1 Introduction

In academic and professional environments, group projects play a central role in learning and workplace collaboration. However, communication and organization tools are often distributed across multiple platforms. Students may rely on texting, email, and shared documents, while workplace teams frequently switch between chat systems, task boards, and file storage tools. This fragmentation leads to lost messages, missed deadlines, unclear responsibilities, and difficulties keeping team members aligned. As collaboration becomes increasingly essential in both educational and professional settings, the need for a unified, dependable workspace has grown. The proposed application was developed to address this challenge.

The proposed ConnectPlus application provides a full-stack project management application designed to integrate essential collaboration features into one platform. Its purpose is to offer a simple, reliable environment where teams can communicate, assign tasks, track progress, and manage shared information without depending on multiple disconnected tools. Whether the context involves student collaboration, teacher–student coordination, or workplace project teams, the application acts as a centralized hub where communication and work remain organized and accessible [geeksforgeeks2024mern, bateman2021].

To achieve this functionality, the backend is implemented using Node.js and Express, supported by a modular routing structure that separates major features into well-defined components. Key capabilities include secure user authentication, personalized project spaces, task management for creating and tracking work items, group creation for structured collaboration, and shared workspaces that keep updates in a centralized location. A notable feature is the alerts system, which provides real-time notifications to inform team members about urgent updates, assistance requests, or location-specific needs, ensuring that critical communication is never lost. MongoDB, accessed through Mongoose, offers persistent and structured storage for users, tasks, alerts, and project data. Middleware such as CORS, JSON parsing, and environment configuration ensures reliable client-server communication. Together, these components create a system that reflects real-world full-stack architecture while

delivering practical value to end users [geeksforgEEKS2023arch].

This project is designed as an independent study, and the student led all development phases. The project highlights learning outcomes beyond the code itself, including research, self-directed learning, and understanding asynchronous communication, security practices, and interface usability. While similar collaboration tools exist, the educational emphasis of ConnectPlus demonstrates the practical skills acquired during independent study and provides insights into learning full-stack development from the ground up.

The significance of this work lies in both its practical application and educational contribution. Practically, the application addresses a common communication challenge experienced by students, educators, and professionals. Educationally, the project demonstrates core concepts in full-stack development, backend API design, database modeling, and distributed system communication. By integrating technical depth with real-world usefulness, the system contributes meaningfully to both software development learning and collaborative productivity.

2 Related Work

Many collaboration tools offer features like task management and file sharing to help teams stay organized. Platforms such as Slack, Google Drive, and Microsoft Teams support real-time communication and shared workspaces in different ways. Slack emphasizes fast, channel-based messaging, while Google Drive focuses on shared document storage and co-editing. These solutions demonstrate various approaches to enabling teamwork, but they can feel overwhelming for beginners or for groups that only need a lightweight tool for a one-time project. This project addresses that gap by introducing a simpler, developer-friendly project management application that blends essential capabilities, task alerts, messaging, and basic collaboration, within a single full-stack codebase. It uses React for the interface, Express for the server, and MongoDB for data storage.

3 Methodology

The methodology for the proposed ConnectPlus application focuses on developing a unified collaboration system that integrates messaging, task management, notifications, and workspace organization into one platform. Instead of treating these capabilities as separate tools, the system merges them so users can move naturally between communication and productivity without switching applications. A modular design approach is used, where each major feature, such as authentication, alerts, messaging, and task tracking, is developed as an inde-

pendent module with its own API endpoints and data models. These modules communicate through well-defined interfaces, improving maintainability and allowing the system to evolve as new features are added [bateman2021].

The proposed application is implemented using a three-layer architecture composed of a React client, an Express server, and a MongoDB database. Each layer performs distinct responsibilities while remaining loosely coupled, ensuring that updates in one part of the system do not disrupt others. The React frontend organizes the application into pages for authentication, dashboards, task views, alerts, and messaging. Vite is used as the build tool to support rapid development and hot module replacement. The backend, built with Node.js and Express, follows a domain-based structure where each feature has dedicated route files and controllers, simplifying error handling, validation, and business logic. This structure ensures that new features, such as a future task-history module, can be integrated with minimal changes to existing components.

The database layer uses MongoDB Atlas for its flexible document schema, which supports quick iteration as features evolve. This flexibility was important when adding fields such as task attachments, priority levels, and archived status. The proposed architecture also allows for future expansion, including collections for permanent task history or extended user activity tracking. During the design phase, tools such as Figma and visual inspiration from dashboards like Slack and Microsoft Teams guided the layout and interface decisions. The implementation followed standard CRUD operations with controllers handling data processing, field validation, and integration with Mongoose.

Additional tools such as Mailtrap for email testing, Thunder Client for API verification, GitHub for version control, and Bootstrap for styling supported development. Authentication was implemented using bcrypt for secure password hashing, validation rules for required fields and password complexity, and email-based reset functionality through Mailtrap [thunderclient2024, mailtrap2024].

4 Demo Description

The ConnectPlus demo highlights the core functionality of the early development version of the project management system. Its purpose is to walk viewers through the primary modules, such as login, signup, and the dashboard, while demonstrating how users interact with the interface and how the system responds in the background. Although the platform is still under development, the demo presents the structure, workflow, and key features, offering a clear view of the foundation on which future enhancements will be built.

The demonstration begins on the login page, where valid credentials are

entered to access the system. This step illustrates password validation, username verification, and backend authentication through the database. After successful login, viewers are brought to the main dashboard, which gives an overview of the layout and provides navigation to the other major modules.

The My Tasks module is shown next. The demo walks through creating a task by entering a title, assignee, dates, priority level, description, and attachment type. Once submitted, the task appears in the task table after a state update or page refresh. The demonstration then covers editing an existing task, archiving a completed task using the completion checkbox, and permanently deleting a task while storing its details in the Task History database. The Summary/History view is also shown, displaying archived tasks separately from active ones.

The Alerts module follows, demonstrating how an alert is created, stored in MongoDB, and displayed on the alerts page after refresh. Fields such as title, project name, description, priority, and creator are shown to confirm proper database integration. The Messaging module is presented last. While still in an early prototype stage, the demo shows static messages, the addition of test messages, and the intended layout for future real-time communication.

Throughout the demonstration, viewers observe how the system handles routing, state updates, form validation, and communication with the backend. The demo is significant because it showcases a functional and scalable foundation for a full-stack project management platform in early development. It confirms the successful implementation of authentication, structured navigation, task workflows, alert creation, and initial communication features. It also highlights the system's potential for upcoming enhancements, including real-time messaging and collaborative task interactions. Overall, the demo validates steady progress toward a polished and deployable application. The full codebase and project can be accessed on GitHub¹.

5 Educational Integration

The development of ConnectPlus was carried out as an independent study in full-stack development, providing a structured environment for hands-on learning and skill application. The student led all phases of the project, including system design, frontend and backend implementation, database modeling, testing, and evaluation. This active engagement allowed practical application of core computer science concepts, including modular programming, data modeling, CRUD operations, middleware design, and secure authentication. The development process also reinforced experiential learning by bridging theory and practice. React components, RESTful APIs, and MongoDB collections

¹<https://github.com/Coop-3/Project-management-app.git>

were applied in real scenarios, connecting classroom-taught concepts to functional software development. Middleware development reinforced principles of error handling, input sanitization, and secure request processing, while browser developer tools supported structured debugging and iterative testing.

The student explored key new concepts, such as asynchronous communication, server-side rendering, and secure authentication. This included understanding how encryption works and assessing security practices beyond using standard libraries. While “improved usability” is claimed, user testing and interface iteration were conducted to confirm enhancements in user experience.

Evaluation and testing of each module encouraged systematic problem-solving, debugging strategies, and attention to reliable system design. Beyond individual learning, the proposed ConnectPlus application serves as a valuable teaching asset within peer-learning environments and formal coursework. The project demonstrates how an independent study can expand knowledge from basic web development skills (CSS, HTML, JavaScript) to advanced full-stack concepts, including React, Node.js, database integration, security practices, and asynchronous operations. The modular structure of the system, including authentication, alerts, messaging, and task management, allows classmates to collaboratively explore how distinct components communicate within a larger architecture. This promotes peer discussion, shared debugging activities, and cooperative design thinking. Instructors can reference the application when explaining secure login flows, CRUD operations, state updates, or error-handling patterns, giving students concrete illustrations that reinforce abstract concepts. The system also enables hands-on classroom engagement: students can extend features, debug existing modules, or propose architectural enhancements, making the platform a foundation for collaborative labs and peer-led mini-projects.

Overall, the proposed application demonstrates how self-directed development work can support CS pedagogy in full-stack development, software architecture, and collaborative system design. It contributes to both learning and teaching by linking theoretical instruction with practical development, supporting peer collaboration, and providing a robust example of modern full-stack software design.

6 Results and Discussion

The ConnectPlus project, an early-stage full-stack project management platform, was evaluated for core functionality and system behavior. Testing focused on prototype behavior, feature verification, debugging outcomes, and validation of the system architecture. Core modules, including authentication, task management, alerts, and messaging, were examined for correct data flow, API responses, and consistent storage within MongoDB. Interactions between fron-

tend and backend components highlighted both the strengths of the modular design and areas requiring refinement.

1. The task management module demonstrated the most complete functionality. Task creation, updating, deletion, and archival workflows were tested and validated. Tasks submitted via the frontend were stored in MongoDB, updated accurately, and archived tasks were correctly moved to the Task History collection for review or potential restoration. This confirms effective distinction between temporary archiving and permanent deletion, and that backend controllers and routing function as intended. Figure 1 shows a screenshot of the task management interface in action.

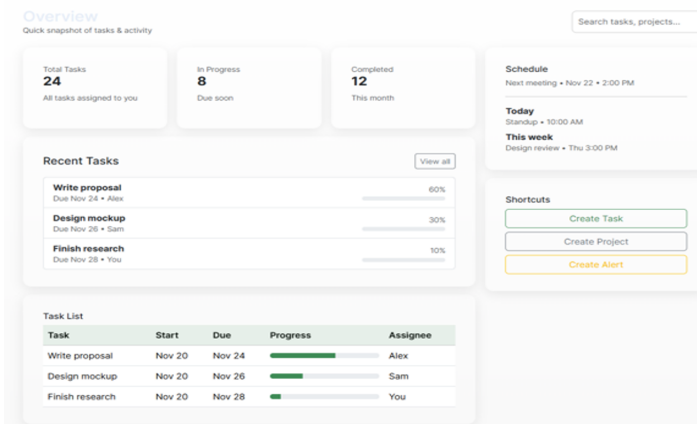


Figure 1: Task Management

2. The alerts module functioned reliably at a basic level. Alerts submitted via the interface were stored in MongoDB with fields such as title, project name, description, priority, creator, and resolved status accurately recorded. The system verified that data flows correctly from frontend forms to the database. Figure 2 shows a screenshot of the alerts interface and database integration.
3. The messaging module currently serves as a placeholder, supporting static messages and manual input. Demo messages rendered correctly, new sample messages appeared in the container, and the layout remained stable across page interactions. Real-time updates, user-to-user threading, message history, and advanced communication features have not yet

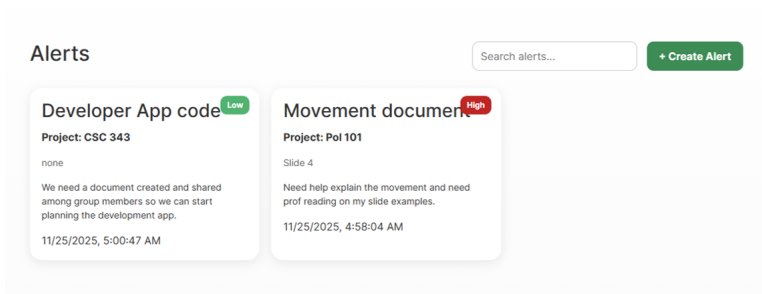


Figure 2: Alert Module

been implemented. Figure 3 shows a screenshot of the messaging module demonstrating its current functionality.

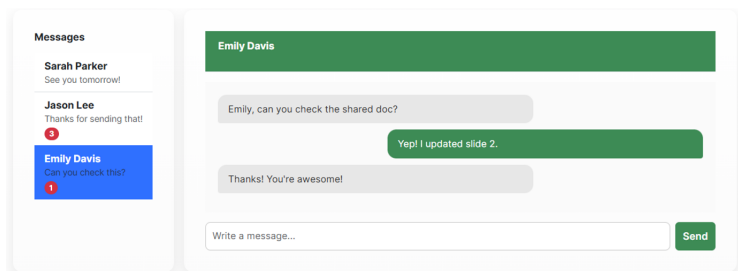


Figure 3: Message Module

4. The dashboard module successfully loaded primary user sections, including My Tasks, Projects, Alerts, and Settings, and demonstrated consistent page routing. Visual components such as cards, icons, and summary panels displayed expected placeholder data, reflecting the intended interface structure.

System testing confirmed proper operation of authentication and database interactions. User accounts were created with password complexity enforcement, email validation, and secure hashing. Login attempts correctly verified credentials, and errors were displayed for invalid attempts. Figure 4 shows the input validation feedback for user registration and login, demonstrating the functionality of login and registration interfaces. Task data, alerts, and user credentials were stored consistently and reliably in MongoDB. Overall, results

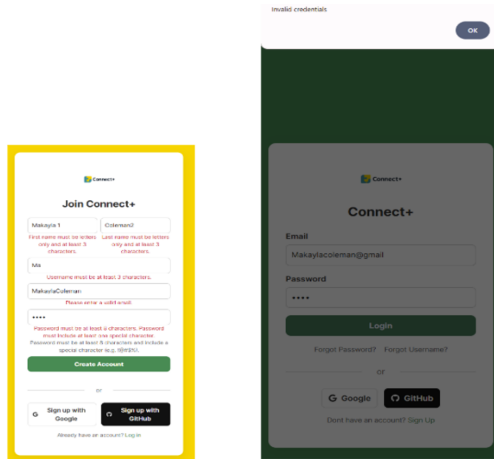


Figure 4: Input Validation Feedback for User Registration and Login

show that the proposed application provides a stable and scalable foundation, with task management demonstrating strong functional readiness and alerts and messaging modules establishing a framework for future development.

7 Conclusions and Future Work

The development of the proposed application ConnectPlus represents an important step toward a fully integrated project management platform for academic, workplace, and personal productivity environments. Testing and validation demonstrate that the core architecture is structurally sound and capable of supporting expanded functionality. Task management showed the most reliable results, including creation, updating, deletion, and archival workflows correctly routed through Express and stored in MongoDB. Authentication and user management were successfully validated, confirming secure password handling, login validation, and database storage practices. Alerts and messaging modules functioned at a basic but verifiable level. Alerts can be created and stored, while messaging supports static input. Advanced features such as real-time updates, filtering, prioritization, role-based visibility, and threaded communication remain to be implemented. Future work for the proposed application will focus on expanding interactivity and collaboration features, including:

1. Real-time communication: Implement WebSockets to enable live messaging, automatic task updates, and immediate alerts.

2. File management: Allow users to attach documents, images, and other resources to tasks, messages, or alerts.
3. Enhanced interface: Refine dashboard visuals, navigation, and module layouts to improve usability.
4. Advanced task and alert features: Introduce prioritization, filtering, notifications, and role-based visibility to increase productivity.

These enhancements aim to evolve the proposed application into a polished, scalable platform capable of supporting collaborative workflows. The current prototype demonstrates a strong foundation in full-stack development, modular architecture, and practical implementation of databases, frontend, and backend principles.

References

- [1] Thunder Client Team. Thunder Client REST API Testing Extension. VS-Code Marketplace, 2024. <https://www.thunderclient.com/>. Accessed: 2024.
- [2] Mailtrap Team. Email Testing Sandbox Documentation. Mailtrap.io, 2024. <https://mailtrap.io/>. Accessed: 2024.
- [3] GeeksforGeeks. MERN Stack Tutorials. GeeksforGeeks.org, 2024. <https://www.geeksforgeeks.org/tag/mern-stack/>. Accessed: 2024.
- [4] GeeksforGeeks. Architecture of a System. GeeksforGeeks, 2023. <https://www.geeksforgeeks.org/system-design/architecture-of-a-system/>. Accessed: 2024.
- [5] HuarcayaDeLacruzLuisFernando. mern-todo-app. GitHub repository, 2024. <https://github.com/huarcayadelacruzluisfernando/mern-todo-app>. Accessed: 2024.
- [6] Andrew J. Bateman. mern-stack-records. GitHub repository, 2021. <https://github.com/andrewjbateman/mern-stack-records>. Accessed: 2024.

Structured Post-Evaluation Interviews and Remediation (SPEIR): A Formative Assessment Workflow. *

Pablo Frank-Bolton¹, Michael Robson¹,
R. Jordan Crouser¹, and Rahul Simha²

¹Computer Science Department
Smith College, Northampton, MA 01063

{ pfrank, mrobson, jcrouser } @smith.edu

²Computer Science Department
George Washington University, Washington, DC 20052
simha@gwu.edu

Abstract

Students frequently choose correct answers for incorrect reasons, leaving traditional formative assessments unable to surface the misconceptions that matter most for learning. We introduce **Structured Post-Evaluation Interviews and Remediation (SPEIR)**, a workflow that extends two-tier Justified Multiple-Choice Questions (JMCQs) with guided discussions and targeted recovery opportunities to surface and address those hidden misunderstandings. Implemented across ten course sections and compared with a traditional MCQ control, SPEIR showed that correctness alone substantially overestimates understanding, while per-question analyses revealed higher rates of fully correct reasoning in SPEIR sections. Students who completed recovery quizzes demonstrated notable gains, and instructors reported that SPEIR enabled efficient, focused feedback. These results suggest that SPEIR is a scalable approach for integrating diagnostic assessment with timely remediation.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than CCSC must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1 Introduction

Formative assessment plays a central role in helping students refine their understanding, confront misconceptions, and strengthen emerging mental models. A substantial body of work has demonstrated that regular, low-stakes assessments can support conceptual change [12], improve motivation [8], and reduce anxiety [2]. Yet many common formats, particularly multiple-choice questions (MCQs), offer only a limited window into student thinking. Students can select the correct answer for the wrong reason, masking misunderstandings and limiting instructors' ability to target feedback.

Recent work highlights the need for assessments that focus not on what students chose but *why* they chose it [6]. Two-tier formats move in this direction by pairing correctness judgments with reasoning statements. Previous work in this area introduced the Justified Multiple-Choice Question (JMCQ) format [5], which was shown to surface misconceptions that traditional MCQs obscure. However, identifying misconceptions is only the first step; instructors also need a scalable workflow connecting diagnostic assessment to targeted remediation.

This work introduces **SPEIR** (**Structured Post-Evaluation Interviews and Remediation**), a workflow that integrates diagnostic formative assessment with guided reflection and opportunities for grade recovery. SPEIR extends the JMCQ format by using its diagnostic insights to structure follow-up discussions and remediation opportunities. The workflow aims to (1) reveal students' underlying reasoning, (2) support just-in-time clarification of misconceptions, (3) motivate purposeful restudy, and (4) provide a low-stress path for improving performance and understanding. SPEIR can be implemented with varying degrees of scaffolding, making it adaptable to courses with different levels of student preparation and autonomy.

To evaluate SPEIR, we implemented the workflow across ten sections of two undergraduate computer science courses taught by three instructors over three semesters (approved by the college institutional review board #21-037). We compared multiple implementations, including JMCQ-based SPEIR sections and a traditional MCQ control section, and examined outcomes at the question, quiz, and course levels. We also collected student perceptions of SPEIR's fairness, clarity, and impact on anxiety and motivation. This study addresses the following research questions:

1. Does SPEIR help reveal errors, misconceptions, and misunderstandings?
2. Does SPEIR help structure the follow-up discussions?
3. Is SPEIR scalable and transferable to other subjects?
4. Does SPEIR help with student engagement and motivation?
5. Does SPEIR help reduce student anxiety?

2 Previous Work

Formative assessment is most effective when it not only checks students' performance but also illuminates how they are reasoning through problems. While low-stakes assessment has been shown to support conceptual change, motivation, and self-regulation [8, 12, 7], many commonly used formats offer only limited insight into student thinking. This is particularly evident in computing education, where learners may select correct answers while relying on fragile or incorrect mental models [2, 3, 11].

A recurring challenge is the diagnostic gap inherent in traditional MCQs. As Hubbard et al. note, MCQs efficiently measure correctness but often obscure the reasoning behind students' choices, making it difficult to detect misconceptions or partially-formed understanding [6, 9]. Comparative studies of free-response, multiple-true-false, and other structured formats reinforce this point: assessment design strongly influences the visibility of student reasoning [6, 15].

To address these limitations, researchers have developed multi-level and reasoning-augmented assessment formats (most prominently in STEM education), where students justify their selected answer choices. Early work by Treagust [13] established the value of such approaches for diagnosing misconceptions by explicitly pairing correctness judgments with reasoning statements. More recent work in feedback research has highlighted the tension between diagnostic richness and instructor workload, noting that free-response justifications provide deep insight, but incur a high grading cost [14].

The Justified Multiple-Choice Question (JMCQ) format used in this study was introduced to provide a middle ground: a structured two-tier assessment in which students select both an answer option and a justification from a curated set of statements [5]. Prior research on JMCQs showed that justification scores are often substantially lower than correctness scores, revealing misconceptions that correctness-only scoring masks. The JMCQ format also reduces instructor workload compared with free-response approaches while retaining diagnostic value. However, this work focused primarily on the assessment mechanism itself rather than on how instructors might act on its diagnostic insights.

Although prior studies have examined the timing and structure of feedback [1, 4, 10, 14], few have explored a full, replicable workflow that combines diagnostic question formats with structured remediation that is both scalable and adaptable across courses. SPEIR is designed to fill this gap by providing a systematic approach to eliciting reasoning, clarifying misconceptions, and supporting targeted intervention in a low-stress environment.

Course	Semester	Prof-Section	Quizzes	Follow-up	N Students
Programming	S24	A1	6	Recovery	27
Programming	F24	A1, A2, B3	6	Recovery	30; 29; 21
Programming	S25	A1, B2	4	Recovery	28; 15
Theory	S24	A1, C2	7	Resubmit	26; 21
Theory	F24	A1	8	Resubmit	15
Theory	S25	A1	6	Resubmit	25

Table 1: Study sections.

3 Methodology

The trials were run in two different computer-science courses at Smith College, from Fall 2023 to Spring 2025. Table 1 shows the different implementations using two courses: an introductory programming course in Python (graded as S/U or Satisfactory/Unsatisfactory) and a theory of computation course (with traditional grading) usually taken after a student’s sophomore year. One instructor (denoted with the letter “A”) participated in all semesters and courses, and two other instructors (“B” and “C”) taught additional sections of the programming and theory courses using the SPEIR protocol. In total, 10 different sections in two different courses were taught over three semesters.

The trials were structured to address the five research questions outlined earlier. To characterize the types of issues that SPEIR can detect and correct, we extracted the distribution and frequency of correctness and justification errors for every section, quiz, and student. In Fall 2024, one programming section (A2) served as a control in which the full SPEIR workflow was used but quizzes were delivered in a traditional MCQ format. This provided a baseline for comparing the JMCQ format with an MCQ-only implementation of the same workflow.

For statistical comparisons of quiz performance, we used non-parametric tests due to the ordinal nature of the scoring. Pairwise comparisons were conducted using the Wilcoxon rank-sum test, and the Kruskal–Wallis test was used when three or more groups were compared, with post-hoc tests applied where appropriate.

To examine the effects of differing instructional scaffolding, the workflow was implemented in two remediation schemes. In the introductory programming course, instructors provided a more structured, closely guided follow-up discussion. In the upper-level theory course, students were expected to manage the review process more independently, reflecting the greater level of self-regulation expected at that stage in the curriculum.

The SPEIR protocol was implemented in courses organized around intro-

ducing a new topic every one to two weeks, each with defined learning objectives. After each topic, students completed a homework assignment designed to reinforce the newly introduced skills, followed by a short formative quiz. The quiz served as an opportunity for both students and instructors to detect and address emerging errors, misconceptions, and misunderstandings before they solidified. Because the SPEIR workflow depends on making student reasoning visible, the design of these quizzes plays a central role in how effectively the protocol surfaces conceptual difficulties.

3.1 Format of a JMCQ quiz

Each JMCQ quiz has two components: a *correctness* choice and a *justification* choice. The correctness component presents one correct answer and two distractors, each constructed to reflect a specific misconception. After selecting an answer, students choose a justification from a curated set of statements explaining why each option is correct or incorrect. This structure makes students' reasoning visible and enables automatic scoring.

JMCQs are designed around narrowly focused concepts and their associated misconceptions rather than broad, catch-all questions. Creating these items requires some initial authoring effort, but once written, they can be reused with minimal overhead. Quizzes were administered and scored in Qualtrics, which provided students with immediate feedback including their selections and the answer key. All students received a brief tutorial and a non-graded practice quiz to familiarize them with the format.

In the original JMCQ trials [5], post-assessment discussion was tied to recovery questions embedded in the next midterm exam. In that work, the authors indicated that this structure encouraged students to restudy, it increased exam length and offered limited point recovery relative to the effort required. In the current study, those high-stakes recovery opportunities were replaced with multiple short, low-stakes quizzes.

3.2 SPEIR with per-quiz recovery

The first remediation model was used in the introductory course, where students typically benefit from closer guidance. In this version, students gained access to a recovery quiz only after discussing their first-round results with the instructor. The process proceeded as follows:

1. Students completed a first-round quiz. All quizzes used the JMCQ format except in the Fall 2024 A2 section, where traditional MCQs were used as a control.

2. Students scoring below a threshold (typically 80–85%) could opt to take a recovery quiz. Before doing so, they were required to meet with the instructor either one-on-one or in small groups during office hours to review their results. The purpose of this discussion was to diagnose the source of errors and clarify misconceptions.
 - (a) During the discussion, both student and instructor had access to the student’s first-round responses and the quiz answer key, which students were expected to review beforehand.
 - (b) For each incorrect or incorrectly justified question, the instructor asked the student to explain (a) their original reasoning and (b) how their thinking changed after reviewing the correct answer. These conversations aimed to identify whether mistakes stemmed from conceptual errors, flawed reasoning, or misinterpretations, and provided space for students to ask clarifying questions.
3. After the discussion, students received access to a recovery quiz containing items closely aligned with those in the first round. Grades from the original and recovery quizzes were averaged and capped at the threshold to ensure fairness for students not requiring remediation. This variation has the effect of bringing to office hours those students who likely need it the most, and remains within the committed time of the instructor.

3.3 SPEIR without per-quiz recovery

The second implementation of SPEIR was used in the theory course, where students were expected to work with greater independence. In this version, the post-assessment discussions with the instructor served as the final step of the remediation process. Rather than completing recovery quizzes, students were allowed to revise and resubmit the homework assignments at the end of the semester. The intention was that insights gained from homework grading and post-quiz discussions would enable students to produce clearer, more accurate, and better-reasoned revisions. As in the introductory course, quiz results were analyzed to evaluate the effectiveness of SPEIR across courses, semesters, and instructors. These data allowed us to compare how the workflow operated under differing expectations for student self-regulation and support.

4 Results

This section reports comparisons between SPEIR and control conditions, differences across instructors, and patterns in correctness and justification performance. We also summarize student attitudes toward the SPEIR workflow.

4.1 MCQ vs Justification

Each JMCQ quiz produces a correctness score (M) and a justification score (J), averaged to obtain the overall grade (G). Figure 1 and Table 2 summarize results for the three programming sections in Fall 2024, where A2 served as the MCQ-only control.

Across all JMCQ sections, justification scores consistently trend lower than correctness scores. In a minority of quizzes, the difference between M and J scores is statistically significant (Wilcoxon signed-rank test), but even when not significant, the consistent downward shift in J highlights that many students can select the correct answer while relying on incomplete or incorrect reasoning. This contrasts with the MCQ-only control section, where high correctness scores provide no visibility into underlying misconceptions.

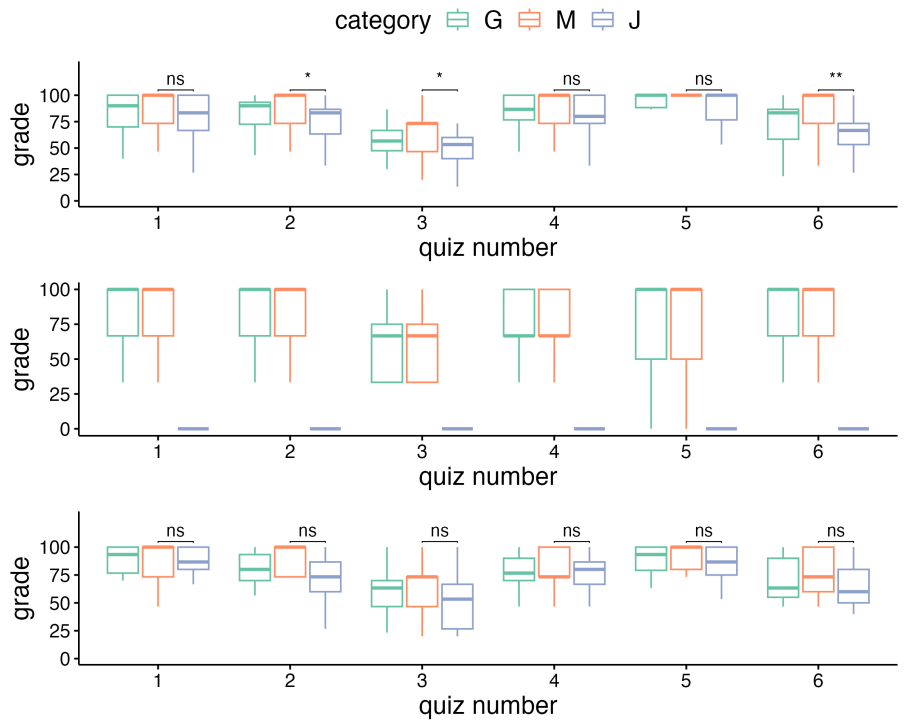


Figure 1: Prog F24 A1 (top), A2 (middle), and B3 (bottom): Correctness vs Justification Grades per quiz

Prof-Sec	Quiz	Treatment	G	M	J	p
A1	1	jmcq	90.00	100.00	83.33	0.0560
A1	2	jmcq	90.00	100.00	83.33	0.0250
A1	3	jmcq	56.67	73.33	53.33	0.0230
A1	4	jmcq	86.67	100.00	80.00	0.1810
A1	5	jmcq	100.00	100.00	100.00	0.0570
A1	6	jmcq	83.33	100.00	66.67	0.0020
A2	1	mcq	100.00	100.00	–	–
A2	2	mcq	100.00	100.00	–	–
A2	3	mcq	66.67	66.67	–	–
A2	4	mcq	66.67	66.67	–	–
A2	5	mcq	100.00	100.00	–	–
A2	6	mcq	100.00	100.00	–	–
B3	1	jmcq	93.33	100.00	86.67	0.2350
B3	2	jmcq	80.00	100.00	73.33	0.0590
B3	3	jmcq	63.33	73.33	53.33	0.0510
B3	4	jmcq	76.67	73.33	80.00	0.7970
B3	5	jmcq	93.33	100.00	86.67	0.0900
B3	6	jmcq	63.33	73.33	60.00	0.3970

Table 2: Stats summary for all programming sections. The p represents the probability that the M and J distributions are similar.

4.2 Between-Section Comparisons

Instructor Consistency. The two SPEIR sections (A1 and B3) show highly similar correctness and justification patterns, suggesting that the workflow is robust to instructor variation. Theory-course results show the same pattern, with the exception of a level shift in Spring 2024.

Control vs. SPEIR. Correctness-only scores from the control section (A2) were compared to correctness scores from the SPEIR sections using a Kruskal–Wallis test. No significant differences in median correctness were detected, reinforcing that correctness alone does not meaningfully distinguish treatments.

4.3 Per-Question Analysis

To better understand underlying reasoning, we analyzed the proportion of fully correct answers per question ($M_{\%}$ for correctness; $J_{\%}$ for justification). Table 3 reveals two patterns in the results for Fall 2024. First, justification proportions ($J_{\%}$) are consistently lower than correctness proportions ($M_{\%}$), reinforcing that many apparently correct answers reflect incomplete understanding or guessing. Second, the MCQ-only control section showed a general tendency to obtain

lower per-question correctness than in the SPEIR sections.

Prof-Sec	Quiz	NQs	M_{num}	J_{num}	$M_{\%}$	$J_{\%}$
A1	1	90	75	50	83.33	55.56
A1	2	84	69	47	82.14	55.95
A1	3	90	49	10	54.44	11.11
A1	4	81	67	52	82.72	64.20
A1	5	90	83	68	92.22	75.56
A1	6	78	61	34	78.21	43.59
A2	1	84	53	–	81.0	–
A2	2	84	48	–	79.8	–
A2	3	84	25	–	63.1	–
A2	4	81	34	–	74.1	–
A2	5	69	36	–	75.4	–
A2	6	57	33	–	75.4	–
B3	1	60	53	44	88.33	73.33
B3	2	60	47	29	78.33	48.33
B3	3	63	38	18	60.32	28.57
B3	4	63	45	36	71.43	57.14
B3	5	54	49	39	90.74	72.22
B3	6	21	15	11	71.43	52.38

Table 3: Proportion of fully correct answers in the programming course. The control (A2) shows lower proportions of correctness per question.

The consistently lower correctness proportions on the final quiz may partly reflect the **S/U-effect**: once students received accurate grade projections, those already assured of a “Satisfactory” grade in the course sometimes treated remaining quizzes less seriously or skipped them entirely.

4.4 First-Round vs. Recovery Quizzes

Across all semesters of the programming course, paired comparisons show consistent improvement on recovery quizzes for every section except one. Although expected, these gains do not necessarily reflect improved understanding: in the MCQ-only control, many correct answers on both attempts still masked misconceptions. In SPEIR sections, improvement in justification scores provides a stronger indication of conceptual change.

4.5 Student Attitudes

A five-point Likert-scale survey administered in Fall 2024 measured student attitudes toward structure, fairness, anxiety, and workload. A sample of the

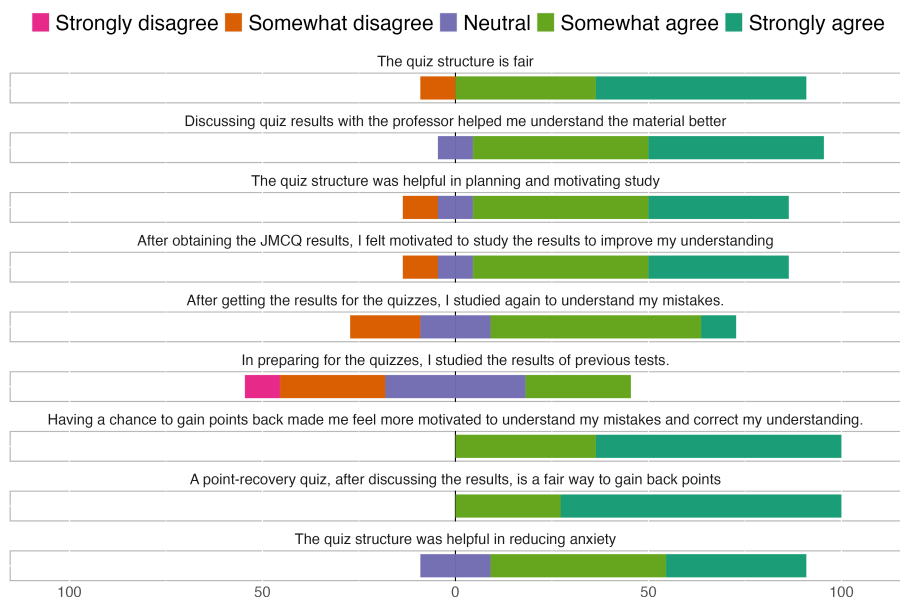


Figure 2: Survey responses for the Programming course in Fall 2024

most relevant questions is shown in Figure 2. Students expressed strongly positive views of the SPEIR workflow. They described the quizzes as low-stakes, the discussions as helpful rather than burdensome, and the overall structure as fair and clear. Notably, many students reported not restudying quiz results on their own, underscoring the importance of the guided “just-in-time” discussions that SPEIR provides.

5 Discussion

In addition to the quantitative results, the implementation of SPEIR yielded qualitative observations that clarify how and why the workflow functions in practice. These insights, together with the empirical findings, illuminate student reasoning, instructor experience, and the affordances of reasoning-centered formative assessment. The sections that follow interpret our results through the lens of the five research questions that guided the study.

5.1 RQ1: Does SPEIR help reveal errors, misconceptions, and misunderstandings?

Our results show that SPEIR reliably surfaces gaps between performance and understanding. Across semesters and course levels, correctness scores were consistently higher than justification scores, replicating prior findings on JMCQs [5]. These discrepancies indicate that correctness alone substantially overestimates student understanding. Per-question analyses, when taken in conjunction with the MCQ-only over-estimation of understanding, revealed that SPEIR sections demonstrated a tendency to show higher rates of fully correct reasoning than the MCQ control section, highlighting the diagnostic value of requiring students to articulate (or select) justifications. Moreover, instructors observed that the structured justification component made misconceptions easier to identify and categorize during follow-up conversations. The combination of student responses, paired justifications, and an answer key provided a coherent framework for diagnosing whether an error stemmed from a misunderstanding of concepts, flawed reasoning, or misinterpretation of question wording.

5.2 RQ2: Does SPEIR help structure productive follow-up discussions?

Both in-class and office-hours discussions benefitted from the structure provided by JMCQs. In-class reviews typically took only 10–15 minutes, while office-hours conversations generally lasted 5–10 minutes per student or group. In both contexts, instructors reported that the availability of student-selected justifications allowed them to target misconceptions directly and maintain consistency across sections.

Students' behavior also reflected the utility of the structured follow-up. Although survey results indicated that students did not routinely restudy assessment results on their own, they consistently reviewed their answers prior to meeting with the instructor to gain access to recovery opportunities. This suggests that SPEIR's follow-up mechanism acts as a just-in-time motivator for meaningful reflection.

5.3 RQ3: Is SPEIR scalable and transferable to other subjects or instructors?

Evidence across 10 sections taught by three instructors suggests strong scalability. JMCQ items require an initial authoring effort, but instructors noted that questions were reusable across semesters with minimal overhead. An additional implementation using Moodle's matching-columns tool yielded com-

parable patterns, suggesting that SPEIR's correctness/justification structure can be delivered through multiple platforms.

SPEIR's workflow also proved robust across instructors: results from different sections followed similar trends, with correctness and justification scores displaying parallel patterns even when instructional style varied. One exception appeared in a Spring 2024 theory section, where mismatches between lecture terminology and justification phrasing likely contributed to lower scores. This underscores the importance of close alignment between instructional language and JMCQ construction but does not undermine overall transferability.

5.4 RQ4: Does SPEIR support student engagement and motivation?

Survey results indicate that students perceived SPEIR as clear, fair, and helpful to their learning. In particular, students responded positively to the recovery quiz option. Many viewed it not merely as a chance to regain points but as an incentive to restudy targeted material. Instructors observed that students appeared more engaged and reflective during follow-up discussions than during traditional MCQ reviews. A notable finding is the motivational leverage embedded in the workflow: although students rarely revisited quiz results unprompted, they consistently prepared for their recovery discussions, revealing that SPEIR's structure supports engagement at the moment when it is most pedagogically impactful.

5.5 RQ5: Does SPEIR help reduce student anxiety?

Students overwhelmingly reported that the SPEIR workflow felt low-stakes and low-anxiety. Discussing results with an instructor before attempting recovery quizzes helped reframe assessments as learning tools rather than evaluative hurdles. Instructors likewise noted that this environment encouraged struggling students to seek help earlier, contributing to a narrowing of performance gaps. Recovery quiz gains support this interpretation: most students improved substantially between attempts. While gains in the MCQ-only control may reflect memorization, improvements in justification scores within SPEIR sections more strongly indicate conceptual refinement, which in turn reduces uncertainty and anxiety around future assessments.

5.6 Limitations and Future Work

SPEIR depends on careful alignment between the language used in instruction and the phrasing of justification statements. When this alignment breaks down, as seen in one theory section, students may struggle for reasons unrelated

to conceptual understanding. Developing guidelines or tools for constructing consistent JMCQs may help address this issue.

Our qualitative observations provided valuable context, but the study did not employ a formal qualitative analysis framework. Future work incorporating interview protocols, think-aloud studies, or coded written reflections would deepen our understanding of how students reason through justification options and how instructors interpret diagnostic information.

This study was conducted at a single institution across two computing courses; broader replication is needed to evaluate generalizability. Investigating SPEIR in other STEM and non-STEM disciplines, in courses with different class sizes, and across institutions with varying student demographics would help determine its broader applicability.

Finally, there is an opportunity to explore partial automation of the workflow. Systems that help generate justification sets, analyze common reasoning patterns, or scaffold student reflection could reduce instructor workload and support adoption at scale.

6 Conclusion

Taken together, these findings demonstrate that SPEIR provides a robust, flexible framework for reasoning-centered formative assessment. It reveals misconceptions that correctness-only approaches miss, structures efficient and productive instructor-student conversations, scales across courses and instructors, and offers motivational and affective benefits through supportive recovery.

6.1 Auxiliary Materials

For a detailed example of how to develop JMCQ questions, and comprehensive versions of the tables and student responses, please visit the following site: [SPEIR Auxiliary Materials](#).

7 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2141772 and 2141705. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- [1] Dinesh K Badyal et al. “Impact of immediate feedback on the learning of medical students in pharmacology”. In: *Journal of Advances in Medical Education & Professionalism* 7.1 (2019), p. 1.
- [2] Lais Tono Cardozo et al. “Active learning methodology, associated to formative assessment, improved cardiac physiology knowledge and decreased pre-test stress and anxiety”. In: *Frontiers in physiology* 14 (2023), p. 1261199.
- [3] Jerrell C Cassady and Betty E Gridley. “The effects of online formative and summative assessment on test anxiety and performance”. In: *The Journal of Technology, Learning and Assessment* 4.1 (2005).
- [4] Mercedes Douglas, Juliette Wilson, and Sean Ennis. “Multiple-choice question tests: a convenient, flexible and effective learning tool? A case study”. In: *Innovations in Education and Teaching International* 49.2 (2012), pp. 111–121.
- [5] Pablo Frank-Bolton et al. “The Justification Effect on Two-Tier Multiple-Choice Exams”. In: *2024 ASEE Annual Conference & Exposition*. 2024.
- [6] Joanna K Hubbard, Macy A Potts, and Brian A Couch. “How question types reveal student thinking: An experimental comparison of multiple-true-false and free-response formats”. In: *CBE—Life Sciences Education* 16.2 (2017), ar26.
- [7] Martijn Leenknecht et al. “Formative assessment as practice: The role of students’ motivation”. In: *Assessment & Evaluation in Higher Education* 46.2 (2021), pp. 236–255.
- [8] Suzanne McCallum and Margaret M Milner. “The effectiveness of formative assessment: student views and staff reflections”. In: *Assessment & Evaluation in Higher Education* 46.1 (2021), pp. 1–16.
- [9] Yizhou Qian and James Lehman. “Students’ misconceptions and other difficulties in introductory programming: A literature review”. In: *ACM Transactions on Computing Education (TOCE)* 18.1 (2017), pp. 1–24.
- [10] Mohammad Ali Rostaminezhad. “Students’ perceptions of the strengths and limitations of electronic tests focusing on instant feedback”. In: *Journal of Information Technology Education. Research* 18 (2019), p. 59.
- [11] Jagadeeswaran Thangaraj, Monica Ward, and Fiona O’Riordan. “A Systematic Review of Formative Assessment to Support Students Learning Computer Programming”. In: *4th International Computer Programming Education Conference (ICPEC 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2023, pp. 7–1.

- [12] Miki K Tomita. *Examining the influence of formative assessment on conceptual accumulation and conceptual change*. Stanford University, 2009.
- [13] David F Treagust. “Development and use of diagnostic tests to evaluate students’ misconceptions in science”. In: *International journal of science education* 10.2 (1988), pp. 159–169.
- [14] Naomi E Winstone and Edd Pitt. “Approaches to feedback on examination performance: research, policy, and practice”. In: *Assessment & Evaluation in Higher Education* (2025), pp. 1–21.
- [15] Chunliang Yang et al. “Testing (quizzing) boosts classroom learning: A systematic and meta-analytic review.” In: *Psychological bulletin* 147.4 (2021), p. 399.

Teaching Software Development in the GenAI Era: Experiences and Course Design*

Weidong Liao
Computer Science, Mathematics and Engineering
College of STEM
Shepherd University
Shepherdstown, WV 25443
wliao@shepherd.edu

Abstract

This paper examines how Generative AI coding tools impact computer science education and describes instructional experiments conducted across several courses that inform the design of a new Special Topics course, *Software Development in the GenAI Era*. The newly designed course introduces students to the evolving landscape of AI-assisted development through five modules: The Stochastic Engine, Prompt & Context Engineering, Vibe Coding, Agentic Coding, and Specification-Driven Development. Drawing from classroom experiences, the paper highlights how students respond to AI tools, where they benefit most, and where they struggle with the verification and documentation practices required for responsible AI use. The goal is to offer a practical, evidence-based framework for preparing students to become architects and supervisors of AI-driven software systems, equipped with the foundational CS knowledge and modern AI-native skills needed in today's rapidly evolving development environment.

1 Introduction

The emergence of Generative AI (GenAI) and Large Language Models (LLMs) is transforming software development at a pace that challenges long-standing

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

assumptions in computer science education. Tasks that once required substantial programming expertise—prototyping interfaces, generating full applications, orchestrating workflows, and even coordinating autonomous agents—can now be initiated through natural language descriptions. As AI tools evolve from snippet generators to full collaborators capable of planning, reasoning, and executing code, the role of the developer is shifting from manual coding toward supervision, verification, and architectural design.

This rapid transformation raises important questions for educators. How should introductory programming adapt when students can build functional prototypes from a single prompt? What competencies do graduates need to work effectively with agentic systems and specification-driven pipelines? And how can programs preserve the rigor of traditional CS foundations while preparing students for AI-native development workflows?

This paper responds to these questions by examining recent instructional experiments and presenting the design of a new Special Topics course, *Software Development in the GenAI Era*. Drawing on classroom experiences, the paper outlines a structured curriculum grounded in the evolving eras of AI-powered development—Vibe Coding, Agentic Coding, Context Engineering, and Specification-Driven Development. The goal is to provide a practical, evidence-informed framework for teaching software engineering in a landscape where AI is no longer optional, but an integral part of the development process.

2 Background and Motivation

The past few years have brought remarkable advances in Generative AI, reshaping software engineering in ways increasingly difficult to ignore. Tools that once provided small conveniences—such as generating helper functions or boilerplate—have evolved into systems capable of producing full prototypes, reasoning through multi-step tasks, and even coordinating with other AI agents. As a result, software development is no longer defined solely by writing code; it now involves guiding, verifying, and collaborating with intelligent systems that participate in the development process.

For educators, these shifts raise important questions. What does it mean to teach software development when students can generate a working program from a single prompt? How do we prepare graduates for a world where developers must supervise AI agents, evaluate their outputs, and design the specifications that govern them? Understanding how AI-powered development has evolved—and how quickly it continues to evolve—is essential for contextualizing both our teaching experiences and the design of course modules described in this paper. It also helps frame the changing expectations and emerging practices that students will encounter as they enter the profession.

2.1 The Snippet Era: AI as a Code Generator

The earliest stage of AI-assisted programming was modest in scope but still exciting for developers (and CS students) encountering these tools for the first time. Systems like GPT-3 and Codex could produce small, useful pieces of code—functions, loops, class definitions, API calls, or short documentation snippets—based on short prompts. These models behaved much like intelligent autocomplete: they could respond quickly and helpfully, but they had no real understanding of the broader system a developer was building.

During this period, AI tools served mainly as time-savers for repetitive or boilerplate tasks. Developers still handled everything that required deeper reasoning: integrating components, making architectural decisions, debugging unpredictable behavior, and ensuring the final product worked correctly. While the technology accelerated typing and reduced routine work, it did not fundamentally change the structure of the software development lifecycle. Developers remained firmly in the driver’s seat—AI simply made a few parts of the ride smoother.

2.2 The Vibe Coding Era: AI as a Rapid Prototyping Partner

As language models grew more capable and tools like Cursor, Claude Code, and AI-enhanced IDEs became widely available, developers began using AI for far more than small code snippets. Instead of asking AI to fill in missing lines, developers increasingly relied on it to generate entire working prototypes from natural language descriptions. This shift—often described as *Vibe Coding*—moved software development toward an intent-driven process.

In this era, a developer could describe a goal at a high level—“Build a dashboard that ingests a CSV and visualizes monthly revenue”—and AI tools would generate a complete, runnable project that met that specification. The role of the human shifted from writing the code themselves to *reviewing, correcting, and refining* AI-generated output.

While Vibe Coding dramatically accelerated early phases of development and opened creative possibilities, it was not without challenges. Developers frequently encountered hallucinations, subtle logic errors, and architectural inconsistencies that required significant oversight. In many ways, Vibe Coding expanded what developers could accomplish quickly, but also demanded a new level of vigilance and critical evaluation.

2.3 The Agentic Coding Era: AI as an Autonomous Solver

The next major leap occurred when AI systems gained the ability not just to generate code, but to *act*. With the introduction of agentic frameworks such as the Model Context Protocol (MCP), LangChain, and LangGraph, models

could perform multi-step tasks, call external tools, run code, fetch data, and reason through their own plans.

In this era, AI systems began to behave less like sophisticated autocomplete and more like autonomous problem solvers. They could:

- **Use Tools:** call APIs, query databases, read and write files, or even initiate deployment steps.
- **Plan:** break down complex tasks into smaller steps before acting.
- **Self-iterate:** evaluate their own outputs and revise them.
- **Collaborate:** work in coordinated multi-agent teams—planner, developer, reviewer, tester—each specializing in a different part of the workflow.

For developers, this changed the nature of their work. Instead of directly producing every component, they increasingly served as supervisors—setting constraints, monitoring AI behavior, diagnosing failures, and ensuring that agents stayed aligned with the project’s goals. Agentic Coding did not replace the developer, but it redefined the developer’s role into something closer to a system architect and quality overseer.

2.4 The Specification-Driven Era: AI as a Full Development Pipeline

As developers pushed beyond rapid prototyping (Vibe Coding) and began supervising autonomous agents (Agentic Coding), a new challenge emerged: although AI systems could now take on far more responsibility, they often did so in unpredictable ways. Hallucinations, inconsistent design decisions, and architectural drift still occurred—sometimes amplified by the complexity of agentic workflows. This created a natural need for a more disciplined framework to ensure reliability, safety, and coherence.

This need led to the most recent and rapidly advancing stage of AI-powered development: **Specification-Driven Development (SDD)**. In this approach, the developer focuses on writing clear, structured specifications—requirements, JSON schemas, state machines, acceptance criteria—that define the behavior and architecture of the system. Instead of coding directly, the human provides the “blueprint,” and the AI generates the implementation, tests, and supporting artifacts from that blueprint.

In an SDD workflow:

- The specification becomes the single source of truth.
- AI agents generate architecture, code, and tests directly from the specification.
- Verification is continuous and automated, ensuring that outputs remain aligned with the spec.

- Provenance and traceability are built in, making it possible to see how each component was derived.

This paradigm forms the foundation of a new branch of software engineering, named **agentic engineering** [1][2]. It shifts software development away from writing code line by line. Instead, developers design and govern intelligent systems that carry out implementation tasks autonomously under human-defined constraints.

In summary, the four eras, Snippet, Vibe, Agentic, and Specification-Driven, reflect a steady movement from manual coding toward intent expression, architectural reasoning, verification, and orchestration. Developers now collaborate with probabilistic systems that can reason about requirements and execute complex workflows, fundamentally reshaping the nature of software engineering.

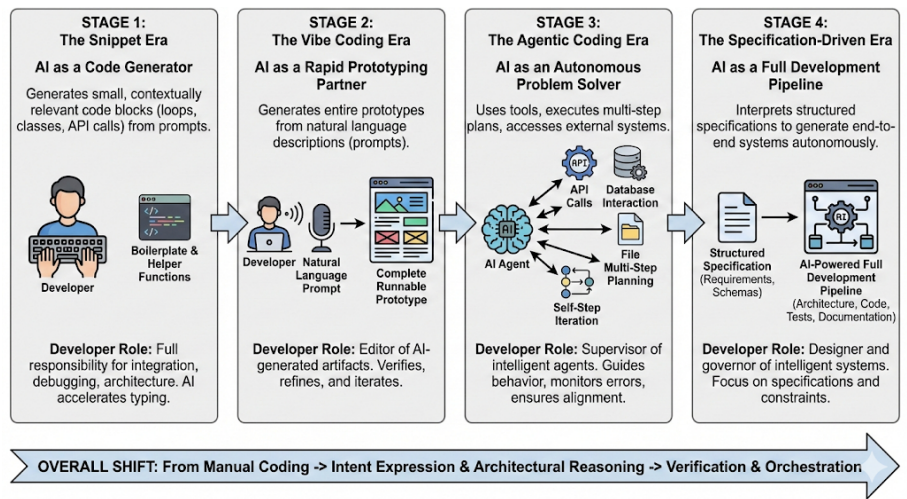


Figure 1: GenAI Shifting Software Development (Created with assistance of AI)

As AI capabilities continue to advance, computer science education must evolve alongside them. Preparing students for the future requires teaching not only how to write code, but how to design specifications, supervise autonomous agents, evaluate model-generated outputs, and construct reliable, verifiable AI-driven development ecosystems.

3 Adopting AI Coding in Project Assignments

In recent semesters, the author has experimented with introducing AI-powered development workflows into several existing classes. These early trials helped illuminate how students at different stages of the curriculum respond to AI tools—how they balance automation with learning goals, how they approach verification and documentation, and how AI changes their workflow and mindset. The observations from these courses provided valuable insights and directly shaped the development of the new course presented later in this paper.

Two implementations, in particular, illustrate how students' experiences and attitudes vary depending on their academic preparation and confidence as emerging software developers.

3.1 Web Programming Fundamentals (Sophomore/Junior Level)

In the Web Programming Fundamentals course, students work in teams on a half-semester project that walks through the full lifecycle of web development: from early ideation and requirements gathering, to wireframing and UX/UI design, to implementation, testing, and deployment. Because students at this level are still solidifying core concepts in front-end and back-end development, AI tools were intentionally withheld at the start of the project.

Teams were required to design the system architecture and build a working prototype manually, giving them firsthand experience with the challenges of implementing a full web application. Only after completing and submitting their initial prototype were students given two additional weeks to refactor and enhance their work using AI coding tools.

This staged approach helped students appreciate the contrast between traditional development and AI-assisted workflows. It also ensured that they acquired foundational knowledge—HTML/CSS structure, asynchronous logic, debugging practices—before relying on AI for productivity gains.

3.2 Senior Capstone Course (Senior Level)

In the Senior Capstone course, students had considerably more autonomy in how they approached their final projects. Unlike the Web Programming course, where AI tools were intentionally withheld early on, Capstone teams were allowed to use Generative AI from the beginning—provided they committed to doing so in a responsible and well-documented way. Any team wishing to incorporate AI tools had to outline their intended workflow in their formal project proposal and justify how AI would be integrated into the broader software engineering process.

To support responsible use, teams adopting AI were required to design a systematic and traceable development process that would ensure the final product met production-level expectations. This included:

- Writing detailed product specifications that captured functional requirements, constraints, and edge cases.
- Documenting system architecture and design rationale, rather than relying solely on AI-generated structures.
- Following a test-driven or verification-first approach, acknowledging the nondeterministic nature of AI-generated code.
- Maintaining comprehensive documentation of prompts, system interactions, and iterative refinements, including specifications, Markdown files, and logs.

In this environment, AI was framed not as a shortcut to bypass foundational skills, but as a tool that needed to be **directed, monitored, and evaluated**. Students quickly discovered that using AI effectively required as much discipline as traditional development—sometimes more—because ensuring correctness and coherence across AI-generated components remained their responsibility.

3.3 Student Experiences and Observations

Across both courses, a clear pattern emerged in how students engaged with AI-assisted development.

Students in the **Web Programming Fundamentals** course responded with genuine enthusiasm once AI tools were introduced. After spending several weeks building their projects manually, they immediately recognized the benefits of AI-supported refactoring. Many remarked that AI allowed them to try ideas they would not have attempted on their own, and the noticeable boost in productivity made the development process feel more creative and less intimidating. For these students, AI felt like an empowering extension of their abilities—something that helped them build confidence rather than replace foundational skills.

Reactions in the **Senior Capstone** course were more mixed. Only about one-third of the teams chose to integrate Generative AI throughout their full project lifecycle. Some students worried that relying too heavily on AI might limit their opportunity to strengthen their own technical skills before graduation. Others acknowledged the potential of AI tools but were hesitant to adopt them because of the additional work required: documenting prompts, validating outputs, and maintaining a detailed record of design and verification steps. For these students, the overhead associated with responsible AI use felt burdensome compared to the familiarity of traditional coding workflows.

Together, these observations highlight an important pedagogical insight: *Students are eager to leverage the speed and creativity that AI tools offer, but many are less comfortable with the rigorous documentation, specification writing, and verification processes that must accompany responsible AI-assisted development.*

These early lessons directly influenced the design of the Special Topics course, *Software Development in the GenAI Era*, where Vibe Coding, agentic workflows, and Specification-Driven Development are taught explicitly—not as optional enhancements, but as core practices required for modern software engineering.

4 Course Design: *Software Development in the GenAI Era*

The Special Topics course *Software Development in the GenAI Era* was created in direct response to the rapid and ongoing transformation of software engineering. As AI tools evolved from simple code-suggestion systems to generators of full prototypes, autonomous agents, and specification-driven pipelines, it became clear that traditional computer science curricula were no longer sufficient on their own. Students today must graduate with skills that go far beyond memorizing syntax—they must know how to design architectures, guide intelligent agents, manage context, verify nondeterministic outputs, and write precise specifications that govern AI-driven systems.

This course, first to be offered in Spring 2026, represents an intentional effort to prepare students for this emerging landscape. Although outcomes from the first offering are still in progress, the structure and pedagogical approach have already provided a strong foundation for rethinking how software development should be taught in the age of Generative AI.

To help students build the competencies needed for AI-native engineering, the course is organized around a structured progression through the major eras of AI-driven development described in Section 2:

- The Stochastic Engine,
- Prompt & Context Engineering,
- Vibe Coding and Agentic Coding,
- Specification-Driven Development (SDD).

Each module reflects a stage in the evolution of modern AI tooling, allowing students to experience firsthand how software development has expanded from manual coding to intent-driven and agent-driven workflows. Throughout the course, LLMs are treated not as passive “assistants,” but as **probabilistic**

components within a deterministic software stack—systems that must be understood, constrained, and verified.

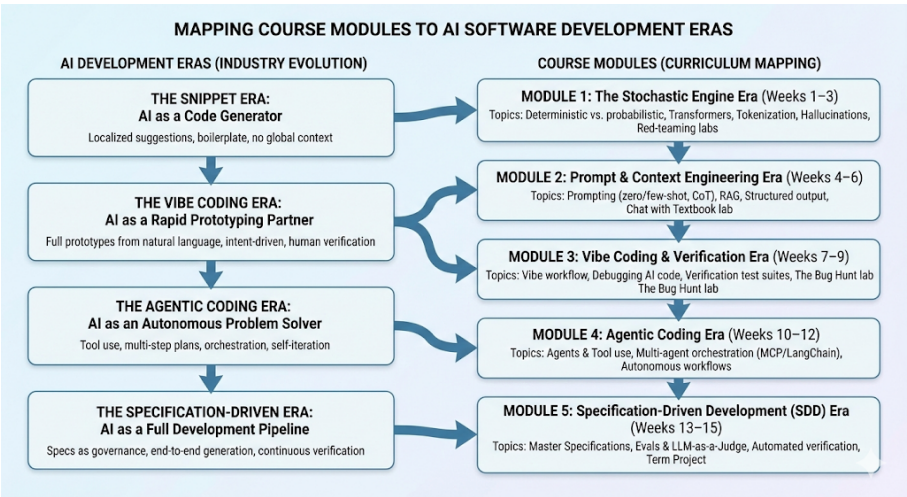


Figure 2: Curriculum Mapping (Created with assistance of AI)

4.1 Module 1 — The Stochastic Engine Era: Understanding the AI Developer (Weeks 1–3)

This first module helps students build a conceptual foundation for working with Large Language Models. Before they can meaningfully collaborate with AI tools, students need to understand what LLMs are doing “under the hood” and why their outputs can be both powerful and unpredictable. This phase mirrors the early “Snippet Era” of AI development, when models were first introduced as probabilistic code generators.

To develop this understanding, students explore topics such as:

- The difference between deterministic and probabilistic computing
- How transformer architectures and attention mechanisms shape model behavior
- How tokenization, context windows, and cost constraints affect prompts
- Why hallucinations occur, and how vulnerabilities like jailbreaking and prompt injection arise

Hands-on labs include **red-teaming exercises**, where students intentionally try to break or manipulate models. This gives them firsthand experience

with the limitations and risks of LLMs—an essential skill for anyone working with modern agentic systems.

4.2 Module 2 — Prompt & Context Engineering Era (Weeks 4–6)

Building on their understanding of the underlying model, students next learn how to guide AI systems effectively through prompting and context design. This module represents the shift toward **intent-driven development**, where the quality of the developer’s instructions becomes just as important as the code itself.

Students practice a range of prompting techniques, including:

- Zero-shot, few-shot, and chain-of-thought prompting
- Designing system prompts and persona-based directives
- Constructing Retrieval-Augmented Generation (RAG) systems using embeddings and vector databases
- Enforcing reliable outputs through JSON schemas and structured formats

The laboratory component emphasizes **context engineering**, showing students how grounding an LLM with curated external information can dramatically reduce hallucinations. A key assignment in this module is “*Chat with Your Textbook*,” where students build a RAG pipeline that allows an LLM to answer questions based on course readings. This activity illustrates how structured context—not clever prompting alone—is the foundation of reliable AI behavior.

4.3 Module 3 — Vibe Coding & Verification Era (Weeks 7–9)

This module introduces students to the heart of **Vibe Coding**—rapid prototyping supported by AI—while emphasizing that speed must always be balanced with careful human oversight. Students quickly learn that while AI can generate substantial amounts of code, it cannot be trusted blindly. The goal of this module is to help students experience the excitement of accelerated development while building the verification skills needed to ensure correctness and reliability.

Topics in this module include:

- Understanding the Vibe Coding workflow, including “*tab-tab-tab*” speed coding in AI-enabled editors
- Reading, interpreting, and debugging AI-generated code
- Designing verification-oriented test suites to catch subtle errors
- Refactoring legacy or “spaghetti code” into well-structured, modern architectures

A highlight of this module is the **Bug Hunt** lab, where students inspect AI-generated programs that contain intentional (and occasionally surprising) errors. Students must identify, isolate, and correct these issues—reinforcing the notion that even when AI assists with writing the code, the human developer remains the architect, evaluator, and ultimate decision-maker.

4.4 Module 4 — Agentic Coding Era (Weeks 10–12)

In this module, students move beyond generating code with an AI and begin working with **agentic systems**—AI models that can plan, make decisions, call external tools, and perform multi-step tasks. This reflects the industry’s shift toward Agentic Development, where AI tools no longer act merely as code generators but as active participants in the development workflow.

Students learn to design and build systems such as:

- **Tool-using agents**, capable of calling APIs, querying databases, or manipulating local files
- **Multi-agent workflows**, using frameworks like MCP, LangChain, and LangGraph, where agents play specialized roles (planner, coder, reviewer, tester)
- **End-to-end pipelines** that fetch data, organize it, make decisions, and generate outputs or visualizations through autonomous execution

By “giving AI hands,” students experience what it means for an AI to take action rather than simply provide code. This module helps students transition from thinking about “what code should I write?” to “what workflow should I design, and how do I supervise the agents that execute it?” It encourages students to operate at a higher level of abstraction—specifying behavior, validating decisions, and ensuring alignment between agent actions and project goals.

4.5 Module 5 — Specification-Driven Development (SDD) Era (Weeks 13–15)

The final module brings students into the newest and most forward-looking stage of AI-powered software development: **Specification-Driven Development (SDD)**. At this stage of the course, students have experienced how AI can generate code, assist with prototyping, and operate as an autonomous agent. SDD asks them to step into an even more advanced role—designing the *rules* and *structures* that AI systems must follow.

Rather than writing the code themselves, students learn to write the **Master Specifications** that function as the governing source of truth for a project. These specifications define the system’s data models, decision logic, behavior,

and constraints. The AI then uses these specifications to generate architecture, code, tests, and documentation.

During this module, students work on:

- Crafting detailed **Master Specifications** that guide the entire development pipeline
- Designing **JSON schemas**, data models, acceptance criteria, and edge-case rules
- Building **Evals** and experimenting with **LLM-as-a-Judge** for nondeterministic testing
- Architecting automated verification systems to validate AI-generated components
- Completing and presenting final projects built largely through AI, but governed by human-written specifications

The capstone projects produced in this module emphasize four core artifacts—**Specification, Verification Suite, Interaction Log, and Prototype**—each reflecting a crucial skill in modern agentic engineering. Students not only build functioning software systems but also demonstrate how they supervised, verified, and refined the AI throughout the development lifecycle.

Bringing the Modules Together

Taken together, the five modules trace the same historical progression seen in industry: *from stochastic engines, to prompt & context engineering, to vibe-driven prototyping, to autonomous agentic systems, and finally to specification-driven pipelines.*

By experiencing this sequence firsthand, students learn to think like architects of AI-native systems rather than traditional coders. They would leave the course prepared to design specifications, orchestrate intelligent agents, verify AI-generated outputs, and deploy modern software responsibly—skills increasingly essential in today’s rapidly evolving computational landscape.

5 Discussion

The integration of Generative AI and agentic workflows into software engineering brings both exciting opportunities and serious challenges for educators. As these tools become more capable and more deeply embedded in real-world development practices, they force us to reconsider long-standing assumptions in computer science education. Questions that once seemed purely hypothetical are now urgent:

- What should introductory programming look like when students can generate a functional prototype with a single prompt?
- How should institutions prepare students for development workflows that involve coordinating and supervising AI agents rather than writing every line of code themselves?
- How do we ensure that students retain strong foundational skills while also benefiting from the productivity gains that AI tools provide?

These questions do not mean that traditional CS fundamentals are obsolete. In fact, core concepts—computational thinking, algorithms, data structures, complexity analysis, systems design, and software engineering principles—are more essential than ever. These fundamentals provide the conceptual grounding students need to understand, critique, and guide AI-generated systems. Without them, students cannot reliably supervise AI agents, diagnose errors, or recognize when the AI has taken a design in an unintended direction.

What *has* changed is the role these fundamentals play. Historically, we taught algorithms so students could implement them by hand; we taught data structures so students could build them manually; we taught control flow so students could write loops and conditionals. In an AI-native development environment, these same concepts matter—but for different reasons. Students now need to understand fundamentals deeply enough to:

- Recognize when an AI-generated algorithm is inefficient or incorrect
- Improve or reject architectural choices made by the model
- Specify the appropriate data structures and performance requirements up front
- Interpret the behavior of multi-agent systems working across several components
- Diagnose subtle or emergent errors in AI-generated code

In short, fundamentals shift from being primarily about *writing* code to being about *understanding and supervising* it. Developers must now manage nondeterministic systems whose internal reasoning is not always transparent, and this requires a stronger—not weaker—foundation in computer science.

Our classroom experiments reinforce this point. Students can strengthen their understanding of core concepts through AI-native practices: debugging AI-generated code deepens comprehension of logic and control flow; designing JSON schemas or state machines reinforces data modeling; writing verification suites sharpens understanding of invariants and edge cases; and orchestrating multi-agent systems promotes architectural thinking.

These experiences suggest several avenues for future research and curriculum development:

- **Reimagining introductory programming:** Should we teach specifications and algorithms before syntax?
- **Developing new assessment models:** How can we measure understanding when AI assistance is ubiquitous?
- **Building AI-native teaching tools:** How can agents help instructors detect misconceptions or provide personalized support?
- **Studying long-term skill retention:** How do we ensure foundational knowledge remains strong in AI-rich learning environments?

Ultimately, AI does not reduce the need for human expertise—it expands it. Students must learn not only how to *use* AI tools, but how to understand, critique, verify, and govern the intelligent systems that increasingly shape modern computing. As this transformation continues, the challenge for educators is clear: **preserve the rigor of traditional CS foundations while embracing the AI-native practices that will define the next generation of software development.**

6 Summary and Future Work

GenAI is changing software development from a code-centric activity to a supervision-centric workflow centered on specification, verification, and orchestration. Our deployments show that students experience genuine productivity gains, but also encounter predictable breakdowns when verification and architectural reasoning are not explicitly taught. We present a five-module course design that operationalizes these skills through required artifacts and staged interventions.

Future work will (i) collect stronger quantitative evidence using common rubrics for specification and verification quality, (ii) compare staged versus unrestricted GenAI policies, and (iii) evaluate whether specification-driven assessments improve long-term retention and transfer.

Acknowledgements

This work was supported by the Shepherd University Professional Development Grant and NSF S-STEM Grant (Award# 2130267). The author also wishes to thank all students in CIS 499A for their support and assistance in this project.

References

- [1] W. D. Liao. *Beyond Vibe Coding: Building Production-Grade Software with AI Agents and Specification-Driven Software Development*. ISBN: 979-8-271-00432-2. 2025.

- [2] Nidal Otoum and Nidal Elkhaili. “Methods and Techniques of Agentic Software Engineering: A Systematic Literature Review”. In: *IEEE Access* 14 (2026), pp. 7443–7465.

ReRead Text Pattern Editor for Open-Ended and Coding Questions*

Vladislav D. Veksler
Computer Science & Information Systems
Caldwell University
Caldwell, NJ 07006
vveksler@caldwell.edu

Abstract

Assessments featuring open-ended and coding questions are staples of the Computer Science curriculum, yet automatically validating these answers remains a significant challenge. While Regular Expressions (regex) are the industry standard for text pattern matching, their cryptic syntax makes them notoriously difficult to write, read, and maintain. Existing tools typically focus on testing regex or inserting common patterns, but fail to make the editing process intuitive or the results readable. This work introduces *ReRead*, a human-readable text pattern editor and JavaScript library that generates standard regular expressions compatible with major Learning Management Systems (LMS) and auto-graders. We demonstrate ReRead in two distinct scenarios: (1) creating flexible answer keys for open-ended questions in LMS exams, and (2) generating unit tests for student code within a custom Python auto-grading platform.

1 Introduction

In higher education, valid and reliable assessment is crucial for measuring student learning outcomes. Assessments often take the form of open-ended questions, with each question having not one single exact answer, but rather

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

a generic pattern of near-infinite correct answers. Validating answers to such questions requires flexible text pattern matching rather than strict string equality. The gold standard for defining such text patterns is the Regular Expression (regex). Regex use for scaling assessments is not just useful, but often necessary [1, 11].

Within Computer Science (CS) curricula in particular, open-ended assessments often request that students write code snippets. Such coding questions will have large numbers of ‘correct’ solutions, which are difficult to check at scale for human graders, and often require regex text pattern matching. Some testing platforms allow for coding questions to be validated via unit-tests, where student code is executed with specific inputs or parameters to ensure valid execution. While many student code solutions can be verified using simple logical assertions (e.g., checking if a function returns a specific value), a significant portion of introductory CS coursework involves generating console output or formatted strings, and validation of such outputs still requires regex pattern matching.

Another common form of open-ended questions in CS involves questions where students are provided code snippets, and must predict the expected output. Even in these cases, pattern matching is often preferred to exact string matching. A student’s answer might be logically correct but may vary slightly in spacing, capitalization, or the ordering of elements in unordered data structures (like Python sets). To accommodate these variations without unfairly penalizing students, grading scripts must employ robust pattern matching.

Regex sequences are extremely powerful, allowing for precise definition of complex string patterns, and they are allowed as answers for open-ended questions across all major testing platforms, including Canvas, Moodle, and Blackboard. However, regex patterns are notoriously difficult to read, fix, and edit [3]. The syntax is dense and non-intuitive; a pattern that makes perfect sense at the moment of writing often becomes indecipherable to the author just days later. This "write-only" nature of regex poses a significant hurdle in educational settings, where instructors must be able to comprehend and potentially change test answers.

Despite the abundance of regex tools available today, the majority suffer from a common limitation: they do not solve the fundamental readability problem. Most focus on syntax highlighting or testing, while generators that offer "human-readable" inputs often output the same cryptic, uneditable regex strings.

This work introduces *ReRead*, a human-readable text pattern editor designed to address these shortcomings. ReRead enables users to construct patterns using human-readable text, which the software then compiles into standard regular expressions. The software includes an online editor for regex

generation and a JavaScript library, allowing seamless integration into any web-based application or educational assessment software.

2 Related Work

The most common regex editing tools, such as RegViz [9], Regex101 [7], and RegExr [8], focus on syntax highlighting and testing patterns against input strings. They offer excellent micro-level breakdowns of what specific characters do (e.g., explaining that ‘\d’ matches a digit). However, for complex patterns, understanding the individual components does not necessarily help the user parse the overall logic of the text pattern.

Several generators attempt to simplify creation by providing buttons for inserting common regex patterns, such as "email" or "URL." However, such tools (e.g., RegExr [8]) merely insert the raw regex definition directly into the string. The human-readable abstraction is lost immediately upon generation. For instance, to match any string that might be an email address or a URL, you might click the "email" pattern and copy it, then click the "url" pattern and copy it, then paste them together with the "|" symbol in between, and you will end up with a string that looks like this:

```
(\\b[^\s@]+|(\\.[^\s@]+)*@([^\s@]+\.)+[\s@]{2,63}\\b|(https?:\\/\s|/|localhost|((25[0-5]|2[0-4]\\d|01?\\d?\\d)\\.){3}(25[0-5]|2[0-4]\\d|01?\\d?\\d)|([\\p{L}\\d]([-\\p{L}\\d]*[\\p{L}\\d])?\\.)+[\\p{L}\\d]{2,63})\\b)(:\\d+)?(\\/([\\p{L}\\d\\-\\_\\:\\/\\#\\[\\]\\@!$&'()*+;,=]|%[0-9A-Fa-f]{2}))*?)
```

Once generated, the user is left to maintain this complex string rather than simple human-readable text (whereas ReRead would show this email or url).

Recently, AI-driven tools have emerged that convert natural language prompts into regex (e.g., Rows [10]). While promising, these tools often produce incomplete or slightly inaccurate patterns. Using AI to generate regex bears the danger that any inaccuracies and imprecisions in the resulting pattern will go unnoticed. This is because despite the input prompt being written in natural language, the output is still the standard cryptic regex – difficult to read and difficult to fix.

Another category of regex editors attempt to map regex logic to box-and-arrow diagrams. Tools like Regex-Vis and Debuggex [6, 2] provide such visualizations, but these are still very difficult to interpret, and become very cluttered for complex patterns. Finally, there are also drag-and-drop interfaces that enable users to generate regex directly from box-and-arrow diagrams (e.g., NodeXR [4], VSCode RegEx Generator [5]). These come the closest to friendly and visually interpretable regex generation and editing, but often suffer from cumbersome drag-and-drop UIs and still remain difficult to interpret and edit for anything beyond the simplest text patterns.

3 Educational Integration

ReRead was integrated into an introductory Computer Science course to streamline the creation of assessment materials. The tool facilitated several key aspects of the testing process that were previously pain points for the instructional staff. Students took 8 coding tests, a midterm and a final exam. The unit tests for the coding tests and the open-ended questions on the midterm and final exams all allowed for variability in student answers. The ReRead editor enabled far easier test generation and answer management than would be possible via raw regex. Qualitative feedback from students indicated that they appreciated the immediate feedback provided by the ReRead-generated unit tests, specifically noting that it reduced uncertainty. From the instructor's point of view, the ability to easily generate and manage auto-grading via ReRead for coding and open-ended questions enabled far more testing than would otherwise be possible, aiding to achieve the sought learning outcomes.

3.1 Unit-Testing and Output Validation

The primary use case involved creating unit tests for Python coding questions. In introductory courses, students are frequently asked to write programs that produce specific console outputs. Validating this output is challenging due to the variability in student code.

The instructor used ReRead to generate flexible patterns that could account for common, non-fatal discrepancies. For example, when students printed the contents of a Python set (an unordered collection), the order of elements in the string representation would vary. ReRead allowed the instructor to easily construct patterns that matched the required elements regardless of order, accommodating minor spacing or spelling errors without impacting student grades.

3.2 Code Checking and Analysis

Beyond validating output, regular expressions were used to enforce static code requirements. Some exam questions required students to include specific structural components, such as comments or specific function headers. ReRead was used to generate the patterns necessary to scan student source code for these required elements, ensuring students met all stylistic and structural criteria.

3.3 Pedagogical Impact

The integration of ReRead directly addresses a friction point in pedagogy. While many testing platforms support regex, the difficulty of generating them

discourages instructors from asking complex open-ended questions, leading them to default to multiple-choice questions, limiting the scope of assessment.

For CS in particular, the ability to ask coding questions and other open-ended questions, like those asking for expected output of provided code, is absolutely critical. ReRead enables instructors to ask such questions, and to easily generate human-readable text patterns for correct answers and unit-test outputs to assess at scale.

By making the generation of regular expressions accessible and readable, ReRead empowers instructors to design more authentic assessments. It allows for the validation of complex student outputs without requiring the instructor to become an expert in regex syntax, ultimately leading to more robust and fair automated grading.

4 Demo Description

Software demo will comprise two scenarios designed to illustrate how instructors can rapidly deploy fair, flexible grading without regex expertise. Scenario 1 will demonstrate how open-ended questions with regex text pattern answers may be added in a sample popular Learning Management System (LMS) often used across many universities – Blackboard. In this scenario the ReRead Editor will be used to create and copy correct answer text patterns, and then these answers will be pasted into Blackboard questions. Scenario 2 will demonstrate how coding question answers and unit-test outputs may be checked against text patterns. In this scenario the ReRead JavaScript Library will be employed as a component of code testing software to aid in generation of answer patterns and unit tests.

4.1 Scenario 1: Adding an open-ended question to a quiz in LMS

The scenario will involve the creation of two “Fill in the Blank” questions on Blackboard. The first question will be: “Who is credited with coining the term ‘Artificial Intelligence’ at the Dartmouth Conference in 1956?” Presenter will ask for audience volunteers to write the correct regex. Presenter will then navigate to the ReRead online editor, and create the correct regular expression to match all possible answers (John McCarthy, j. mccarthy, McCarthy, etc.), as shown in Figure 1 (top). Presenter will highlight the ease of ReRead generation and interpretability, as compared to regex. Presenter will then copy the generated regex into the Blackboard question’s answer field.

The second question will be: “Write a SQL query to retrieve all columns from the Customers table where the ID is 10.” Presenter will ask for audience volunteers to write the correct regex. Presenter will then navigate to the ReRead online editor, and create the correct regular expression to match all

possible answers, as shown in Figure 1 (center). Presenter will highlight the ease of ReRead generation and interpretability, as compared to regex. Presenter will then copy the generated regex into the Blackboard question’s answer field.

The scenario will conclude by showing how student answers matching these patterns are automatically graded as correct. Presenter will navigate to the student view, take the test, fill in answers, and submit the test. The final scenario screen will display Blackboard auto-grading for the submission, as shown in Figure 1 (bottom).

4.2 Scenario 2: Coding question with unit-tests

This scenario will demonstrate how the ReRead JavaScript library (reread.js) can be effectively integrated into a testing platform. The demonstration will feature a custom web-based auto-grading platform designed for introductory computer science courses, capable of executing Python code and verifying its output. In this scenario reread.js is used to streamline the creation of robust unit tests for coding questions.

Presenter will first navigate to the instructor interface and create two Python coding questions. The first will be a simple “Ask user for their name. Output, "hello " followed by their name.”. The second question will be : “Write a loop that continues to ask user to enter text, then echoes the text back, until user enters no input.” Presenter will open the unit test editing screen for each question, create unit tests and validate them.

Highlighted within this interface will be the ReRead JavaScript library components, seamlessly embedded into the test creation form (see sample partial screenshot in Figure 2). Instead of writing a complex and error-prone regular expression to capture every possible permutation of the output, presenter will use the ReRead visual editor to construct a pattern. Presenter will demonstrate how easily they can define a pattern that matches varying output with flexible whitespace using ReRead’s human-readable UI, emphasizing that instructors do not require prior knowledge of regex syntax.

The scenario will conclude by switching to the student view. Presenter will input a correct solution that produces output in a different order than a standard hard-coded string might expect. Upon submission, the system will correctly identify the output as valid and award full points, demonstrating the tool’s immediate practical value in ensuring fair and accurate automated grading.



Figure 2: Unit-test partial screenshot.

5 Significance and Conclusions

The primary significance of ReRead lies in its ability to democratize the power of regular expressions for educational assessment. By abstracting the cryptic syntax of standard regex into a human-readable and editable format, ReRead lowers the technical barrier for instructors. This is particularly impactful in large-scale Computer Science courses, where the ability to grade open-ended coding questions and variable outputs is essential for authentic assessment and achieving sought learning outcomes. Furthermore, ReRead directly addresses the “write-only” nature of legacy regex tools. By ensuring that answer patterns remain intelligible over time, the tool ensures the sustainability of assessment materials. Instructors can confidently modify existing questions or debug grading logic without the need to decipher dense strings of special characters. This fosters a more fair grading environment, where students are not penalized for minor, non-semantic variations in their answers simply because the grading script was too rigid or too difficult to adjust.

In conclusion, ReRead offers scalability for open-ended assessment questions, enabling pattern-matching flexibility while obscuring the technical complexity of standard regex. Whether integrated directly into an LMS or embedded within custom auto-grading software, ReRead reduces the gap between the ease of writing open-ended and multiple choice questions, empowering educators to implement assessments based on their rigor and value.

References

- [1] Enrique Sánchez Acosta and Juan José Escribano Otero. “Automated assessment of free text questions for MOOC using regular expressions”. In: *Information Resources Management Journal (IRMJ)* 27.2 (2014), pp. 1–13.
- [2] *Debuggerx*. <https://debuggex.com/>. Accessed: 2025.
- [3] Louis G. Michael et al. “Regexes are hard: decision-making, difficulties, and risks in programming regular expressions”. In: *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE ’19)*. IEEE Press, 2019, pp. 415–426. DOI: 10.1109/ASE.2019.00047.
- [4] *NodeXR*. <https://www.nodexr.net/>. Accessed: 2025.
- [5] *Regex Generator*. Visual Studio Marketplace. VSCode Extension.
- [6] *Regex-Vis*. <https://regex-vis.com/>. Accessed: 2025.
- [7] *regex101*. <https://regex101.com/>. Accessed: 2025.
- [8] *Regexr*. <https://regexr.com/>. Accessed: 2025.
- [9] *RegViz*. <https://regviz.org>. Accessed: 2025.
- [10] Rows. *AI Regex Generator*. <https://rows.com/tools/regex-generator>. Accessed: 2025.
- [11] Oleg Sychev. “Open-answer question with regular expression templates and string completion hinting”. In: *Software Impacts* 17 (2023), p. 100539.

Teaching AI the Learner-Centered Way: Designing a Meaningful AI Literacy Lesson Model for Middle School*

Yewon Lee¹ and Hia Ghosh²

¹Computer Science Department
Mount Holyoke College
South Hadley, MA 01075

lee65y@mholyoke.edu

²Computer Science Department
Smith College
Northampton, MA 01063

hghosh@smith.edu

Abstract

As artificial intelligence(AI) becomes increasingly visible in everyday life, many students are experiencing growing anxiety about what AI is and how it affects them. This signals the urgent need for meaningful AI literacy education that helps learners understand how AI systems work and empowers them to use these tools confidently and effectively. Recognizing the importance of ensuring that all learners can develop this understanding, this paper introduces a learner-centered, research-informed lesson model for AI literacy designed for students in grades 6–8. The lesson envisions AI literacy through a learner-centered lens, prioritizing accessibility, agency, and inclusivity. Grounded in AI4K12’s “Big Five Ideas in AI” and systematically integrating learning science frameworks such as Universal Design for Learning (UDL), Funds of Knowledge, and Depth of Knowledge, the proposed lesson model demonstrates how theoretical principles can be operationalized within AI literacy lesson design.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

AI is increasingly embedded across various sectors of society, including education [19]. As AI technologies grow more visible and powerful, students and the broader public are experiencing rising concerns, often rooted in uncertainty about AI's potential to outperform human capabilities [11]. The research shows anxieties and misconceptions underscore the importance of AI literacy, which can help demystify AI systems and empower learners to understand, evaluate, and engage with them responsibly.

This paper presents a learner-centered AI literacy lesson model aimed at supporting students in grades 6–8 in developing a grounded and confident understanding of AI. The lesson is anchored in the AI4K12 initiative's Big Five Ideas in AI, a well-established framework for K-12 AI education. To ensure the lesson is meaningful and accessible for all learners, three learning science frameworks: Universal Design for Learning (UDL), Funds of Knowledge, and Depth of Knowledge are integrated throughout. These frameworks guide the design toward inclusivity, relevance, and engagement, helping sustain students' investment in learning about AI and ultimately fostering robust AI literacy.

The lesson model articulates a set of core design commitments that integrate AI4K12 content with established learning science frameworks. This offers a structured and transferable approach to sequencing, scaffolding, and assessing AI literacy concepts in developmentally appropriate ways. The research questions guiding this work are:

1. How can AI literacy be introduced in accessible ways for 6-8th grade learners?
2. How can learning science frameworks enhance the inclusivity of AI education?
3. What does the lesson model look like when AI4K12 and learning science frameworks are systematically integrated?

2 Conceptual Framework: AI Literacy in Education

AI literacy has been defined in various ways across educational technology literature, encompassing a range of competencies related to using, interpreting, and understanding AI. This paper focuses on the definition of AI literacy that explains an understanding of how AI systems function and how they can be effectively applied in everyday contexts. This includes the processes underlying AI systems, awareness of their capabilities and limitations, and the ability to make informed decisions about when to use or not use AI [3, 12, 17].

Prior research has shown that individuals who understand how AI operates are better equipped to interpret AI-generated outputs, recognize AI's limita-

tions and biases, and avoid over-reliance on AI [15, 10]. Developing AI literacy, therefore, plays a critical role in reducing fear and misconceptions towards AI [11] and supports more empowered, critical engagement with AI. Rather than perceiving AI as a threat or replacement for human creativity and agency, learners can see AI as a tool that enhances learning and complements human expertise [25]. This orientation prepares them with the skills necessary to navigate future learning environments where AI will be pervasive [7].

To structure the lesson design, the study draws on AI4K12’s Big Five Ideas in AI [22], a curriculum framework that articulates foundational concepts across five domains:

1. Perception: how AI perceives the world using sensory inputs;
2. Representation and Reasoning: how AI organizes information and draws inferences;
3. Learning: how AI learns by identifying patterns and improving performance;
4. Natural Interaction: how AI engages in human-like communication;
5. Societal Impacts: how AI influences individuals and communities in diverse ways.

The framework’s Progression Chart provides a clearer picture of the AI curriculum by outlining age-appropriate subsidiary concepts for each idea, spanning grades K-2 through 9-12. For every concept, it identifies both specific learning objectives and “enduring understandings” that students should develop in order to engage meaningfully with AI concepts [1]. This structured, developmentally sequenced set of guidelines offers a strong foundation for designing inclusive, learner-centered AI literacy lessons that are both pedagogically sound and adaptable to diverse learning needs.

3 Lesson Design Approach

The lesson model presented in this paper serves as a theory-informed instructional framework that teachers may adapt to diverse classroom contexts. It is structured as a two-part sequence: the first lesson includes a warm-up and instruction and introduces a take-home application task, and the second lesson centers on student presentations and a whole-class discussion.

Prior scholarship highlights several challenges in the development of AI literacy for K–12 learners. Systematic reviews identify a gap in public understanding of AI that is reflected in societal fears and misconceptions [3]. Exploratory reviews also note variation in how AI literacy is defined across studies, suggesting that the field continues to evolve conceptually [17]. These

findings point to the need for instructional approaches that promote clearer conceptual grounding in how AI systems function.

Issues of trust and fear further complicate AI learning. Research indicates that providing explanations for AI decisions can reduce overtrust, particularly when AI outputs contradict users' expectations or reasoning processes [15]. Studies also identify widespread concerns about privacy, bias, job displacement, and transparency, while noting that although AI literacy is often proposed as a solution, how it helps users manage these risks remains underexplored [11]. Prior work on AI explanations emphasizes alignment with users' mental models to support meaningful human–AI interaction [10]. Together, this research underscores the importance of strengthening learners' conceptual understanding and critical reasoning about AI systems.

Equity considerations further highlight the need for inclusive AI literacy design. Research on the application of Funds of Knowledge in STEM education identifies gaps in understanding how students respond to such interventions, particularly regarding learning outcomes and identity development [6]. Limited assessments of belonging and self-efficacy, alongside the continued marginalization of groups and diversity challenges in STEM [6], reinforce the importance of equity-focused instructional strategies. Scholars have also noted that although Universal Design for Learning has shown promise in STEM contexts, it has not yet been fully leveraged for inclusive AI learning design [21].

To address these conceptual, trust-related, and equity-oriented challenges, the lesson model introduced in this paper integrates three learning science frameworks: Universal Design for Learning, Funds of Knowledge, and Depth of Knowledge.

3.1 Universal Design of Learning (UDL)

Universal Design for Learning (UDL) offers a framework for designing accessible instruction that anticipates learner variability and reduces potential barriers to learning [20]. UDL emphasizes providing multiple pathways for students to perceive information, stay motivated, and demonstrate understanding. These principles [4] are operationalized through:

- Multiple means of representation: offering multimodal ways of presenting concepts
- Multiple means of action and expression: allowing students flexible options to communicate what they know
- Multiple means of engagement: providing choices and supports that promote purposeful learning.

UDL helps create learning experiences that feel enjoyable, relevant, and cognitively manageable for all learners [2]. By ensuring flexible entry points

and support, UDL enhances students' readiness to engage with complex ideas and prepares them for a future in which AI is increasingly pervasive [4].

3.2 Funds of Knowledge (FoK)

The Funds of Knowledge(FoK) framework emphasizes the rich cultural, social, and experiential knowledge that learners bring from their homes and communities [16]. Integrating FoK into lesson design positions students' lived experiences as valuable resources for understanding concepts, enabling them to connect disciplinary content with familiar concepts [9]. In STEM fields, FoK-based instruction can broaden participation by making learning feel relevant and personally meaningful [6]. By bridging students' everyday knowledge with formal AI concepts, FoK supports more inclusive learning environments and helps reduce barriers that may deter learners from historically underrepresented groups.

3.3 Depth of Knowledge (DoK)

The Depth of Knowledge(DoK) framework outlines four levels of cognitive complexity, ranging from simple recall to extended strategic reasoning:

- Level 1: Recall and reproduction
- Level 2: Basic application of skills and concepts
- Level 3: Strategic thinking and reasoning
- Level 4: Extended, sustained inquiry

Designing lessons across DoK levels enables learners to gradually progress from foundational understanding to more sophisticated application, analysis, and transfer of knowledge [8]. Leveraging DoK supports students in moving beyond surface-level understanding toward active, critical engagement with AI concepts. This progression promotes deeper ownership of knowledge and enhances students' ability to apply AI literacy skills across contexts [14].

Together, these frameworks inform a coherent design logic within the lesson model: (1) establish an intuitive cognitive entry point grounded in students' lived experiences; (2) scaffold abstraction from human sensemaking to computational perception; (3) distribute structured autonomy through multimodal choice; and (4) consolidate understanding through evaluative comparison and reflective discussion. This progression ensures that inclusivity, cognitive rigor, and conceptual clarity are structurally embedded rather than appended to instruction.

4 Lesson Model Design

This paper presents one instantiation of the lesson model, focusing on the AI4K12 concept of “Perception.” Drawing on the learning principles outlined above, the lesson is structured to activate students’ prior knowledge, introduce core ideas of multimodal perception of AI, and support deeper inquiry into how humans and AI interpret information from multiple sources.

4.1 Lesson at a Glance

- **Target grade level:** Grades 6–8
- **Format and pacing:** Two class sessions plus a take-home task.
 - **Lesson 1 (45–55 minutes):** Warm-up, direct instruction on modality and multimodal AI, introduction of the choice board task, and planning time.
 - **Between lessons (20–40 minutes, outside of class):** Students complete the selected choice board deliverable (zine, slides, or podcast) individually or in small groups.
 - **Lesson 2 (45–55 minutes):** Student presentations, whole-class comparative analysis of human vs. AI perception, and reflection.
- **Expected prior knowledge:** Students should be able to describe observations from images or sounds, participate in discussion, and distinguish between basic data types (for example, text vs. image). Teachers may optionally pre-teach vocabulary, such as data, model, and prediction. No coding required.
- **Materials:** projected or printed warm-up clues, optional device access for research, and a template or checklist for the take-home deliverable.

4.2 Lesson 1: Warm-up, Instruction, and Task Launch

4.2.1 Warm-up Activity: ‘What is Happening?’

To spark curiosity and establish a shared entry point, students engage in a warm-up activity called ‘What is happening?’. This activity draws on students’ everyday experiences and thus aligns with the Funds of Knowledge framework. Students are shown a sequence of progressively revealed clues, with each step presenting a new piece of information in a different modality (e.g., visual or audio). After each reveal, students are asked to update their prediction about what is happening in the full scenario, refining their thinking as more multimodal evidence becomes available. For example:

Scenario 1: Storm or Construction?

- Step 1: Dark sky, clouds rolling in

- Step 2: A loud boom
- Step 3: Construction workers using a jackhammer under gray clouds

Scenario 2: Cafe Setup or Closing?

- Step 1: A cafe, chairs on tables, lights half-dimmed
- Step 2: Doorbell rings
- Step 3: A person carrying a bag of supplies
- Step 4: The clock shows 5:50 am

After all clues are revealed, they collaboratively discuss the following questions: “How did your understanding change as more clues were added?” “How would your interpretation differ if only one clue were available?” This activity intentionally incorporates familiar, culturally adaptable contexts, allowing teachers to tailor scenarios that resonate with students’ lived experiences. By grounding early sensemaking in accessible examples, the lesson establishes a flexible entry point consistent with the FoK principle and prepares students to connect human perception with AI perception.

4.2.2 Introducing Multimodal Perception in AI

Building on the warm-up activity, the lesson shifts from human sensemaking to artificial perception by introducing how AI systems interpret the world through multiple types of data. This transition reinforces the connection between how students combined clues in the earlier activity and how AI integrates different inputs to form understanding.

The section begins with a guided whole-class discussion using the prompt, “What technologies do we know that can see, hear, or understand language?” Students brainstorm familiar examples such as voice assistants, self-driving cars, chatbots, and facial recognition systems. This discussion situates multimodal AI within everyday technology and helps students recognize that many common systems rely on more than one type of data.

Students are then introduced to the concept of modality. A modality is defined as a type of data, such as text, images, audio, or video; multimodal AI is explained as an AI system that can process and integrate multiple modalities at the same time [5]. The instruction then outlines how multimodal AI works in three stages [23]. First, AI processes different data types using specialized models such as computer vision for images, speech recognition for audio, and language models for text. Second, the system combines this information through information fusion to identify patterns and relationships across modalities, such as linking the visual features of a dog to the text label “golden retriever.” Third, the AI uses this integrated understanding to produce outputs such as answering questions, describing images, or interpreting spoken commands.

Throughout this section, students are encouraged to compare human perception with AI perception. Just as they relied on multiple clues to refine their interpretations in the warm-up, multimodal AI systems use multiple data inputs to improve accuracy and understanding. This connection prepares students for the application activity that follows, where they explore real-world examples of multimodal AI.

4.3 Take-Home Activity: ‘Deeper Dive into Multimodal AI in Our Lives’

To support the core tenets of UDL, choice and autonomy, the lesson incorporates a multimodal choice board, which is inspired by the digital choice board method developed by the University of Massachusetts [13]. The board organizes examples of multimodal AI by modality (visual, sound, language) and societal domain (healthcare, education, transportation, content creation). Students select one modality-domain intersection to investigate multimodal AI in a real-world context. The choice board is shown below. This structure provides:

- Multiple means of engagement: autonomy in selecting topics relevant to students’ interests or lived experiences.
- Collaborative knowledge-building: students explore different technologies and prepare to contribute diverse examples to the class discussion.

Table 1: Choice board of AI Applications Using Visual Data

Domain	Examples
Healthcare	AI scans X-rays or MRIs to spot illnesses like cancer. Smart cameras track patients’ movements to prevent falls in hospitals.
Education	AI reads handwritten math problems or scans worksheets to check answers. Virtual labs show chemical reactions through simulations.
Transportation	Self-driving cars use cameras to detect traffic lights, pedestrians, and road signs. Drones inspect roads or bridges for safety.
Content Creation	AI chatbots use facial recognition to detect customer emotions during video calls. Smart stores use cameras to track shelf stock.

Table 2: Choice board of AI Applications Using Sound Data

Domain	Examples
Healthcare	Voice assistants help doctors take notes by listening to patient conversations. AI listens to coughing sounds to help diagnose respiratory illnesses.
Education	AI language tutors listen to pronunciation and give feedback for better speaking practice. Classroom devices use sound to measure student participation.
Transportation	AI listens for sirens or engine sounds to alert drivers or fix car issues early. Smart buses announce stops using speech recognition.
Content Creation	Music AIs compose songs or beats based on a theme. Voice generators create character voices for podcasts or videos.

Table 3: Choice board of AI Applications Using Language Data

Domain	Examples
Healthcare	AI summarizes medical research for doctors. Chatbots help patients understand their prescriptions.
Education	AI tutors explain lessons or answer student questions in natural language. Essay checkers give writing feedback.
Transportation	Navigation systems explain directions clearly using spoken or written language. AI systems read traffic reports and plan routes.
Content Creation	AI tools write stories, scripts, or captions. Chatbots help brainstorm video ideas or social media posts.

Students' research and final deliverables are designed to reach level 4 of Depth of Knowledge, requiring synthesis, transfer, and evaluation across contexts. In their work, students explain:

- How the selected multimodal AI sees, listens, or understands;
- How these capabilities support AI's real-world applications;
- What challenges or risks arise, including cultural bias, privacy concerns, and over-reliance on multimodal AI.

Students begin selecting a modality-domain topic and planning their approach in class, then complete the final deliverable as a take-home task before the second session.

4.4 Lesson 2: Presentations and Comparative Discussion

4.4.1 Student Presentations and Share-Out

During the second lesson, students take turns sharing the work they completed as part of the take-home activity. Each student presents their findings in the format selected from the multimodal choice board, such as a zine, slide presentation, or podcast. This structure reflects the principle of multiple means of action and expression within Universal Design for Learning while maintaining consistency in the analytical expectations across formats.

Peers are encouraged to ask clarifying and critical questions, fostering collaborative knowledge construction and strategic reasoning. By comparing examples across domains and modalities, students deepen their conceptual understanding of AI perception and strengthen transfer and synthesis aligned with higher levels of Depth of Knowledge.

4.4.2 Comparative Analysis: Human vs. AI Perception

After sharing their work, students engage in a whole-class discussion to analyze differences between human and AI perception by addressing prompts such as:

- How does AI's understanding differ from ours?
- What additional information would an AI need to fully understand a scene?
- How might cultural bias influence AI perception?

These discussions target level 3 of Depth of Knowledge by supporting student evaluation, comparison, and reasoning. Teachers may use culturally specific examples, such as an AI misinterpreting a local dialect or a traditional sport, to help students recognize how cultural diversity shapes interpretation and

how bias can manifest in AI. This again leverages students' Funds of Knowledge, framing cultural experience as an asset for critically understanding AI perception and its societal implications.

5 Teacher Evaluation Questions and Measures

Recent work on AI literacy assessment proposes that learning outcomes should be examined across affective, behavioural, cognitive, and ethical dimensions [18]. Scholarship on AI literacy assessment emphasizes grounding evaluation in students' projects and artifacts rather than relying solely on decontextualized testing [24]. Drawing on these approaches, the following classroom-aligned measures are proposed.

5.1 Formative conceptual checks (activity-based assessment).

Teachers may pose short-answer or discussion questions such as:

- What is a modality? Provide two examples.
- Compare human and AI perception. What does AI need that humans often infer automatically from context?
- Name one risk of multimodal AI (bias, privacy, or overreliance). What is one action you can take to use it more carefully?

These questions assess students' grasp of foundational AI concepts and common misconceptions.

5.2 Performance-based product rubric (project-based concept inventory).

Students' take-home artifacts and presentations may be evaluated using the following criteria:

- Accurate explanation of how the system uses modalities to produce output;
- Identification of a limitation or failure case tied to a real-world risk;
- Justification of when to rely on the system and when not to (calibrated trust versus overtrust);
- Clear communication in a chosen medium, reflecting multiple means of expression.

Teachers may further extend evaluation through concept discussions or brief artifact interviews, inviting students to reflect on how their understanding

evolved and to clarify reasoning during presentation. Collectively, these measures reflect research-based principles of AI literacy assessment while remaining developmentally appropriate for middle school learners.

6 Contribution

This work contributes to the growing field of K–12 AI education by offering a concrete, theory-informed model for learner-centered AI literacy instruction. While prior research has emphasized the importance of AI literacy and concerns related to fear, trust, and conceptual ambiguity, fewer studies have articulated how learning science frameworks can be systematically integrated into AI lesson design for middle school students.

This paper advances the field in three ways. First, it operationalizes AI4K12’s “Big Five Ideas in AI” within a structured lesson model on multimodal perception, translating abstract AI concepts into cognitively accessible classroom experiences. Second, it demonstrates how Universal Design for Learning, Funds of Knowledge, and Depth of Knowledge function as complementary design lenses guiding engagement, representation, and cognitive rigor. Third, it articulates a replicable yet adaptable lesson model that educators and curriculum designers can extend to additional AI concepts while maintaining conceptual depth and cultural responsiveness.

By bridging AI education frameworks with inclusive pedagogical theory, this work offers a practical model for equitable AI literacy instruction.

7 Limitation and Future Work

As a theory-informed instructional design model, this work has not yet undergone classroom-based validation. Future implementation studies will examine student learning outcomes, engagement patterns, and equity-related impacts.

Implementation will enable empirical examination of how students engage with multimodal AI concepts and how effectively the integrated frameworks support inclusive participation. Observing how students from diverse cultural and educational contexts respond to Funds of Knowledge-based activities will further clarify how culturally responsive AI instruction influences engagement and identity development in STEM.

8 Conclusion

As artificial intelligence becomes increasingly embedded in everyday life, preparing young learners with foundational AI literacy is both urgent and essential. This paper presents a learner-centered, research-informed lesson model

grounded in AI4K12’s “Big Five Ideas in AI” and informed by Universal Design for Learning, Funds of Knowledge, and Depth of Knowledge, providing a cohesive foundation for accessible, culturally responsive, and intellectually rigorous instruction.

The lesson on multimodal perception demonstrates how middle school students can move from intuitive sensemaking to a structured understanding of how AI systems interpret and integrate data. Through multimodal exploration, deliberate cognitive progression, and opportunities for student choice and reflection, the design supports deeper engagement while addressing concerns related to fear, bias, and over-reliance on AI.

Rather than positioning AI as a distant or threatening technology, this approach invites students to see AI as understandable, critiqueable, and usable, offering a promising pathway toward equitable and empowering AI literacy education.

References

- [1] AI4K12. *Big Idea 1 – Perception*. URL: <https://ai4k12.org/big-idea-1-overview/>.
- [2] Susan M. Barteaux. “Universal Design for Learning”. In: *BU Journal of Graduate Studies in Education* 6.2 (2014), pp. 50–54. URL: <https://files.eric.ed.gov/fulltext/EJ1230738.pdf>.
- [3] Lorena Casal-Otero et al. “AI literacy in K-12: a systematic literature review”. In: *International Journal of STEM Education* 10 (2023), pp. 1–17. DOI: <https://doi.org/10.1186/s40594-023-00418-7>.
- [4] Susan Joan Courey et al. “Improved Lesson Planning With Universal Design for Learning (UDL)”. In: *Teacher Education and Special Education: The Journal of the Teacher Education Division of the Council for Exceptional Children* 36.1 (2013), pp. 7–27. DOI: <https://doi.org/10.1177/0888406412446178>.
- [5] Abby Curtis and Chrissy Kidd. *What Is Multimodal AI? A Complete Introduction / Splunk* — [splunk.com](https://www.splunk.com/en_us/blog/learn/multimodal-ai.html). [Accessed 08-12-2025]. 2024. URL: https://www.splunk.com/en_us/blog/learn/multimodal-ai.html.
- [6] Maya Denton and Maura Borrego. “Funds of Knowledge in STEM Education: A Scoping Review”. In: *Studies in Engineering Education* 1.2 (2020), pp. 71–92. URL: <https://seejournal.org/articles/10.21061/see.19>.

- [7] Fredrik Heintz. “Three Interviews About K-12 AI Education in America, Europe, and Singapore”. In: *KI - Künstliche Intelligenz* 35 (2021), pp. 233–237. DOI: <https://doi.org/10.1007/s13218-021-00730-w>.
- [8] Karin Hess et al. “Cognitive Rigor: Blending the Strengths of Bloom’s Taxonomy and Webb’s Depth of Knowledge to Enhance Classroom-level Processes”. In: (2009). URL: <https://files.eric.ed.gov/fulltext/ED517804.pdf>.
- [9] Linda Hogg. “Funds of Knowledge: An investigation of coherence within the literature”. In: *Teaching and Teacher Education* 27.3 (2011), pp. 666–677. DOI: <https://doi.org/10.1016/j.tate.2010.11.005>.
- [10] Yu Liang et al. “Benefits of Machine Learning Explanations: Improved Learning in an AI-assisted Sequence Prediction Task”. In: Association for Computing Machinery, 2025, pp. 231–246. ISBN: 9798400713064. DOI: <https://doi.org/10.1145/3708359.3712155>.
- [11] Jiahui Liu. “When Usefulness Fuels Fear: The Paradox of Generative AI Dependence and the Mitigating Role of AI Literacy”. In: *International Journal of Human-Computer Interaction* (2025), pp. 1–22. DOI: <https://doi.org/10.1080/10447318.2025.2544006>.
- [12] Duri Long and Brian Magerko. “What is AI Literacy? Competencies and Design Considerations”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–16. ISBN: 9781450367080. DOI: <https://doi.org/10.1145/3313831.3376727>.
- [13] Robert W. Maloy et al. *Digital Choice Boards and Interactive Learning Materials for Teachers and Students*. URL: <https://scholarworks.umass.edu/collections/ed96c4c4-b005-4396-8d93-5f67ae7d1415>.
- [14] Feruza Masharipova. “From Bloom’s Taxonomy Into Webb’s Depth of Knowledge: Enhancing Lesson Planning Strategies”. In: *Emergent: Journal of Educational Discoveries and Lifelong Learning (EJEDL)* 6.2 (2024), pp. 1–10. DOI: <https://doi.org/10.47134/emergent.v3i4.50>.
- [15] Sina Mohseni et al. “Machine Learning Explanations to Prevent Overtrust in Fake News Detection”. In: *Proceedings of the International AAAI Conference on Web and Social Media* 15.1 (May 2021), pp. 421–431. DOI: <https://doi.org/10.1609/icwsm.v15i1.18072>.
- [16] Luis C. Moll et al. “Funds of Knowledge for Teaching: Using a Qualitative Approach to Connect Homes and Classrooms”. In: *Theory Into Practice* 31.2 (1992), pp. 132–141. DOI: <https://doi.org/10.3390/languages6020102>.

- [17] Davy Tsz Kit Ng et al. “Conceptualizing AI literacy: An exploratory review”. In: *Computers and Education: Artificial Intelligence* 2 (2021), p. 100041. DOI: <https://doi.org/10.1016/j.caeai.2021.100041>.
- [18] Davy Tsz Kit Ng et al. “Design and validation of the AI literacy questionnaire: The affective, behavioural, cognitive and ethical approach”. In: *British Journal of Educational Technology* 55.3 (2024), pp. 1082–1104. DOI: 10.1111/bjet.13411.
- [19] Adib Bin Rashid and MD Ashfakul Karim Kausik. “AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications”. In: *Hybrid Advances* 7 (2024), p. 100277. DOI: <https://doi.org/10.1016/j.hybadv.2024.100277>.
- [20] David Rose. “Universal Design for Learning”. In: *Journal of Special Education Technology* 15.4 (2000), pp. 47–51. DOI: <https://doi.org/10.1177/016264340001500407>.
- [21] Yukyeong Song et al. “A framework for inclusive AI learning design for diverse learners”. In: *Computers and Education: Artificial Intelligence* 6 (2024), p. 100212. DOI: <https://doi.org/10.1016/j.caeai.2024.100212>.
- [22] David Touretzky et al. “Envisioning AI for K-12: What Should Every Child Know about AI?” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 9795–9799. DOI: <https://doi.org/10.1609/aaai.v33i01.33019795>.
- [23] *What is Multimodal AI? — salesforce.com*. [Accessed 08-12-2025]. URL: <https://www.salesforce.com/artificial-intelligence/multimodal-ai/>.
- [24] Randi C. Williams. *Constructionist AI Assessments for K-12 Education*. Accessed 2025-03-01. 2023. URL: https://randi-c-dubs.github.io/K12-AI-ed/Constructionist_AI_Assessments.pdf.
- [25] Jun Zhao. “The role of learners’ AI literacy and resilience in boosting their engagement and motivation in AI-based settings: From an achievement goal theory perspective”. In: *Learning and Motivation* 91 (2025), p. 102152. ISSN: 0023-9690. DOI: <https://doi.org/10.1016/j.lmot.2025.102152>.

Reviewers — 2026 CCSCNE Conference

Ali Al-Faris	<i>Worcester State University</i>
Mohit Chandarana	<i>Codio</i>
R. Jordan Crouser	<i>Smith College</i>
Lawrence D’Antonio	<i>Ramapo College</i>
Peter Drexel	<i>Plymouth State University</i>
Razieh Fathi	<i>UMass Dartmouth</i>
Michael Gousie	<i>Wheaton College (MA)</i>
William Harrison	<i>St. John Fisher University</i>
Delbert Hart	<i>SUNY Plattsburgh</i>
E. K. Iskrenova-Ekiert	<i>SUNY Brockport</i>
Michael Jonas	<i>University of New Hampshire</i>
David Levine	<i>University of Southern Maine</i>
Sriharsha Mallapuram	<i>Plymouth State University</i>
Kevin McCullen	<i>SUNY Plattsburgh</i>
Sandeep Mitra	<i>Rochester Institute of Technology</i>
Sofya Poger	<i>Felicia University</i>
Rajesh Prasad	<i>Saint Anselm College</i>
Angel Rivera	<i>Utica University</i>
Elijah Rivera	<i>Brandeis University</i>
Ingrid Russell	<i>University of Hartford</i>
Unnati Shah	<i>Utica University</i>
Babafemi Sorinolu	<i>Houghton University</i>
Vikas Thammanna Gowda	<i>Champlain College</i>
Scott Valcourt	<i>Northeastern University</i>
Drew van der Poel	<i>Northeastern University</i>
Yang Wang	<i>La Salle University</i>