The Journal of Computing Sciences in Colleges

Papers of the 31st Annual CCSC Central Plains Conference

April 4th-5th, 2025
Drake University
Des Moines, IA

Bin Peng, Associate Editor
Park University

Joan Gladbach, Regional Editor
University of Missouri Kansas City

Volume 40, Number 6 **April 2025**

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.

Table of Contents

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.

Cathy Bareiss, Membership Secretary (2025), cathy.bareiss@betheluniversity.edu, Department of Mathematical & Engineering Sciences, Bethel University, Mishawaka, IN 46545.

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.

David Naugler, Midsouth Representative (2025), dnaugler@semo.edu, Brownsburg, IN 46112.

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

Mark Bailey, Northeastern Representative (2025), mbailey@hamilton.edu, Computer Science Department, Hamilton College, Clinton, NY 13323.

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

Mohamed Lotfy, Rocky Mountain Representative (2025), mohamedl@uvu.edu, Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Mika Morgan, South Central Representative (2027), mikamorgan@wsu.edu, Department of Computer Science, Washington State University, Pullman, WA 99163.

Karen Works, Southeastern Representative (2027), keworks@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.
Brian Hare, Associate Treasurer & UPE Liaison, hareb@umkc.edu, School of Computing & Engineering, University of Missouri-Kansas City, Kansas City, MO 64110.
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.

Gold Level Partner
Rephactor
ACM2Y
CodeGrade
GitHub

Silver Level Partner
CodeZinger

Welcome to the 2025 CCSC Central Plains Conference

Welcome to the 31st annual Consortium for Computing Sciences in Colleges Central Plains Region Conference, which includes opportunities to see great research papers on a variety of topics in CS education; workshops on teaching with generative AI, collaborative projects, Codespaces, and containers; nifty assignments, and a panel on developing academic programs focusing on AI. We are also looking forward to showcasing students with a research poster competition, a hackathon, and a programming competition.

We are especially excited about our keynote speakers. Meredith Moore will share her expertise on accessible computing and what every CS educator should know. John Emmons will share on experiences in leading AI teams.

This conference would not be possible without the dedication of many contributors. We extend our gratitude to the authors who submitted their work, the reviewers who ensured a high-quality program with a 53% paper acceptance rate, the committee members who planned and organized the event, the session moderators, and all the on-site volunteers working to create a pleasant experience for everyone. And, thank you to our national partners for their very generous support!

We hope you have an enjoyable and enlightening conference!

Eric Manley
Drake University
CCSC-2025 Central Plains Conference Chair and Host

2025 CCSC Central Plains Conference Steering Committee

Conference Chair
Eric Manley ..... Drake University
Conference Publicity
Bill Siever ..... Washington University in St. Louis
Joan Gladbach ..... University of Missouri Kansas City
Shane Adams ..... Graceland University
Keynote Speaker
Eric Manley ..... Drake University
Pre-Conference Workshop
Wen-Jung Hsin ..... Park University
Judy Mullins ..... Retired
Joan Gladbach ..... University of Missouri Kansas City
Papers
Charles Riedesel ..... University of Nebraska-Lincoln
Judy Mullins ..... Retired
Ron McCleary ..... Retired
Panels, Tutorials, Workshops
Ron McCleary ..... Retired
Judy Mullins ..... Retired
Mohammad Rawashdeh ..... University of Central Missouri
Mahmoud Yousef ..... University of Central Missouri
Nifty Assignments
Mohammad Rawashdeh ..... University of Central Missouri
Brian Hare ..... University of Missouri Kansas City
Bill Siever ..... Washington University in St. Louis
Ron McCleary ..... Retired
Judy Mullins ..... Retired
Lightning Talks
Wen-Jung Hsin ..... Park University
Bill Siever ..... Washington University in St. Louis
Joseph Kendall-Morwick ..... Washburn University
K-12 Nifty Assignments and Lightning Talks
Bill Siever ..... Washington University in St. Louis
Perla Weaver ..... Johnson County Community College
Belinda Copus ..... University of Central Missouri
Mohammad Rawashdeh ..... University of Central Missouri
Student Paper Session
Scott Sigman ..... Drury University
Wen-Jung Hsin ..... Park University

Mahmoud Yousef ..... University of Central Missouri
Joseph Kendall-Morwick ..... Washburn University

Student Poster Competition

Joseph Kendall-Morwick ..... Washburn University
Ron McCleary ..... Retired

Student Hack-a-thon

Scott Sigman ..... Drury University
Chris Branton ..... Drury University
Mahmoud Yousef ..... University of Central Missouri
Bill Siever ..... Washington University in St. Louis

Student Programming Contest

Charles Riedesel ..... University of Nebraska-Lincoln
Joan Gladbach ..... University of Missouri Kansas City
Brian Hare ..... University of Missouri Kansas City
Tim Urness ..... Drake University

Two-Year College Outreach

Suzanne Smith ..... Johnson County Community College
Trisch Price ..... Johnson County Community College
Mahmoud Yousef ..... University of Central Missouri

Local Arrangements

Eric Manley ..... Drake University
Tim Urness ..... Drake University
Andrei Migunov ..... Drake University

Regional Board — 2025 CCSC Central Plains Region

Regional Rep & Board Chair
Judy Mullins ..... Retired
Registrar & Membership Chair
Ron McCleary ..... Retired
Current Conference Chair
Eric Manley ..... Drake University
Next Conference Chair
Tiffany Ford ..... Ozarks Technical Community College
Past Conference Chair
Kevin Brunner ..... Graceland University
Secretary
Diana Linville ..... Northwest Missouri State University
Regional Treasurer
Ajay Bandi ..... Northwest Missouri State University
Regional Editor
Joan Gladbach ..... University of Missouri Kansas City
Webmaster
Deepika Jagmohan ..... St. Charles Community College

Reviewers — 2025 CCSC Central Plains Conference

Imad Al Saeed ..... Saint Xavier University, Chicago, IL
Rad Alrifai ..... Northeastern State University, Tahlequah, OK
Beth Arrowsmith ..... University of Missouri - St. Louis, Saint Peters, MO
Kevin Brunner ..... Graceland University, Lamoni, IA
John Buerck ..... Saint Louis University, St. Louis, MO
David Bunde ..... Knox College, Galesburg, IL
Karla Carter ..... Bellevue University, Bellevue, NE
Belinda Copus ..... University of Central Missouri, Warrensburg, MO
Aziz Fellah ..... Northwest Missouri State University, Maryville, MO
Ernest Ferguson ..... Northwest Missouri State University, Maryville, MO
David Furcy ..... University of Wisconsin Oshkosh, Oshkosh, WI
Brian Hare ..... University of Missouri-Kansas City, Kansas City, MO
David Heise ..... Lincoln University, Jefferson City, MO
Suvineetha Herath ..... Carl Sandburg College, Galesburg, IL
Charles Hoot ..... Northwest Missouri State University, Maryville, MO
Wen Hsin ..... Park University, Parkville, MO
Jeff Ifland ..... State Of Nebraska, Lincoln, NE
Joseph Kendall-Morwick ..... Washburn University, Topeka, KS
Srinivasarao Krishnaprasad ... Jacksonville State University, Jacksonville, AL
Diana Linville ..... Northwest Missouri State University, Maryville, MO
Baochuan Lu ..... Southwest Baptist University, Bolivar, MO
Eric Manley ..... Drake University, Des Moines, IA
Jose Metrolho ..... Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
Kian Pokorny ..... McKendree University, Lebanon, IL
Hassan Pournaghshband ..... Kennesaw State University, Kennesaw, GA
Charles Riedesel ..... University of Nebraska - Lincoln, Beatrice, NE
Michael Rogers ..... University of Wisconsin Oshkosh, Oshkosh, WI
Jamil Saquer ..... Missouri State University, Springfield, MO
William Siever ..... Washington University, St. Louis, MO
Cindy Tu ..... Northwest Missouri State University, Maryville, MO
Timothy Urness ..... Drake University, Des Moines, IA
Henry Walker ..... Grinnell College (retired), Napa, CA
Maria Weber ..... Saint Louis University, St. Louis, MO
Mudasser F Wyne ..... National University, San Diego, CA
Cong-Cong Xing ..... Nicholls State University, Thibodaux, LA

Accessible Computing: What Every CS Educator Should Know*

Opening Keynote

Dr. Meredith Moore
Assistant Professor of Computer Science, Drake University

Abstract

Computer science education plays a crucial role in shaping the future of technology, but accessibility and inclusion are often overlooked in curricula. This keynote will provide a practical guide to accessible computing, equipping attendees with strategies to create more inclusive learning environments. We will explore why accessibility matters—ethically, legally and pedagogically—and how it benefits all students, not just those with disabilities. Key topics include the difference between teaching accessibility as a subject and teaching accessibly as an educator, the fundamentals of disability language, and the distinction between assistive and accessible technology. Attendees will also gain insight into accommodations, Universal Design for Learning (UDL), and effective ways to support neurodivergent students in CS classrooms. The session will conclude with simple, actionable steps that educators can implement immediately, along with a curated list of resources for those who want to explore accessibility in more depth. Whether you are new to accessibility or looking to refine your approach, this talk will provide the knowledge and tools to make computing education more accessible.

Bio

Dr. Moore is an Assistant Professor of Computer Science at Drake University. Dr. Moore obtained her B.S. in Computer Science and Neuroscience from Drake University in 2015. She then went on to get her Ph.D. from Arizona State University in Computer Science as a National Science Foundation Graduate Research Fellow. Dr. Moore's research focuses on using machine learning to improve the accessibility of technology for individuals with disabilities.

AI Engineering: Lessons from the Frontlines*

Banquet Speech

John Emmons
Director of AI at Salesforce

Abstract

From self-driving cars to generative AI, John Emmons shares the career decisions that took him from the Midwest to Silicon Valley, then back again, and the lessons learned along the way. Drawing from his experience leading high-pressure AI projects at Tesla Autopilot and Salesforce AI, he provides insights into how real-world AI systems are built, the skills that matter most in the industry, and practical advice for those looking to break into the field from the perspective of an AI engineering manager.

Bio

John Emmons completed the engineering dual degree program between Washington University in St. Louis and Drake University in 2016. He went on to get a Master's degree in computer science, focusing on deep learning, from Stanford University in 2018. He spent the next five years as a machine learning scientist at Tesla, eventually leading the Autopilot Vision team. For the last two years, he has served as the Director of AI at Salesforce. When John isn't running AI teams, he like to run long distances on the road and boasts a sub-3-hour marathon time.

Adapt DevOps Method to Information Systems Capstone Course Projects*

Gary Yu Zhao and Cindy Zhiling Tu
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{zhao, cindytu}@nwmissouri.edu

Abstract

This paper explores the design of a DevOps-based information systems (IS) capstone course in a Master of Science in Information Systems program. The researchers aim to investigate this teaching case to gather direct student feedback and provide insights for instructors to adapt the DevOps method in IS capstone projects. The study addresses the research questions and identifies key findings through inductive content analysis of 32 student feedback responses from two consecutive semesters. The preliminary results contribute to the field of information systems education and encourage more instructors to engage with DevOps-based IS capstone projects.

1 Introduction

Many companies in the industry expect higher education institutions to prepare students as future practical experts, enabling them to gain experience in system development and teamwork before entering the IT workforce [2]. Information systems (IS) capstone courses, typically at the end of a curriculum, help students gain and demonstrate relevant skills and insights [9]. By gaining sufficient practical experience with the capstone courses, students can avoid

the costs and effort of exploring uncertain career paths and become valuable assets to project teams immediately upon joining the workforce [3].

In the concurrent IS education context, "project-based learning" (PBL) has been widely adopted to explore the development of students' comprehensive hands-on skills and experience in system analysis and design, project management, and team collaboration [3]. IS students take courses in technical subjects like programming, networking, cybersecurity, and databases, along with business and management topics such as project management and financial modeling. They also work on UX design, requirements gathering, and systems diagrams through student projects. These courses help students build the skills needed for their capstone project, where they apply technical, management, and professional skills to complete the project and interact with clients [15]. The IS capstone course encompasses the full cycle of information systems development, from conceptualization and design to prototyping or implementation. Students typically work in small teams on independent projects, encouraging self-directed learning through structured reflection. Given the small group size and short time frame, many instructors adopt agile methodologies which emphasize adaptive planning, iterative development, early delivery, and continuous improvement [15]. A key element of success in Agile-based capstone projects is client communication and rapid feedback. However, Agile methodology emphasizes improving the speed and flexibility of development and focuses less on the collaboration between developers and IS end-users – operations teams.

In recent years, DevOps, as the next step of agile, has become one of the mainstream methods in software development projects. DevOps is a software development methodology that integrates development and IT operations to enhance the speed, quality, and reliability of software delivery [1]. It fosters a culture of collaboration, allowing project stakeholders and developers to work closely together to ensure software meets business needs. Traditional software development methodologies employed in IS Capstone projects also emphasize user involvement, such as Waterfall, RAD, Agile, etc. However, DevOps has advanced previous methods and focuses on the cooperation between development and user operations. Based on the features of IS capstone projects, is DevOps suitable? What factors affect the successful application of the DevOps method in the IS capstone course? Little research has been done on this topic. Our study focuses on the application of the DevOps method on students who executed capstone projects by conducting a case study.

2 Theoretical Background

2.1 DevOps Methodology

DevOps, which stems from software engineering, merges development and operations practices. DevOps emphasizes automation, streamlining tasks like testing and deployment, enabling system analysts to focus on analyzing requirements [7]. The methodology promotes continuous improvement, with system analysts providing feedback to refine the development process. Automated testing and continuous integration ensure the software is reliable and aligned with business goals, while faster release cycles help deliver timely updates [1]. The integration of DevOps enables faster, higher-quality software delivery, which supports business growth and improves client satisfaction[5]. Through these processes, the project team can ensure both the effectiveness and timeliness of software solutions[4]. DevOps is a relatively recent practice that has been widely adopted in the software industry. However, in the context of IS education, the focus tends to be more on teaching development (Dev) skills than operational (Ops) skills. As a result, while DevOps is well-established in the software industry, there are limited reports of its use as a teaching method in the IS education context [5].

2.2 Project-based Learning

Project-based learning (PBL) has attracted significant attention, especially in IS education, as educators look to improve classroom instruction using simulated or real-world projects. Previous research has largely focused on how PBL affects students' learning outcomes, both mentally and physically, with many studies relying on self-reported feedback from students [8]. [14] also argued that the practical sessions may be a possible channel to improve students' skills in project management. [7] introduced a distinctive approach by incorporating DevOps-based learning into PBL. Building on this, we adopt a similar approach by integrating DevOps into IS capstone course, focusing on local real-world projects to further increase student motivation and engagement.

Figure 1 shows the conceptualized processes of the IS capstone project adapting the DevOps method. The end users – operations teams work with the capstone project team through all the project phases, including business requirements engineering, system analysis, design, prototype, implementation, testing, deployment, and feedback.

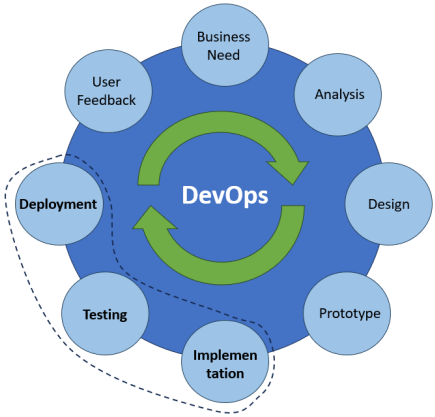


Figure 1: Conceptual framework of adapting DevOps to IS capstone projects.

3 Case Study

We conducted a case study using semester-long observations, informal interviews, and surveys. The case study is a suitable method for this research as it helps identify potential opportunities and rich elements from participants' behavior in specific, observable settings [12]. The study was conducted at a regional public university in the Midwest, where a Master of Science in Information Systems (MSIS) program includes a capstone project course. The study was carried out during the spring 2024 from January to May and fall 2024 semester from August to December. One university department served as a client for developing a campus recycling management system. A total of 32 students were assigned to 6 project teams and went through the information systems development processes. In Spring 2024, project teams worked on collecting user requirements, system analysis, design, and prototype. In Fall 2024, the teams completed system implementation, testing, and deployment.

All student teams were required to study and apply DevOps method in their real-world projects. In each semester, the first two weeks were for project kick-off and DevOps training/reviewing stage. The client's operational staff and manager hosted kick-off meetings with all project teams and introduced the system's business needs. Then, 3 sprints were scheduled for system development, with 4 weeks duration for each sprint. Projects were wrapped up in the last two weeks. During each sprint, project teams worked with client operation staff closely, completed a specific set of tasks, and demonstrated the sprint outcomes or prototypes to the client. The client operations staff collaborates with project teams through interviews, emails, MS Teams, and face-to-face

meetings. The client's cooperation with teams facilitated valuable feedback, enabling teams to refine their approach for the next sprint. To conclude the project, teams presented the final products to the client with a final report. The implemented system was deployed on the university application server. Table 1 shows the technologies and tools being used in the DevOps-based capstone projects.

Table 1: Technologies and Tools used in the DevOps-based Capstone Projects

Project Tasks	**Technologies**
Project management	Trello, Jira
System analysis and design	LucidChart, Mockups, Visual Studio
Prototype	Wix, Figma
Implementation tools	GitHub, GitHub Co-Pilot
Frontend	HTML, CSS, JavaScript, React.js, Vue.js
Backend	Node.js, C#, Python, Java, MySQL, REST APIs, SQL Server Management Studio (SSMS), SQLite, MongoDB
Testing	SoapUI, Automated Python scripts
Deployment	Docker, University application server
Observation and feedback	Slack

During the semester, periodic observations were conducted to assess the collaboration between students and the client in the DevOps-based capstone projects. The study focused on the growth of student's learning motivation and outcomes, which were measured through qualitative feedback collected through interviews and a survey. A questionnaire survey was completed close to the end of the second semester to gather feedback on the impact of DevOps on Capstone projects and project team members. The survey includes 30 questions with a 5-lickert scale covering 7 aspects. A total of 32 feedback responses from students over two semesters were gathered and analyzed. To show the results clearly, we aggregated the 5 scales into 3 scales (Agree, Neutral, Disagree). Figure 2 shows the results.

4 Findings and Discussion

The DevOps-based IS capstone course design was well-received by students. The main findings demonstrate how this course structure enhances students' learning outcomes and motivation and what factors primarily affect the DevOps teams.

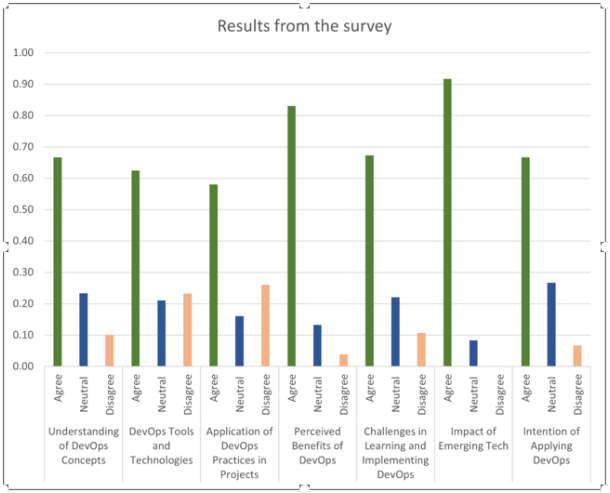


Figure 2: Results of student survey.

4.1 Positive Impact on Student Team Members

The student feedback, alongside discussions with clients, helped adjust teaching strategies and highlighted the benefits of DevOps in strengthening ties between the classroom and real-world businesses. Overall, the project team members were very satisfied with the progress, output artifacts, teamwork, and cooperation with the operations staff. From the learning outcome perspective, the students emphasized the experience and skills they gained from this DevOps-based capstone course.

"Working in a DevOps team taught us the importance of collaboration between development and operations. It broke down silos and helped us communicate more effectively, making our project run smoothly."

The interview feedback highlights that DevOps teams provide hands-on training, integrating students or new team members into the necessary activities, as regular courses often do not teach all the competencies required for DevOps roles. DevOps teams foster a more interconnected environment where individuals develop broader, more generalized skills [4]. Figure 2 shows that 67% of students gained a better understanding of DevOps concepts, and about 60% of students confidently applied DevOps to their projects.

"Using tools like Jira, Docker, and GitHub in our project gave us valuable

exposure to industry-standard technologies, which I'm sure will be helpful in future job roles."

Above feedback emphasizes the benefits of DevOps tools and automation used in implementation, defect validation, and deployment, allowing developers to quickly and easily configure environments without manual intervention. 63% of students use DevOps-related technologies and tools effectively (Figure 2). By adopting DevOps to optimize processes, teams can achieve greater efficiency and improved outcomes[10].

"The iterative nature of the DevOps process helped us continuously learn and improve as we received feedback and made improvements in real-time."

The practice of DevOps teams facilitated large-scale learning and knowledge sharing as team members gained experience in multiple roles, enhancing their skills and autonomy [11]. The single-case study showed that DevOps contributes to innovation in software development by promoting adaptability and continuous learning, which supports the implementation of innovative practices[4]. Consistently, Figure 2 shows that about 70% of students have the intention of applying DevOps in their future projects.

4.2 Success Factors of DevOps-based Capstone Project

Based on feedback from student DevOps project teams, the following key factors were identified as critical to the success of a DevOps-based capstone project:

"I appreciate the fit scope of the project, as it allowed us to gain experience in managing tasks similar to those we will encounter in future work. Overall, the course was well-planned, and I have learned a great deal."

One challenge in using project-based learning (PBL) to enhance motivation is determining an appropriate project scope. A narrow scope limits practical application, while an overly broad scope makes timely completion difficult. An adequately scoped project tailored to the course level and student experience is critical to the DevOps-based PBL success. This balance allows students to gain practical experience, work with client operations staff, and gain deeper course understanding while ensuring they can complete the project within two semesters. These findings align with prior research[3], [7]and are particularly relevant in information systems courses. Furthermore, the survey shows that about 70% of students agreed that more structured training or courses help them handle the challenges of learning and implementing DevOps in the cap-

stone projects (Figure 2).

"Receiving constant feedback from the instructor and team members allowed us to continuously improve our work continuously, leading to better results by the end of the project."

Both literature and interviews with DevOps practitioners identify building a collaborative culture as the primary critical success factor in DevOps adoption. While adopting new tools is relatively straightforward, changing the culture within software development teams is more challenging due to differing perspectives between developers and operators[4]. The feedback emphasized the importance of breaking down silos and promoting effective communication and cross-functional collaboration.

"Defining roles from the beginning and holding each team member accountable ensured that tasks were completed on time and to a high standard."

Teams found that assigning clear responsibilities for both development and operations tasks helped avoid confusion and redundancy. Defined roles help optimize the use of resources, ensuring that team members with specific expertise handle the tasks best suited to them. This contributes to both productivity and cost-effectiveness. Accountability ensures that tasks are completed on time and with quality, contributing to the overall success of the project. Additionally, learning how to manage time in a DevOps setting mimics real-world scenarios, enhancing their readiness for industry roles where deadlines and efficiency are critical. Student DevOps teams must ensure project success, balance academic demands, and develop the discipline needed for future careers [6].

"DevOps taught us to be flexible and adaptable. When things went wrong, we were able to quickly make adjustments and continue moving forward."

DevOps involves using a wide range of tools for continuous integration, deployment, monitoring, and collaboration[13]. These tools evolve rapidly, and new ones frequently emerge. As shown in Figure 2, about 60% of team members can use Dockers, GitHub, CI/CD pipelines, and Python scripting effectively through the capstone projects; about 70% of members believe that they can handle the DevOps solutions based on the cloud service by using these tools. Moreover, student teams must be flexible and resilient to quickly adapt to changing technologies and workflows, which is key to their success in real-world environments. More than 90% of students agreed that emerging technologies, such as AI, machine learning, and data-driven, will enrich and

advance DevOps practice (Figure 2).

4.3 Improve Learning Motivation and Outcomes

"Communication between team members from both the development team and operations was essential. This communication encourages our student teams to work dedicatedly, learn efficiently, and achieve the final goals."

Student feedback highlights that the operations staff's passion for communicating and instructing significantly influences their learning motivation. Many students appreciated the client team's proactive and enthusiastic approach, noting their commitment to course planning and student outcomes. Over half of the students stated that the operations team's passion inspired them to work harder on lectures and projects. In the DevOps-based capstone project, which requires significant effort, the operations team serves as an essential role model, assisting students through the complex process. Research typically focuses on role models in primary or secondary education, but this study reaffirms the importance of passionate project clients in motivating students, particularly in IT-related majors[3].

Students often worry about how to apply theoretical knowledge from lectures to real-world scenarios, particularly as many plan to enter the workforce after graduation. In response, the course created an immersive learning environment that fosters intensive discussions among students, instructors, and client operations staff. These discussions help students identify weaknesses, clarify issues in IS system development, and practically apply course concepts. Feedback emphasized the value of such interactions, with students noting that discussions enhanced their understanding and flexibility in using course knowledge. Additionally, hands-on learning, particularly through real projects, allowed students to develop problem-solving skills and gain practical experience. This approach has successfully bridged the gap between theory and practice, fostering collaboration and preparing students for their future careers. As shown in the survey (Figure 2), over 80% of students agree that DevOps-based capstone projects provide valuable opportunities for continuous learning and skill development.

5 Conclusion

This study successfully implemented a practical DevOps-based information systems capstone project in collaboration with a real-world client, integrating both IT and management knowledge to improve performance. A case study examined the impact of DevOps on project success and explored key factors influencing capstone project outcomes. The study found that applying DevOps

to capstone projects yields benefits similar to those of information systems development. It also identified various factors crucial to success. The effective DevOps-based course design model could be extended to other information systems courses and project-based initiatives aligned with university social responsibility goals.

References

[1] Mohammed Airaj. "Enable cloud DevOps approach for industry and higher education". In: *Concurrency and Computation: Practice and Experience* 29.5 (2017), e3937.

[2] Chin-Ling Chiang and Huei Lee. "The effect of project-based learning on learning motivation and problem-solving ability of vocational high school students". In: *International Journal of Information and Education Technology* 6.9 (2016), pp. 709–712.

[3] Chih-Yuan Chou. "A DevOps-based Service-Learning Design in An Advanced IS-related Course". In: *Proceedings of the International Conference on Information Systems (ICIS)*. Jan. 2023, pp. 1–9.

[4] Joao Faustino et al. "DevOps benefits: A systematic literature review". In: *Software: Practice and Experience* 52.9 (2022), pp. 1905–1926. DOI: 10.1002/spe.3096.

[5] Marcelo Fernandes et al. "Challenges and Recommendations in DevOps Education: A Systematic Literature Review". In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. SBES '20. New York, NY, USA: Association for Computing Machinery, Dec. 2020, pp. 648–657. DOI: 10.1145/3422392.3422496.

[6] Alexandre Grotta and Edmir Parada Vasques Prado. "DevOps Didactic Transposition in IS Higher Education: A Systematic Literature Review". In: *AMCIS 2022 Proceedings*. 2022, p. 15. URL: https://aisel.aisnet.org/amcis2022/sig_ed/sig_ed/15.

[7] Alexandre Grotta and Edmir Parada Vasques Prado. "DevOpsBL: DevOps-based Learning on Information Systems Higher Education". In: *AMCIS 2021 Proceedings*. 6. 2021. URL: https://aisel.aisnet.org/amcis2021/is_education/sig_education/6.

[8] Pengyue Guo et al. "A review of project-based learning in higher education: Student outcomes and measures". In: *International Journal of Educational Research* 102 (2020), p. 101586.

[9] Eric H Hobson, Philip E Johnston, and Alisa J Spinelli. "Staging a reflective capstone course to transition PharmD graduates to professional life". In: *American Journal of Pharmaceutical Education* 79.1 (2015), p. 14.

[10] Rajitha Mawananehewa Jayasekera and Shantha Jayalal. "Factors Influencing the Utilization of Cloud Optimization Tools Among DevOps Engineers: Insights From a Software Development Company in Srilanka". In: *International Journal of Vallis Aurea* 10.1 (Jan. 2024), pp. 39–56. DOI: 10.62598/JVA.10.1.4.4.

[11] Michael Maloni, Pamila Dembla, and J Anthony Swaim. "A cross-functional systems project in an IS capstone course". In: *Journal of Information Systems Education* 23.3 (2012), p. 283.

[12] Michael D Myers. *Qualitative research in business & management*. Thousand Oaks, CA: Sage Publications Ltd, 2009, pp. xii, 284.

[13] Mojtaba Shahin, Ali Rezaei Nasab, and Muhammad Ali Babar. "A qualitative study of architectural design issues in DevOps". In: *Journal of Software: Evolution and Process* 35.5 (2023), e2379. DOI: 10.1002/smr.2379.

[14] Harold Smith III, Debra Smarkusky, and Elizabeth Corrigall. "Defining Projects to Integrate Evolving Team Fundamentals and Project Management Skills". In: *Journal of Information Systems Education* 19.1 (2008), pp. 99–110. URL: https://aisel.aisnet.org/jise/vol19/iss1/10.

[15] Cindy Zhiling Tu and Joni Adkins. "Effect of user involvement in information systems capstone course: a case study". In: *Journal of Computing Sciences in Colleges* 35.6 (2020), pp. 107–116.

Improving Student Success Through Parachuting*

Chris Bourke and Nirnimesh Ghose
School of Computing
University of Nebraska–Lincoln
Lincoln, NE 68588
{chris.bourke, nghose}@unl.edu

Abstract

We report on our experience implementing a *parachute program* that identifies struggling students in a Computer Science I course and invites them to "parachute" out and start over mid-semester in a lower-level computing course that still confers credit toward their degree program. This program gives students a second chance at success, aims to improve student outcomes in early computing curricula, and to improve retention overall in computing majors.

1 Introduction

Many students struggle in introductory STEM courses. Within computing disciplines the number of DFW students (those receiving a non-passing grade of D or F or withdrawing from the course) has remained consistent across time, delivery mode, programming language, etc. [15]. Much research and effort has been done in an attempt to address this issue and to improve student outcomes. Introductory computing courses are especially challenging because of the non-uniform and non-standard coverage of computing in high schools. Students come into college with highly varying backgrounds in computing.

We present a novel approach to this problem through a *Parachute Program* that has been used at the University of Nebraska–Lincoln's School of Computing for the last two years. Through a combination of self- and instructor-based assessments, struggling students are identified at pre-defined milestones in a Computer Science I (CS1) course. Students who are at risk of failing are invited to "parachute" out of the CS1 course and safely land in a lower-level Computer Science 0.5 (CS0.5). In this new course they are given the opportunity to start over.

The goal of this program is to improve student outcomes overall. For struggling students in particular, it aims to improve retention by strengthening their foundation and giving them a chance to acclimatize themselves to college life.

2 Related Work

Methods have been tried across many STEM disciplines to address struggling students. Placement exams are commonly used to place students into an appropriate introductory course that matches their background knowledge and readiness. However, these exams are far from perfect. Underplacement (placing well-prepared students in lower level courses) is far more common than overplacement (placing students in a course they are not prepared for) [13]. Placing students into remedial courses can have wide-reaching and long-term negative impacts [7]. Taking a remedial course also has an an impact on time-to-graduation and disproportionately affects under-represented minority students. Not only do remedial courses have extensive costs to institutions and students, the overall research on their benefit is mixed at best [2].

Other efforts attempt to identify struggling students while they are taking a course. Math departments have been administering "gateway" exams since at least the 1980s which have evolved over time. The key difference with gateway exams is that they are given within a course and intended to measure skills which every student in a course should develop rather than whether or not they are prepared to take the course [10]. Reporting indicates passing rates are typically high (though initially may be as low as 50% the first time a student takes it). However, this is in contrast to the typical failure rates of college math courses [4] which are as high as 27% in Calculus I [5].

In addition, though a gateway exam may be part of a student's grade, they only serve as a guide to the student and instructor as to whether or not they have learned the material that is expected of them or whether they are on the right trajectory to complete the course. Whether or not they pass a gateway exam does not affect a student's enrollment in the course.

Within computing disciplines several placement exams have been developed [14, 12]. These efforts have faced challenges because the typical CS-related K–

12 curriculum is not nearly as standard as (say) math and may not exist at all. As a consequence, students enter college with a much wider variety of prior computing experience.

However, these tests have been used mostly to determine whether or not a student should be placed in CS1 (low or no prior experience) or is prepared enough to start off in CS2. This is because historically most computing curricula only have these two options for introductory courses. With the explosion in computing enrollment and the expansion of computing sub-disciplines (CS+X, data science, informatics, etc.), this is less true. Many institutions have started to offer a wider variety of introductory courses catering to different student interests, backgrounds and goals.

A recent effort has been made to adapt and extend existing placement exams as an instrument to assess a student's preparedness to take CS1 or to take an alternative introductory course [3]. Even if no validated instrument is used, some departments have started offering multiple sections of CS1 targeted toward students with differing prior computing experience [9]. Though they may cover the same topics, these courses may differ in the delivery or assessment or offer more or alternative opportunities for practice and collaboration.

No evidence could be found that this parachute program or any similar program is used by other institutions. Nevertheless we mention that the parachute program was inspired by a similar program that was supposedly used in some chemistry departments (folklore).

3 Parachute Program

The parachute program was first piloted in spring 2023 and involved several significant changes to curriculum and to the structures, schedule, and topics of courses.

3.1 Courses

The School of Computing at the University of Nebraska-Lincoln offers a wide variety of introductory computing courses. The two most relevant courses to the parachute program are CSCE 101 and CSCE 155A.

CSCE 101 is a CS0.5 course in that it is more than a rudimentary computing skills course but not fully a CS1 course. It emphasizes problem solving and introduces basic python programming. As a service course, it is intended mostly for non-computing majors as it satisfies general education requirements. Traditionally 101 has been a terminal computing course. As a consequence, enrollment is extremely diverse, attracting students from all levels and dozens of different majors.

CSCE 155A is a full CS1 course [1] intended for computing majors. It covers problem solving and software development principles. Enrollment consists mainly of incoming freshman computing majors serving as their first college-level computing course. There is no computing prerequisite and no prior computing or programming experience is necessary or expected. This course introduces computational thinking by solving puzzles. Students are introduced to programming by first writing formulas in Google Sheets and then developing rudimentary programs to solve real-world examples using Coral [6]. Once the students are comfortable using computational thinking to write programs, Python is introduced using a flipped classroom model. The students learn the basics before the class using an interactive, auto-graded textbook. Class time is spent on live coding demonstrating the problem solving process on problems and puzzles from the well-known Algorithmic Puzzles book [11]. For example, when recursion is covered, class time is used to develop code to solve the three jugs puzzle (given an 8-pint jug full of water and two empty jugs of 5- and 3-pint capacity, get precisely 4 pints of water in one of the jugs by filling up and emptying jugs into others) using recursion. Students are then assessed using a more complex problem for which they develop a full program. For recursion, students are challenged to develop a program to draw the decision tree for the four knights puzzle [8].

Both courses are offered as 15-week semester courses in fall and spring semesters and are synchronized with the same weekly lecture and lab schedules.

3.2 Process

Student progress in CSCE 155A is closely tracked in the first five weeks. Assessments are evenly distributed throughout the semester so that approximately one third of the assessment has been completed by the end of the 5th week. In the first week students are given a Computing Skills Inventory "exam" [3] which is graded and gives both the student and the instructor a baseline on whether or not the student is likely to succeed (pass) in 155A.

In week 5 students whose grade is less than 73% (C) are invited to join the parachute program and switch their enrollment to CSCE 101. The invitation is entirely voluntary. Those that opt-in to the parachute program are re-enrolled into 101 and "restart" their introductory course in week 6. The re-enrollment and administrative processes are handled entirely by faculty and staff. Those that choose not to opt-in continue their enrollment in CSCE 155A. In all cases, tracking of student progress continues through the end of the semester.

3.3 Curricular Changes

Prior to adopting the parachute program, several curricular changes were made. First, CSCE 101 was reorganized so that the first 5 weeks consisted of general computing topics and computational thinking but no formal programming. Introduction to coding (Python) was moved to begin in week 6. Second, curriculum rules were changed to allow CSCE 101 to count toward degree requirements of computing majors *provided* that it was completed before any other computing courses. Prior to this, it did not count toward any computing degree (other than general credit hours).

These two changes were necessary to make the parachute program possible. Students being parachuted into 101 from 155A would have been completely lost if simply dropped into a course 5 weeks into the semester. By front-loading the course with general computing topics, parachuted students get a fresh (re)start in 101 at the point that Python is being introduced in both courses.

It was also necessary to ensure that 101 counted toward a computing degree program so that credit hours were not lost by the student. This change also mitigates the impact that parachuting has on the time-to-graduation for computing students.

4 Results

The parachute program began in spring 2023 and has been offered for four semesters total. During this period, enrollment and DFW rates (see Table 1a) have been fairly static with overall rates being 15.25% and 37.37% for CSCE 101 and CSCE 155A respectively.

The number of invitees and those who accepted the invite to parachute from CSCE 155A to CSCE 101 can be found in Table 1b. Spring 2023 was the initial pilot for this program and so was more targeted with fewer invitees. The number of invitees has grown as adjustments and refinements to the process have been made. The number of students accepting the parachute opportunity is somewhat volatile but consistent with respect to on/off semesters (the bulk of incoming freshman students are in fall while spring has a more diverse enrollment of transfers, retakes, non-majors, etc.).

For students that accepted our invitation the outcomes were very positive. They successfully integrated into CSCE 101 and were able to perform fairly well. The majority (20/25) passed mostly with A (11) or B (6) grades. Nevertheless some were not able to make the transition and either failed (1) or ultimately withdrew (4).

In contrast, those that declined our invitation (37) to parachute did extremely poorly. While one student was able to achieve a B and 6 were earned a C, the vast majority earned grades of D (6) or F (13) or ultimately withdrew

Table 1: Course Data during the parachute program from Spring 23 (S23) through Fall 2024 (F24)

	CSCE 101		CSCE 155A	
Sem	Enr.	DFW	Enr.	DFW
S23	131	18.32%	69	34.78%
F23	99	18.18%	132	35.61%
S24	157	12.74%	86	36.05%
F24	118	12.71%	85	43.53%
Total	505	15.25%	372	37.37%

(a) Semester-by-semester enrollment (Enr.) and DFW rates

Sem	Invited (%)	Accepted (%)
S23	4 (5.80%)	4 (100%)
F23	14 (10.61%)	7 (50%)
S24	24 (27.91%)	3 (12.5%)
F24	20 (23.85%)	11 (55%)
Total	62 (16.67%)	25 (40.32%)

(b) Parachute program invitees and accepts(numbers and percentage of the class) by semester.

(11) giving an overall DFW rate of 81.08%, far exceeding typical failure rates. A sankey flow diagram of these outcomes can be found in Figure 1.

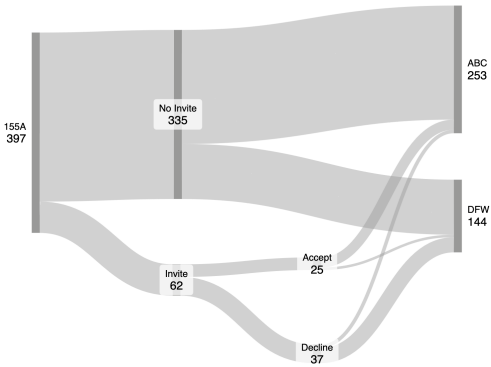


Figure 1: Flow of students through CSCE 155A and the Parachute Program.

Beyond the success rates of these introductory courses, one of the aims of the parachute program is to retain more students in computing majors. Unfortunately, among students prior to fall 2024 that received an invitation to participate (42) nearly half (20) had dropped or failed out of college entirely (though it is possible some transferred to another institution). Nevertheless, among those still enrolled, nearly one third (7/22) persisted in their computing major and were still matriculating a year later. Though the remaining students found different majors (15/22), some continued with computing as a minor. Though it is still early, data from fall 2024 provides a bit more optimism. More students in this cohort (7/11) are continuing with computing courses and all are still enrolled.

5 Discussion

The numbers presented in this paper are admittedly small. However, we are making no claims of generalization or formal methodological analysis. The parachute program attempts to engage with students who are already on the

margin and so the numbers will necessarily be small. Nevertheless, our experiences and data over the last two years give several interesting points of discussion. Our experiences have also led to refinements and we've identified some best practices.

5.1 Common Outcomes

Given that students who accepted our invite did well and those that declined tended to fail, an obvious first question is: why did students decline our invitation? Unfortunately, we don't have much in the way of direct data to answer this question. A small number of students were *unable* to accept the invitation because they had already taken and passed the parachute course or they were non-computing majors whose program required 155A (or would not count 101). We do have some anecdotal evidence that students wanted to persist with CSCE 155A and intended to improve, a sort of sunken cost fallacy. However, more commonly we observed that students who were already failing at that point were completely disengaged with the course to the point that they ignored even our outreach efforts.

This theme continued when we looked at the near-term data. About half of all students who declined parachuting had failed or dropped out of our institution entirely within the next year. This suggests that these students were experiencing problems that went well beyond these courses and the computing curriculum. Most had failed more than one course or even the majority of their courses.

Nevertheless, the parachute program did result in significant positive outcomes. It was able to successfully retain a good number of students who may not have persisted in computing or in college at all. Given the minimal investment in administrative time and effort, these are very positive outcomes.

Though retention in computing majors is certainly a goal, it is not the only positive outcome. Many students switched to other disciplines (which is common) but still earned credit for the parachute course toward their degree program. This program serves as an easy "off-ramp" to students who were not entirely set on a computing major. They were able to get a taste of computing and decide that it was not for them without losing out on the credit hours or negatively impacting their GPA. These students also have the potential to return to computing later on or in alternative ways such as receiving a minor.

Though it was not observed directly, the invitation process itself may serve as a wake-up call to students who are under-performing early in the course and can serve as an early intervention mechanism.

5.2 Refinements

Though the program is only two years old, several refinements and adjustments have been made. Only about a third of the invitees are represented in the final DFW students. That is, a good majority of DFW students were not initially identified and invited to parachute in the first 5 weeks. This suggests that students do not necessarily "fail fast" in this course. To address this, we have adjusted the cutoff for invitees downward over time in an effort to capture more students who would otherwise fail. In the most recent offering of the course, students meeting the parachute criteria were contacted directly with an invitation, however, the entire class was made aware of the program. Students who were doing well or reasonably well were also given the opportunity to opt-into the program. Several students took this opportunity citing stress, workload, or perceived difficulty of the course as reasons.

In Fall 2024, the program was expanded to include another CS1 section designed for students with some computing background. However, among 8 students invited, all declined and all eventually failed. Nevertheless, we intend to continue and expand the program going forward.

5.3 Best Practices

We see the curricular changes discussed in Section 3.3 as necessary to a program like this. If students are to be re-enrolled in a different course during a semester, it is absolutely essential that the course count toward their degree program in a significant way. Without this change, parachuting would only exacerbate time-to-graduation challenges. Any program considering a change like this needs to reexamine and potentially realign their early curriculum as a first step.

Recall that it was necessary to ensure that the content of the parachute course, CSCE 101, be realigned to accommodate students entering the course at week 6. However, our CS1 course content also had to be aligned so that assessments were more evenly distributed using smaller assessments and front-loading the course so that early success or risks could be established. If the course had a more traditional structure, say heavily exam-based, trends in assessment could not have been observed.

Another necessary component is clear and frequent communication. Students need to know if they are doing well in the course even before the 5 week invite period. The invitation also needs to be crafted so that it is not an indictment of the student's performance but instead framed as a way to provide them help and an opportunity to succeed.

Now that we have a good amount of data, we intend to update our communication strategy to provide more transparency with respect to that data. In particular, we intend to share general success rates of those who choose to

parachute versus those that do not.

Though not a best practice, another necessary component to a program like this is administrative buy-in. When this program was originally proposed there was initial concern that students could not be administratively re-enrolled mid-semester. There were questions as to the impact on financial aid, scholarships and other legal issues. As a result, the program, though approved, is required to be voluntary for students. Administrative support was also necessary to handle the logistics of re-enrollment and to prevent late enrollment fees.

6 Conclusion

As with any intervention, change and improvement are incremental; there is no silver bullet. The results of our parachute program are somewhat mixed, but the positive results are compelling especially given the minimum amount of work required to implement and administer it. In doing so, it has also led to other positive curricular changes. This paper has outlined the general process, its results, and best practices so that other institutions might think about adopting or adapting a similar program.

References

[1] Brett A. Becker and Keith Quille. "50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 338–344.

[2] Angela Boatman and Bridget Terry Long. "Does Remediation Work for All Students? How the Effects of Postsecondary Remedial and Developmental Courses Vary by Level of Academic Preparation". In: *Educational Evaluation and Policy Analysis* 40.1 (2018), pp. 29–58.

[3] Ryan Bockmon and Chris Bourke. "Validation of the Placement Skill Inventory: A CS0/CS1 Placement Exam". 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. 39–45.

[4] B. Bonham and H. Boylan. "Developmental mathematics: Challenges, promising practices, and recent initiatives". In: *Journal of Developmental Education* 34.3 (2011), pp. 1–10.

[5] D. M. Bressoud, V. Mesa, and C. L. Rasmussen. "Insights and recommendations from the MAA national study of college calculus". In: *Mathematics Teacher* 109.3 (2015).

[6] Alex Daniel Edgcomb, Frank Vahid, and Roman Lysecky. "Coral: An Ultra-Simple Language For Learning to Program". In: *2019 ASEE Annual Conference & Exposition*. 10.18260/1-2–32550. Tampa, Florida: ASEE Conferences, June 2019.

[7] National Center for Education Statistics. *National Postsecondary Student Aid Study 2020 (NPSAS:20)*. 2020. URL: `https://nces.ed.gov/surveys/npsas/`.

[8] Mehdi Iranpoor. "Knights Exchange Puzzle—Teaching the Efficiency of Modeling". In: *INFORMS Transactions on Education* 21.2 (2021), pp. 108–114.

[9] Michael S. Kirkpatrick and Chris Mayfield. "Evaluating an Alternative CS1 for Students with Prior Programming Experience". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 333–338.

[10] P. Gavin LaRose and Robert Megginson. "IMPLEMENTATION AND ASSESSMENT OF ON-LINE GATEWAY TESTING". In: *PRIMUS* 13.4 (2003), pp. 289–307.

[11] Anany Levitin and Maria Levitin. *Algorithmic puzzles*. Oxford University Press, 2011.

[12] Markeya S Peteranetz and Anthony D Albano. "Development and Evaluation of the Nebraska Assessment of Computing Knowledge". In: *Frontiers in Computer Science* 2 (2020), p. 11.

[13] Judith Scott-Clayton, Peter M. Crosta, and Clive R. Belfield. "Improving the Targeting of Treatment: Evidence From College Remediation". In: *Educational Evaluation and Policy Analysis* 36.3 (2014), pp. 371–393.

[14] Allison Elliott Tew and Mark Guzdial. "The FCS1: A Language Independent Assessment of CS1 Knowledge". In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. Dallas, TX, USA: Association for Computing Machinery, 2011, pp. 111–116.

[15] Christopher Watson and Frederick W.B. Li. "Failure Rates in Introductory Programming Revisited". In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*. ITiCSE '14. Uppsala, Sweden: Association for Computing Machinery, 2014, pp. 39–44.

Getting Your Hands Dirty: Teaching an IoT Course with an Interdisciplinary Component*

Nathan W. Eloe[1] and Alexander W. Taylor[2]
[1]School of Computer and Information Systems
[2]School of Agricultural Sciences
Northwest Missouri State University
Maryville, MO 64468
{nathane, ataylor}@nwmissouri.edu

Abstract

Project-based computer science classes are nothing new; considerable effort goes into creating relevant and realistic educational experiences for students to learn and demonstrate mastery of core learning outcomes. Finding projects that can engage and interest students while still meeting required educational outcomes can be a difficult task, but local opportunities can lead to successful student experiences. This paper details the authors' experiences in adopting an interdisciplinary approach to a project-based Internet of Things course. The intent is to expose students to other areas to drive engagement and interest while also benefiting the larger university community.

1 Introduction

1.1 Related Work

Project-based learning is not a new approach to teaching computer science [3], and efforts to improve measurement of learning outcomes and construct frameworks to administer such classes within the changing academic landscape are

ongoing [1]. Two particularly important factors to the success of such projects are student motivation [3] and how well the project maps to the real world. Student engagement can be driven in part by providing the project the students feel a connection to or by improving their sense of community [6]. Real world project mapping can be addressed by integrating internships into the curriculum [2], but this approach does not scale well to a full class experience. This paper reports the experiences of using a project of importance to the university community while crossing curricular borders in an Internet of Things class.

1.2 Motivation

In the winter months of 2022, the campus greenhouses suffered a heater failure, leading to a catastrophic loss of plants. While monitoring hardware exists in the greenhouses, the manager was not aware of the outage at the time and was unable to react. This led to the idea of targeting the Internet of Things class project at creating a distributed sensor platform that allowed for remote alerting of greenhouse staff in the event of an outage. Over the summer, the structure of the class was planned out and executed in Fall 2023.

1.3 Course Structure

The first half of the class focused heavily on the "things" part of the Internet of Things; programming microcontrollers to react to external stimuli and perform actions based on that stimulus. About six weeks into the semester, the class took a field trip across campus to the Horticulture Complex where they participated in a brief lecture on the importance of greenhouses, the parameters that will be measured (pH, temperature, humidity and light levels) and a tour of the various types of greenhouses that are present at the University. Two exemplars were also presented to highlight the importance of controls on greenhouses (phalaenopsis orchids and habanero peppers). From there, the students were able to design and implement sensor platforms using hardware purchased by the School of Agriculture and 3D printed enclosures.

1.4 The Project

The platforms were built around the ESP32 microcontroller and used the ESPNow [4] mesh technology for communication with a base station. The class of 30 students (primarily juniors and seniors) were divided into groups; each group was responsible for constructing a sensor for a greenhouse and one group was assigned the task of building the base station. Figure 1 shows a block diagram of the system, and Figure 2 shows a deconstructed prototype. Students

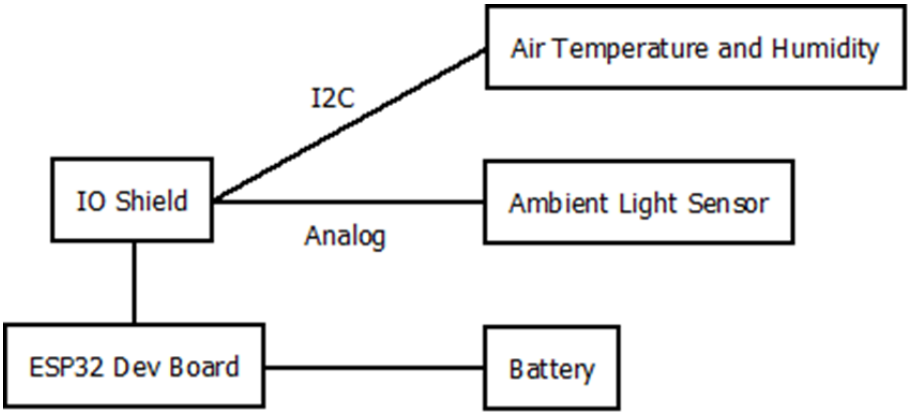


Figure 1: Block diagram of the platform sensors

were provided hardware similar to that used in class and datasheets and given the freedom to explore solutions to a variety of problems not usually encountered in purely software–based projects. Several days throughout the rest of the semester were deemed "greenhouse days" where the class met at the Horticulture Complex to test and improve their implementations. Throughout the second half of the semester, one day would be spent on course content (specifically the "Internet" part of Internet of Things) and the other class day spent working both in their groups and helping other groups.

2 Results and Student Feedback

While very few quantitative results can be shared the project was mostly successful as a prototype of both the sensor system and a class structure. The final product worked with some issues primarily centered around the choice of wireless networking protocol and power in the 2.4GHz band. This results presentation focuses on the project deliverables, qualitative student feedback, faculty observation, and university reception.

2.1 Project Deliverables

The students were able to work together to provide an implementation that was primarily limited by the choice of wireless networking protocols and the structure of the greenhouses and Horticulture Complex. While there were some issues and there was a high rate of board failure over the next few months, some sensors were still operational almost a year after deployment. Those boards

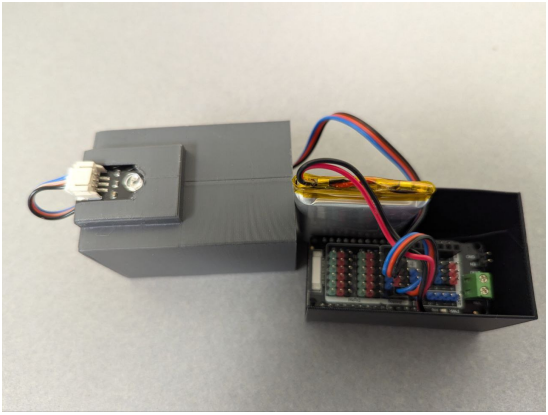


Figure 2: Partially assembled sensor platform

that failed may still be operational, but could not generate enough signal to penetrate the building the base station is located in.

The crop technician responsible for greenhouse management relayed that shortly after the semester ended they received an alert that the temperature was spiking in the greenhouses. Student workers were able to respond quickly to find that the thermostats in the heaters were stuck, causing them to remain on well past their intended working temperature. They were able to react and save the plants, avoiding a second total plant loss. Upon hearing this, students who had implemented the sensors were excited to learn that they had made a difference with their class project.

2.2 Student Evaluations

Student responses to the course were varied but overwhelmingly positive, with an average evaluation score of 3.85/4. The primary negative comment was that the coursework was light towards the end of the semester for the sensor groups while the basestation group was continuing to work dilligently on finalizing their portion of the project. Other comments included:

- With learning the basics of most IOT things, then getting into a practical application of what we learned, it was easy to stay involved in the material and as a result a lot of progress was made in understanding the realm of IOT.
- I think that this class has been the most interesting and the most real world application of a class that I've done so far.

- ...he [the instructor] let us (the students) take over towards the end with the ag project as if its was a real world problem. He of course would help with anything and show us new ideas but he left all of the decision making up to us the students. Really really enjoyed this class. Learned a ton about real world IOT as well, for sure going to recommend to everyone.

- This was a super fun class and a lot was learned by all. I thought the structure of the course was great. With learning the basics of most IOT things, then getting into a practical application of what we learned, it was easy to stay involved in the material and as a result a lot of progress was made in understanding the realm of IOT

Constructive feedback included the need to spend more time covering the operation of basic circuits. Omitted comments didn't address course content constructively or at all but were overwhelmingly positive (for example "It was a fun class, I had so much fun and actually learned something").

2.3 Faculty Observations

From the CS side, the students seemed incredibly motivated. Anecdotally, on the third day of class a student in the front row was heard remarking that he'd found a cheap 3 pack of the microcontrollers on Amazon and had purchased them. Another student took the first two days of lecture content and made the microcontroller control a strip of addressable lights that had been harvested from a PC. Students used office hours to get help and spent extra time working in the greenhouses. Also anecdotally, one student mentioned that this was the one class they still had any motivation to really put effort into and was pushing for a better grade than the minimum needed to graduate.

From the Agriculture/client perspective the implementation of the greenhouse monitoring system did relieve some workload from the crop technician and student workers responsible for maintaining the School of Agricultural Science's greenhouses. The crop technician lives about 45 minutes away from campus, and on cold weekends would drive back up to the school to make sure all the heaters were working properly. The text alert system relieved some of the stress involved with weekend and night-time monitoring. From a School of Agriculture perspective, this project has been lauded as part of the university's commitment to the Career Ready - Day One motto and has been highlighted in prospective student visits. Calls for more student agriculturalists being involved in the project has led to a steadily growing contingent of students being interested in single board computers and sensor networks. As semi- and fully autonomous drones and field equipment become more commonplace in our fields, making sure agriculture students are more cognizant of the work

going into designing and deploying these systems is critical to their future and continued employment.

2.4 University Response

The University response to this project was also overwhelmingly positive. Photographers from the University Marketing team attended the final "presentation" where students remained with their groups in the greenhouses and talked to the invitees. These photographs have been used in university news releases and marketing materials [5]. Additionally in an email responding to the invitation to the final activity the Provost mentioned "This collaboration is perfect and exactly what we need for our students.", and followed through with the awarding of a grant in Fall 2024 to improve on the project and build a base on which we can expand the greenhouse monitoring system. That project is currently in progress using new microcontrollers and different wireless technologies to help with the ineffective short range wireless mesh technologies that were implemented in Fall 2023. This implementation is subject to new and exciting problems for students and faculty to solve, but that is the scope of a future paper.

3 Conclusions

One issue with computer science education is it is not particularly kinetic; unless one is working with drones or other hardware most of the work is one or two people frowning at a computer screen, which does not make for very exciting photo opportunities or the chance to see your code directly interacting with the real world. Every CS student has written Hello World multiple times by the time they are a senior, and yet blinking a LED by pulsing a GPIO pin and getting to learn how the devices they interact with regularly work adds an element of novelty and excitement to the work. Seeing students' faces light up with actual joy when we lit an LED for the first time was particularly gratifying from an educator's standpoint.

It is particularly interesting that one student commented that it was the "most real world application of a class that I've done so far". There was nothing built into the structure of this class that made it remotely like a structured software engineering course; in fact, any structured SDLC was primarily driven by students who had taken the Software Engineering Principles course and were applying skills from other classes. This project was no more or less "real" than other project-based educational experiences. It just felt different, in the sense that you could actually see and feel the products of your work (and smell if you really screwed up). It is easy to lose sight of the fact that while Computer Science is software heavy, the addition of a tactile element that hardware brings

to the educational experience builds an excitement factor; lighting up a single LED on a breadboard is more exciting than a single pixel on a monitor.

Another benefit of this kind of project is it introduces CS students to other disciplines; Computer Science is problem solving, and to do that we need problems to solve. Being able to work in cross- and interdisciplinary teams is an incredibly valuable skill. Getting students to realize that the field does not exist in a vacuum helps put the value of General Education (and in this case specifically science) classes in perspective; a point that was emphasized repeatedly throughout the course was that a lot of the content came from Electrical Engineering (circuit diagrams and construction) and Physics (Ohm's Law and related equations), and specific class demos came from the world of Theater (LED color mixing). The project problem domain was agriculture, but also applied to smart homes and remote sensing (which has applications in Geology and other related Natural Sciences). They could apply the skills and lessons from the class to improve accessibility or provide smart monitoring of power usage with the intent of decreasing energy grid load. The problem domains are nearly endless, assuming one is willing to look for those application areas.

References

[1] Nathan Eloe and Charles Hoot. "Accommodating Shortened Term Lengths in a Capstone Course using Minimally Viable Prototypes". In: *2020 IEEE Frontiers in Education Conference (FIE)*. 2020, pp. 1–8. DOI: 10.1109/FIE44824.2020.9273821.

[2] Arturo Jaime et al. "The Effect of Internships on Computer Science Engineering Capstone Projects". In: *IEEE Transactions on Education* 63.1 (2020), pp. 24–31. DOI: 10.1109/TE.2019.2930024.

[3] Robert Pucher and Martin Lehner. "Project based learning in computer science–a review of more than 500 projects". In: *Procedia-Social and Behavioral Sciences* 29 (2011), pp. 1561–1566.

[4] Espressif Systems. *ESP-NOW Wireless Communication Protocol | Espressif Systems*. 2024. URL: https://www.espressif.com/en/solutions/low-power-solutions/esp-now (visited on 11/25/2024).

[5] Northwest Missouri State University. *Comptuer science students put their skills to work to monitor campus greenhouses*. 2023. URL: https://www.nwmissouri.edu/media/news/2023/12/11internetofthingsproject.htm (visited on 01/14/2025).

[6] Suzanne Young and Mary Alice Bruce. "Classroom community and student engagement in online courses". In: *Journal of Online Learning and Teaching* 7.2 (2011), pp. 219–230.

Creating a Consolidated 3-in-1 Multi-B.S. Program Structure in a Resource-Limited Computer Science & IT Department to Achieve ABET Accreditation for All Three Programs*

Kriti Chauhan, Katherine Lehtola, Dylan Hulon,
Antoni Sayre, James Church, Leong Lee
Department of Computer Science and Information Technology
Austin Peay State University
Clarksville, TN 37044
{chauhank, 1klehtola, 1dhulon, 1asayre, churchj, leel}@apsu.edu

Abstract

Most computing departments offer one or two undergraduate programs, but the CSIT Department at a mid-south state university offers three, all three accredited by the Accreditation Board for Engineering and Technology (ABET): Computer Information Systems, Computer Information Technology, and Computer Science. In 2018, an integrated 3-in-1 B.S. program structure was adopted, which earned ABET accreditation in 2021-2022. With about 650 majors and 100–150 graduates annually, the streamlined curriculum has optimized resources and contributed to the department's success.

1 Introduction

Computing departments, being relatively new in US universities, benefit from accreditation, to assure quality and attract students. This paper outlines the

restructuring of courses offered by a regional four-year, mid-south university's Computer Science and Information Technology (CSIT) Department to improve resource utilization and pursue ABET accreditation.

The CSIT department in the College of Science, Technology, Engineering & Mathematics emerged from the Mathematics department in 2003, as demand grew significantly since offering the first computing course at the university in 1970. Today, it employs 11 tenured/tenure-track faculty, two instructors, 12 adjuncts, three academic advisors, and 30 student workers. The department offers three ABET-accredited Bachelor of Science degrees—Computer Information Systems, Computer Information Technology, and Computer Science—with six concentrations. It is among eight of 411 U.S. universities accredited by the Computing Accreditation Commission (CAC) of ABET to offer at least three accredited programs, reflecting its commitment to quality.

Currently, the CSIT department enrolls approximately 650 computing majors and graduates around 100 to 150 students annually. Offering around 100 to 120 sections of classes during regular school semesters (Fall or Spring), managing the limited resources to meet ABET accreditation requirements and teach the numerous bachelor's and master's classes offered is a significant challenge. This paper presents the consolidated 3-in-1 multi-B.S. program structure designed for efficient resource management, in hopes that this program structure benefits other universities and departments facing similar challenges.

2 Background

The rapidly changing computing industry has led the ABET CAC to frequently revise its accreditation criteria. In 2019, updates were made to both the General Criteria and Program Criteria for computer science, information technology, and information systems programs. Notably, Criterion 3 (Student Outcomes) introduced five core outcomes along with one program-specific outcome for each discipline [6]. Criterion 5 (Curriculum) was also updated, requiring programs to ensure these outcomes are met, including the addition of topics like "substantial coverage of at least one general-purpose programming language" for computer science [1]. Achieving and sustaining ABET accreditation now relies on departments demonstrating adaptability, efficiency, and evidence-based alignment with these criteria.

Institutions have leveraged these updates to improve program effectiveness. For example, the United States Air Force Academy adopted CAC-specified outcomes in 2018 and mapped them to their courses, streamlining curriculum development [4]. Similarly, the University of Central Oklahoma improved ABET evaluation by organizing course materials on a shared server [3]. The New York City College of Technology used detailed assessment models and a focus

on student-centered outcomes to secure re-accreditation [5], while the University of Central Missouri aligned its Cybersecurity program with ABET criteria through evidence-based assessments [8]. East Tennessee State University streamlined its administrative processes to facilitate ABET accreditation for its integrated programs, showcasing the benefits of simplified assessment principles [7]. These examples highlight the importance of flexibility and strategic planning in meeting evolving ABET standards.

3 Seeking ABET Accreditation

In 2018, the CSIT department began the challenging journey of seeking ABET accreditation to validate academic standards. Despite disruptions caused by the COVID-19 pandemic in 2019-2020, the department's faculty persevered, achieving ABET accreditation for all three B.S. programs (CIS, CIT, and CS) during 2020-2022. Accreditation actions officially began for the CIT and CS programs on October 1, 2020, and for the CIS program on October 1, 2021, with the site visit and review completed during the 2021-2022 cycle.

The CAC of ABET ensures computing programs meet quality standards essential for the profession. To keep pace with the rapidly evolving computing industry, the CAC updates its criteria annually. This paper uses the Criteria for Accrediting Computing Programs, 2023-24 [2] as the basis for the department's recent progress report.

The criteria are divided into General Criteria for all baccalaureate-level computing programs and specific Program Criteria for individual programs. ABET evaluates programs to ensure they meet both the General and Program Criteria. There are eight General Criteria applicable to the CIS, CIT, and CS programs. Additionally, each of these programs—CIS, CIT, and CS—must also satisfy its specific Program Criteria.

ABET criteria are divided into General Criteria for all baccalaureate-level programs and Program Criteria for specific disciplines. The eight General Criteria cover: 1. Students, 2. Program Educational Objectives, 3. Student Outcomes, 4. Continuous Improvement, 5. Curriculum, 6. Faculty, 7. Facilities, and 8. Institutional Support.

Program-specific criteria address discipline-specific requirements for CIS, CIT, and CS programs. This paper focuses on Criterion 3 (Student Outcomes) and Criterion 5 (Curriculum) within the department's integrated 3-in-1 B.S. program structure, emphasizing efficient resource management. Additionally, Criterion 4 (Continuous Improvement), which evaluates the effectiveness of Student Outcomes, is discussed in this context.

- General Criteria
  - Criterion 1. Students
  - Criterion 2. Program Educational Objectives
  - Criterion 3. Student Outcomes
  - Criterion 4. Continuous Improvement
  - Criterion 5. Curriculum
  - Criterion 6. Faculty
  - Criterion 7. Facilities
  - Criterion 8. Institutional Support
  - CIS: Information Systems & Similarly Named Computing Programs
  - CIT: Information Technology & Similarly Named Computing Programs
  - CS: Computer Science & Similarly Named Computing Programs
- Program Criteria
  - Criterion 1. Students
  - Criterion 2. Program Educational Objectives
  - Criterion 3. Student Outcomes
  - Criterion 4. Continuous Improvement
  - Criterion 5. Curriculum
  - Criterion 6. Faculty
  - Criterion 7. Facilities
  - Criterion 8. Institutional Support
  - CIS: Information Systems & Similarly Named Computing Programs
  - CIT: Information Technology & Similarly Named Computing Programs
  - CS: Computer Science & Similarly Named Computing Programs

This paper focuses on Criterion 3 (Student Outcomes) and Criterion 5 (Curriculum), exploring how these criteria are integrated into our 3-in-1 multi-B.S. program structure for efficient resource management. Since Criterion 4 (Continuous Improvement) involves the assessment of Criterion 3 (Student Outcomes), it is also relevant to our discussion.

Criteria 3 (student outcomes) and 5 (curriculum) are typically the most challenging aspects of the ABET CAC accreditation process. Improper mapping of Criterion 3 to the program can complicate Criterion 4 (continuous improvement). The next section will present the Criterion 5 Curriculum 3-in-1 Program Map alongside the introduction of the 3-in-1 program structure.

Table 1: Criterion 3. Student Outcomes (SLOs), 3-in-1 Program Map (2023-24 ABET Criteria)

B.S. in Computer Information Systems	B.S. in Computer Information Technology	B.S. in Computer Science
Graduates of the program will have an ability to: 1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. (SLO1) 2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (SLO2) 3. Communicate effectively in a variety of professional contexts. (SLO3) 4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles. (SLO4) 5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (SLO5)		
6. Support the delivery, use, and management of information systems within an information systems environment. [IS] (SLO6-CIS)	6. Use systemic approaches to select, develop, apply, integrate, and administer secure computing technologies to accomplish user goals. [IT] (SLO6-CIT)	6. Apply computer science theory and software development fundamentals to produce computing-based solutions. [CS] (SLO6-CS)

4 Resources, 3-in-1 Program Structure, SLOs, Computing Core, Major Core

Due to limited resources, the CSIT department's 3-in-1 program structure aims to minimize course preparations. Adjunct instructors typically don't develop course materials due to compensation and time constraints, while full-time instructors, though responsible for course development, can only manage 30-40 preparations per semester. To streamline this, the department developed Computing Core and Major Core, and the degree plan comprises of four sections:

- **General Education Core** Courses: cover communications, history, humanities/fine arts, mathematics, natural sciences, and social/behavioral sciences.
- **Computing Core** Courses: Required across all degree programs, allowing one instructor to develop materials for multiple sections, ensuring consistency and efficiency in foundational computing concepts. These courses are offered every semester.
- **Major Core** Courses: Required for all concentrations within a specific program, offering essential skills for any specialization. These are offered annually.
- **Concentration** Courses: Specialized courses for specific concentrations, offered once a year.

The 3-in-1 program structure is mapped in Table 2, which aligns with the department's assessment plan. To meet Criterion 4: Continuous Improvement, assessment is conducted using CSCI 3000-4999 level courses, with each major's core courses used to assess relevant Student Learning Outcomes (SLOs). This structure enables the CSIT department to maintain efficient course development and meet ABET requirements with a manageable workload of 30-40 preparations per semester by the limited full-time instructors. Criterion 4: Continuous Improvement is a substantial topic that warrants further discussion in another paper.

5 Curriculum

The 3-in-1 program structure, as shown in Table 2, aligns with ABET CAC Criterion 5: Curriculum and the ABET curriculum structure outlined in Table 3, based on the 2023-2024 ABET Criteria. Note that concentration courses are not used to satisfy ABET Criteria. Information Systems offers General IS concentration and Cybersecurity concentration. Information Technology offers Database concentration and Web and Networking concentration. Computer Science offers General CS concentration and Software Engineering concentration.

ABET criteria are updated annually, and the university maps its academic bulletin to these criteria each year to ensure compliance. The CSIT faculty integrate these criteria when making curriculum changes, and maintain compliance despite limited resources.

With Computing Core courses offered each semester and Major Core and Concentration courses annually, the department can create a predictable course

schedule that meets ABET requirements. This tiered structure allows the department to operate within its resource constraints, including faculty, budget, and classroom space. This approach may benefit other small universities offering multiple degrees with common courses.

Table 2: The 3-in-1 Program Structure (AY 2023-24 Bulletin for 2023-24 ABET Criteria)

<table>
<tr><th>B.S. in Computer Information Systems</th><th>B.S. in Computer Information Technology</th><th>B.S. in Computer Science</th></tr>
<tr><td colspan="3">General Education Core</td></tr>
<tr><td colspan="3">Computing Core
Complete one option from:
Regular sequence
CSCI 1010 Introduction to Programming I 3
CSCI 1011 Introduction to Programming I Lab 1
CSCI 2010 Introduction to Programming II 3
CSCI 2011 Introduction to Programming II Lab 1
CSCI 2000 Programming for STEM 4
CSCI 2600 Computer Ethics 3
CSCI 2700 Data Communications & Networking 3
CSCI 4200 Principles of Information Security 3
CSCI 4400 Principles of Database Management 3
CSCI 4800 Senior Seminar 1
Total Hours 17-21</td></tr>
<tr><td>CIS Major Core
CSCI 4018 Cloud Computing 3
CSCI 4603 Requirements & Project Mgt 3
CSCI 4750 Systems Analysis & Design 3
ENGL 1100 Technical & Report Writing 3
MATH 1530 Elements of Statistics 3
Complete one course from:
CSCI 2500 Discrete Structures 3
MATH 1810 Elements of Calculus 3
MATH 1910 Calculus I 4
Choose 1 of 2 focuses (Courses Not Shown Here)
Information Systems Environment
Choose 1 of 2 focuses (Courses Not Shown Here)
Total Hours 33-34</td><td>CIT Major Core
CSCI 1005 – Intro. to Information Technology 3
CSCI 1300 Introduction to Web Development 3
CSCI 2500 Discrete Structures 3
CSCI 3300 Client-Side Web Development 3
CSCI 3350 User Experience Design 3
CSCI 4603 Requirements & Project Mgt.
CSCI 4650 Windows Server-Side Web Dev. 3
CSCI 4750 Systems Analysis & Design 3
CSCI 4760 Linux System Administration 3
MATH 1530 Elements of Statistics 3
Total Hours 30</td><td>CS Major Core
CSCI 3005 Graphical User Interfaces 3
CSCI 3250 Data Structure & Algorithms 3
CSCI 3400 Computer Organization I 3
CSCI 4100 Operating Systems & Arch. 3
CSCI 4230 Programming Languages 3
CSCI 4270 Algorithm Design & Analysis 3
CSCI 4805 Computer Science Capstone 3
MATH 1910 Calculus I 4
MATH 1920 Calculus II 4
MATH 3000 Discrete Mathematics 3
MATH 3450 Linear Algebra 3
1 Math Elective; Select 2 Science Lecture & Lab
Total Hours 46</td></tr>
<tr><td>General Info. Systems Concentration OR Info. Assurance & Security Concentration (Courses Not Shown Here) Electives. Total Program Hours 120</td><td>General Info. Systems Concentration OR Info. Assurance & Security Concentration (Courses Not Shown Here) Electives. Total Program Hours 120</td><td>General Comp. Sci. Concentration OR Software Engineering Concentration (Courses Not Shown Here) Electives. Total Program Hours 120</td></tr>
</table>

Table 3: Criterion 5. Curriculum, AY 2023-24 Bulletin (To Satisfy 2023-24 ABET Criteria)

<table>
<tr><th>B.S. in Computer Information Systems</th><th>B.S. in Computer Information Technology</th><th>B.S. in Computer Science</th></tr>
<tr><td colspan="3">General Curriculum Criteria (Almost the same for all computing-related B.S. Programs) – Covered by Computing Core courses
The program includes mathematics appropriate to the discipline and at least 30 semester credit hours of up-to-date coverage of fundamental and advanced computing topics that provide both breadth and depth. The computing topics include:
1. Techniques, skills, and tools necessary for computing practice (CSCI 1010 / CSCI 1011 / CSCI 2000 / CSCI 2010 / CSCI 2011).
2. Principles and practices for secure computing (CSCI 4200).
3. Local and global impacts of computing solutions on individuals, organizations, and society (CSCI 2600 / CSCI 4800).</td></tr>
<tr><td colspan="3">Program Specific Curriculum Criteria – Covered by Computing OR Major Core Courses</td></tr>
<tr><td>Computer Information Systems meets the following curriculum program criteria.
1. Information systems (at least 30 credit hours): Fundamentals and applied practice in application development (CSCI 1010 / CSCI 1011 / CSCI 2000 / CSCI 2010 / CSCI 2011), data and information management (CSCI 4400), information technology infrastructure (CSCI 3400 / CSCI 2700), systems analysis, design and acquisition (CSCI 4750), project management (CSCI 4603), and the role of information systems in organizations (CSCI 4750).
2. Information systems environment requirement (at least 15 credit hours): Cohesive set of topics that provide an understanding of an information systems environment - choose one of two 15-hour focuses – Earth & Environmental Science or Business).
3. Quantitative analysis or methods (include statistics): MATH 1530 (statistics)/ CSCI 2500 / MATH 1810 / MATH 1910.</td><td>Computer Information Technology meets the following curriculum program criteria.
1. Information Technology (at least 45 credit hours):
(a) Fundamentals and applied practice in: information management (CSCI 4400), integrated systems (CSCI 4750), platform technologies (CSCI 4760), system paradigms (CSCI 4603 / 4750), user experience design (CSCI 3350), networking (CSCI 2700), software development and management (CSCI 4603 / CSCI 4750), web and mobile systems (CSCI 1300 / CSCI 3300 / CSCI 3350 / CSCI 4650).
(b) Advanced and supplemental IT topics that build on fundamentals and applied practice to provide depth. (CSCI 1300 / CSCI 3300 / CSCI 3350 / CSCI 4650 / CSCI 4750)
(c) Experiential learning appropriate to the program (CSCI 3350 / CSCI 4650 / CSCI 4750).
(d) Principles and practices of IT project management (CSCI 4603).
2. Mathematics (at least 6 credit hours, include relevant discrete mathematics): Math 1530 / CSCI 2500 (includes relevant discrete mathematics).</td><td>Computer Science meets the following curriculum program criteria.
1. Computer science (at least 40 credit hours):
(a) Substantial coverage of algorithms and complexity (CSCI 3250 / CSCI 4270), computer science theory (CSCI 3250 / CSCI 4230), concepts of programming languages (CSCI 4230), and software development (CSCI 1010 / CSCI 2010 / CSCI 2000 / CSCI 3005).
(b) Substantial coverage of at least one general-purpose programming language (CSCI 1010/2010 / CSCI 2000 / CSCI 3005).
(c) Exposure to computer architecture and organization (CSCI 3400), information management (CSCI 3250 / CSCI 4400), networking and communication (CSCI 2700 / CSCI 4100), operating systems (CSCI 4100), and parallel and distributed computing (CSCI 4100).
(d) Study of computing-based systems at varying levels of abstraction. (CSCI 1010 / CSCI 2010 / CSCI 2000 / CSCI 3005 / CSCI 3400 / CSCI 3250).
(e) A major project that requires integration and application of knowledge and skills acquired in earlier course work. (CSCI 4805).
2. Mathematics and Statistics (at least 15 credit hours, include discrete mathematics, probability, and statistics and must have mathematical rigor at least equivalent to introductory calculus): MATH 1910 / MATH 1920 / MATH 3000 / MATH 3450 / MATH 4670 / STAT 3250 / STAT 4240.
3. Science: Coursework that develops and applies the scientific method in a non-computing area - Natural Science Core for Computer Science</td></tr>
</table>

6 Conclusions

The 3-in-1 program structure, though unconventional, effectively addresses resource constraints while meeting ABET's student learning outcomes, curriculum, and continuous improvement requirements. It provides a clear framework for long-term course scheduling, optimizing limited resources and enhancing

manpower, room allocation, and budget efficiency. This structure also offers flexibility for students, allowing them to switch between B.S. programs and reuse Computing Core courses across CIS, CIT, or CS degrees. Students can change concentrations and apply Major Core courses across different areas, aiding retention.

Ultimately, the 3-in-1 structure helped the CSIT department achieve ABET accreditation for all three programs. Ongoing studies are exploring its impact on continuous improvement, resource allocation, and justification. With a central core, major core, and upper-division classes, this structure offers clarity for advisors, flexibility for students, and better resource management, making it a valuable model for other departments facing similar challenges.

References

[1] ABET. Nov. 2018. URL: https://www.abet.org/wp-content/uploads/2018/11/C001-19-20-CAC-Criteria-11-24-18.pdf.

[2] ABET. *Criteria for accrediting computing programs, 2023 - 2024*. June 2023. URL: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2023-2024/.

[3] Jicheng Fu et al. "Obtaining and maintaining ABET accreditation: An experience-based review of the ABET criteria for computer science programs". In: *Journal of Computing Sciences in Colleges* 29.4 (2014), pp. 13–19.

[4] Steve Hadfield et al. "Streamlining computer science curriculum development and assessment using the new ABET student outcomes". In: *Proceedings of the Western Canadian Conference on Computing Education*. 2019, pp. 1–6. DOI: 10.1145/3314994.3325079. URL: https://doi.org/10.1145/3314994.3325079.

[5] Lili Ma and Benito Mendoza. "Assessing Student Outcomes Related to Design for ETAC-ABET Accreditation". In: *Journal of Computing Sciences in Colleges* 38.3 (2022), pp. 150–164.

[6] Michael J Oudshoorn et al. "Understanding the new abet computer science criteria". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 2018, pp. 429–434. DOI: 10.1145/3159450.3159534. URL: https://doi.org/10.1145/3159450.3159534.

[7] Tony Pittarese et al. "Developing an integrated multi-program computing department". In: *Journal of Computing Sciences in Colleges* 30.5 (2015), pp. 20–26.

[8] Xiaodong Yue, Belinda Copus, and Hyungbae Park. "How to secure ABET accreditation for a cybersecurity program: a case study". In: *Journal of Computing Sciences in Colleges* 37.6 (2022), pp. 15–24.

A Comparison of Different Lab Environments for Digital Forensics Course*

Zhengrui Qin
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
zqin@nwmissouri.edu

Abstract

Nowadays, with the development of cloud computing, various lab environments have emerged, which, however, have seldom been compared with each other. In this paper, we have conducted a survey on different lab environments for Digital Forensics course, i.e. NDG (Network Development Group), Cengage MindTap, and running labs on personal computers. In the survey, students have expressed their preferences and evaluated each lab environment. We believe this study can help instructors select the right lab environment for Digital Forensics course or similar courses.

1 Introduction

In recent years, cloud computing [6] has emerged as a popular and promising computing paradigm, which not only benefits industries and individuals but also changes the teaching approach in colleges. As the result, assignments/labs are shifting from paper-based and/or computer-based approaches to cloud-based approaches. This trend greatly lightens the burden of the teachers; furthermore, it facilitates the online students significantly.

However, when developing a new course or upgrading an existing course, it may be challenging to select the proper cloud product from various cloud vendors. In the field of cybersecurity, several cloud services are available, such as NDG (Network Development Group) [5], Cengage MindTap [1], and TestOut [7]. Each of these cloud services offers assignments/labs for a bunch of courses. Some of these cloud services have been studied separately by scholars [2, 3]. However, to the best of our knowledge, no research or study has compared these cloud services. Furthermore, to be more accurate, it is better to conduct the comparison to a specific course instead to the cloud service as a whole, which will be even more challenging. First, a course usually only uses one cloud service. Second, even a course may use two or more cloud services, it may not use them in the same semester but in different semesters; as the result, one cannot get the feedback for different cloud services from the same group of students.

In this paper, we aim to provide the comparison of the cloud services for Digital Forensics course, a mandatory course in the cybersecurity program. We will compare two cloud services that provide digital forensics labs, NDG and MindTap, along with lab environment in one's own computer; that is, three lab environments will be compared, two are cloud-based and one is PC-based. In our Digital Forensics course in the fall of 2023, we assigned labs from both NDG and MindTap to students, and also offered multiple digital forensics activities on their own computers (provided by the university). At the end of the semester, we conducted a survey among the students who took the course. In this paper, we have presented our survey and shown our observations. We hope that our work can help teachers select the proper cloud service in a similar situation.

The rest of paper is organized as follows. Section 2 briefly describes NDG and MindTap labs. Section 3 presents the questionnaire of our survey. Section 4 shows the survey results and our observations. Section 5 concludes the paper.

2 Preliminaries

In this section, we will describe the three lab environments in our Digital Forensics course. The first two, NDG and MindTap, are hosted in the cloud, and the students only need a web browser to conduct the labs. However, for the 3rd one, the students have to install necessary software to create a lab environment on their own computers.

2.1 NDG (Network Development Group)

NDG Online[5] is a hands-on IT cloud training solution with environments right in users' browsers. It offers a growing selection of online courses and

labs featuring coursebooks, videos, lab exercises, and assessments. It covers seven broad topics, including Cloud, Cybersecurity, Linux, Networking, Open Source, Storage, and Virtualization.

The product used in our Digital Forensics course is NDG Forensics V2 within Cybersecurity. Each student needs an account, which costs $50. We used a cybersecurity grant to purchased one account for each student.

NDG Forensics V2 consists of 21 labs, listed as below.

Lab 01: Creating a Forensic Image
Lab 02: Live Acquisition
Lab 03: Live Forensics
Lab 04: Registry Forensics
Lab 05: File Systems
Lab 06: Keyword Search and Analysis
Lab 07: Data Carving
Lab 08: Metadata and Link File Analysis
Lab 09: Recycle Bin Forensics
Lab 10: Steganography and Alternative Data Streams
Lab 11: Picture File Analysis
Lab 12: Email Analysis
Lab 13: Internet Browser Forensics
Lab 14: Timeline Analysis
Lab 15: IoT Forensics
Lab 16: Mobile Forensic Analysis
Lab 17: Log Capturing and Interpretation
Lab 18: Pagefile Analysis
Lab 19: Password Cracking
Lab 20: File Hashing and Hash Analysis
Lab 21: Chain of Custody

2.2 Cengage MindTap

MindTap [1], powered by Cengage, is an online learning platform, through which the students can access their full eBook, do their homework, conduct live virtual machine labs, complete assessments, and peruse study tools.

The textbook used in our course, *Guide to Computer Forensics and Investigations* [4], contains 16 modules. Each module has a live virtual machine lab, except Module 2 and Module 8. The labs are listed as below.

Lab 1: Understanding the Digital Forensics Profession and Investigation
Lab 3: Data Acquisition
Lab 4: Processing Crime and Incident Scenes
Lab 5: Working with Windows and CLI Systems
Lab 6: Current Digital Forensics Tools
Lab 7: Linux and Macintosh File Systems
Lab 9: Digital Forensics Analysis and Validation
Lab 10: Virtual Machine Forensics, Live Acquisitions, and Network Forensics
Lab 11: E-mail and Social Media Investigations

Lab 12: Mobile Device Forensics
Lab 13: Cloud Forensics
Lab 14: Report Writing for High-Tech Investigations
Lab 15 Expert Testimony in Digital Investigations
Lab 16: Ethics for the Expert Witness

2.3 Labs on Personal Computers

In this option, the students have to install digital forensics tools on their own computers. Tools include VirtualBox, Autopsy, FTK, Sleuth Kit, etc. Below is an example lab.

Defeat Simple Steganography:

Here is a public website: https://github.com/JerryQinNW/386-site. The site illustrates some simple techniques for hiding information in plain view. Explore the site, and try to uncover a hidden message.

Verify: Include a screen shot clearly showing on your machine the secret message(s) uncovered.

Describe and show what you did to uncover each message. The answers are the same for everyone, so your screenshots and explanations are critical for earning credit.

2.4 Lab Selection

As we only had 14 weeks excluding the first week and the final exam week in the fall semester, we could not assign all labs to the students. For NDG labs, we selected Lab 01, Lab 02, Lab 03, Lab 05, Lab 10, Lab 12, and Lab 16; and for MindTap labs, we selected Lab 1, Lab 2, Lab 4, Lab 5, Lab 7, Lab 9, Lab 10. Most of the two chosen sets of labs are of different topics, while there are still some overlapped topics, such as NDG Lab 05 and MindTap Lab 7 with the same topic – File Systems.

3 The Survey

To make a fair comparison, we have designed a survey with 16 questions. The questionnaire is listed below. Questions #1-13 ask students' opinions on MindTap and NDG, respectively. Question #14 directly compare MindTap and NDG. Question #15 compares running labs on one's own computer (the 3rd option) against MindTap and NDG. Finally, Question #16 is asking the students how they like the textbook, *Guide to Computer Forensics and Investigations*.

1. Are NDG labs interesting?
2. Are MindTap (Cengage) labs interesting?
3. How satisfied are you with NDG labs? (latency, bandwidth, instructions, etc.)
4. How satisfied are you with MindTap labs? (latency, bandwidth, instructions, etc.)
5. How effective is NDG labs? (think about what you have learned from the labs)

6. How effective is MindTap labs? (think about what you have learned from the labs)
7. Is it worthy to pay $50 for all the labs in NDG?
8. On average, how many hours do you spend roughly on each NDG lab?
9. On average, how many hours do you spend roughly on each MindTap lab?
10. Are the NDG labs difficult?
11. Are the MindTap labs difficult?
12. On average, how many attempts do you need to finish one MindTap lab?
13. On average, how many attempts do you need to finish one NDG lab?
14. Comparing labs in NDG labs and MindTap, on average, which is better?
15. Do you prefer to run the labs on your own computer rather than NDG or MindTap? (Install all software and create the environment.)
16. How do you like the textbook?

For each question except Question #12 and #13, we measure the students' opinions with the 5-point Likert scale, which consists of the following points: (1 point) strongly disagree; (2 points) disagree; (3 points) neither agree nor disagree; (4 points) agree; and (5 points) strongly agree. For Question #12 and #13, we ask the students to input their numbers.

It is necessary to point out that we did not survey the 3rd option directly. The reason is that at the beginning of the semester, we asked the students whether they would like to do the labs on their own computers or on the cloud; all students chose the cloud option. As the result, we did not assign any labs in the 3rd option. However, students did engage many digital forensics activities during class discussions and demonstrations.

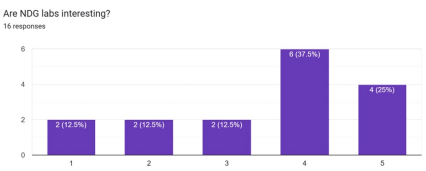
4 Survey Results and Analysis

At the last week of the fall semester of 2023, we conducted an anonymous survey among the students who took the Digital Forensics course. 16 out of 19 students took the survey. The results are shown in Figure 1 - Figure 9 and Table 1. We will explain them one by one.

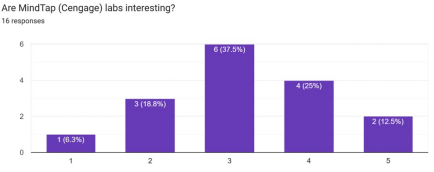
4.1 Survey Observations

Figure 1 shows the interesting level of NDG labs and MindTap labs. Comparing Figure 1a and Figure 1b, we can see that NDG labs are more interesting than MindTap labs from the view of students. In total, 62.5% students agree or strongly agree that NDG labs are interesting, while the percentage for MindTap labs is only 37.5%.

Figure 2 shows how the students are satisfied with NDG labs and MindTap labs considering the latency, bandwidth, instructions, etc., for example, whether the instructions of the labs are easy to follow. Comparing Figure 2a and Figure 2b, we can see that NDG labs are more satisfying than MindTap

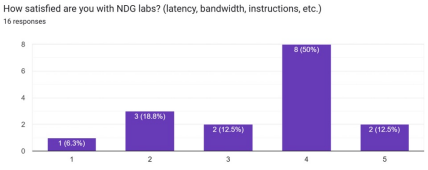


(a) NDG interesting level.

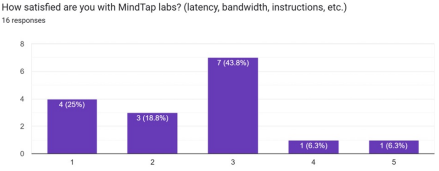


(b) MindTap interesting level.

Figure 1: Interesting level of NDG and MindTap.



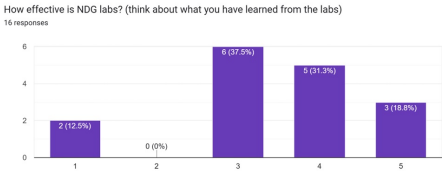
(a) NDG satisfying result.



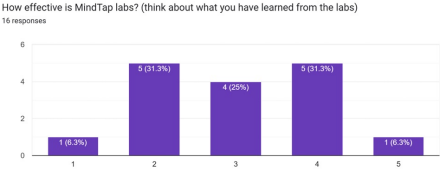
(b) MindTap satisfying result.

Figure 2: How are NDG and MindTap labs satisfying?

labs. In total, 62.5% students agree or strongly agree that NDG labs are satisfying, while the percentage for MindTap labs is only 12.5%. We observed this trend during the semester as well, as some students complained that instructions of MindTap labs sometimes were hard to follow.



(a) Effectiveness of NDG labs.

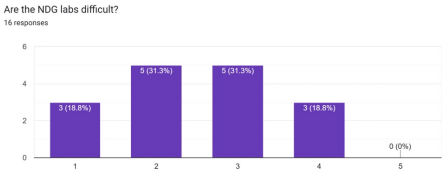


(b) Effectiveness of MindTap labs.

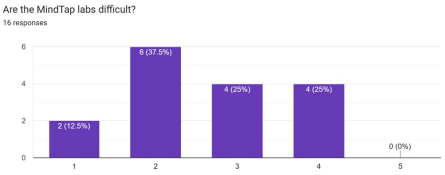
Figure 3: Effectiveness of NDG and MindTap labs.

Figure 3 shows the effectiveness of NDG labs and MindTap labs, i.e., what students can learn from the labs or how the knowledge in class is strengthened. We can see that NDG labs are more effective than MindTap labs. In total,

37.5% of students disagree or strongly disagree that MindTap labs are effective while the percentage for NDG labs is only 12.5%. However, even for NDG labs, only half of the students agree or strongly agree the labs are effective.



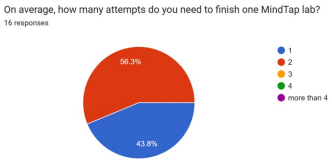
(a) Difficulty level of NDG labs.



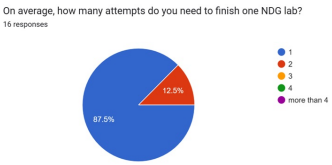
(b) Difficulty level of MindTap labs.

Figure 4: Difficulty assessment for NDG and MindTap labs.

Figure 4 shows whether the labs are difficult. We can see that most of students think that neither NDG nor MindTap labs are difficult. This observation is also reflected during the help session, because there were only a few cases that the students asked for helps on the labs. Even with these cases, with only some hints or guidance from the instructor or the teaching assistant, the students could smoothly go through the labs.



(a) # of attempts needed in NDG labs.



(b) # of attempts needed in MindTap labs.

Figure 5: Number of attempts on average needed for one lab.

Figure 5 shows that on average how many attempts are need for each lab. We designed this question because we noticed that sometimes the labs were frozen and required to be restarted. In this regard, NDG platform performs much worse than MindTap platform, since the students need two attempts for more than half of labs while they only need one attempt for most of MindTap labs.

MindTap labs are included in the ebook, which can be counted as free. NDG labs are $50 for each student account, and we are interested that how the students value the cost. Figure 6 shows that, only a quarter of the students

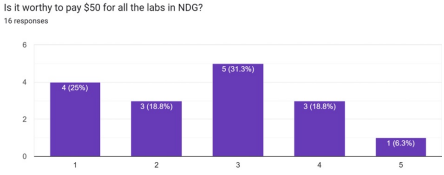


Figure 6: NDG cost effectiveness.

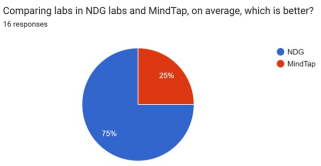


Figure 7: Overall comparison.

agree or strong agree it is worthy to pay $50 for the labs, even though they did not pay a penny (paid by a cybersecurity grant).

Figure 7 shows the overall comparison between NDG labs and MindTap labs. Based on previous comparison in Figure 1 - Figure 5, it is not surprising to see that the students prefer NDG labs to MindTap labs.

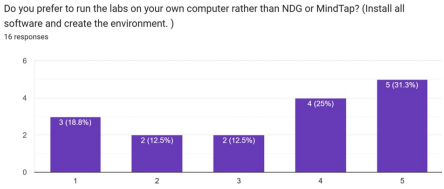


Figure 8: Running labs on PC?

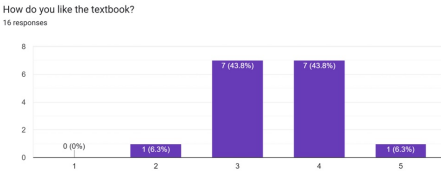


Figure 9: Textbook assessment.

Figure 8 shows that, compared to NDG and MindTap labs, the students are actually willing to run the labs on their own computers (provided by the university). This observation is reasonable, because even NDG labs, the better of the two, are far from perfect. Figure 9 shows how the students value the textbook. The majority of the students either like the textbook or remain neutral, even though they do not like MindTap labs. This suggests that, as instructors, we would better not to use MindTap labs while using the corresponding digital forensics textbook.

We also estimated how much time students spent on each lab on average. Table 1 shows the time spent for each of the 16 students and the average, for NDG labs and MindTap labs, respectively. Since the survey is done at the end of the term, the time is just an estimate based on students' impression. On average, students spent about 1 to 2 hours on each lab, while MindTap labs take about half hour longer.

Table 1: Hours spent on NDG and MindTap labs.

Student	1	2	3	4	5	6	7	8
NDG	0.5	1.5	0.5	1.5	1.0	1.0	1.0	0.7
MindTap	1.0	1.0	1.0	1.5	2.0	1.0	1.0	0.7
Student	9	10	11	12	13	14	15	16
NDG	1.5	0.5	1.0	1.0	1.5	5.0	1.5	1.3
MindTap	1.5	0.5	1.5	5.0	2.0	5.0	1.5	2.0
Mean ± Std Dev	NDG		1.3 ± 1.1		MindTap		1.8 ± 1.3	

4.2 Instructor's View

Based on the students' opinions above and the instructor's investigation, NDG labs are better than MindTap labs in terms of lab design, effectiveness, and knowledge coverage. However, NDG labs presents challenges in grading (Forensics V2 did not provide any grading mechanism in 2023). To grade the NDG labs, the instructor had to ask the students to manually take some screenshots, check the time students spent, and examine the data volume transferred. As the result, the grading could not be very accurate. However, MindTap labs are graded automatically by the platform itself.

5 Conclusion

In this paper, we have investigated different lab environments for Digital Forensics course. We conducted a survey among the students who took the course in the fall of 2023. It turns out that students preferred NDG labs to the MindTap labs. However, both NDG and MindTap labs are far from perfect. Therefore, running all the labs on students' own computers or a cybersecurity lab would be a promising choice. We believe that our findings can provide some guidance for instructors in a similar situation.

References

[1] *Cengage MindTap*. URL: `https://www.cengage.com/mindtap/`.

[2] Jahan Hassan, Anamika Devi, and Biplob Ray. "Virtual laboratories in tertiary education: Case study analysis by learning theories". In: *Education Sciences* 12.8 (2022), p. 554.

[3] Usha Jagannathan and Risa Blair. *Interdisciplinary initiative for infusion of virtual labs in IT and engineering degree programs*. 2015.

[4] Bill Nelson, Amelia Phillips, and Christopher Steuart. *Guide to computer forensics and investigations*. Cengage Learning, 2022.

[5] *Network Development Group*. URL: `https://www.netdevgroup.com/online/`.

[6] Ling Qian et al. "Cloud computing: An overview". In: *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*. Springer. 2009, pp. 626–631.

[7] *TestOut*. URL: `https://w3.testout.com`.

Impact of COVID-19 on Live-Coding in First-Year Computer Science Education: A Literature Review*

Sourabh Kulkarni, Abbas Attarwala, Jaime Raigoza
Computer Science Department
California State University, Chico
Chico, CA 95973
{sdkulkarni, aattarwala, jraigoza}@csuchico.edu

Abstract

The COVID pandemic has significantly influenced educational methodologies, leading to a shift towards more interactive and technology-integrated teaching approaches. Live-coding, which involves real-time coding demonstrations, has gained recognition as a valuable tool in computer science education. This study investigates the impact of the pandemic on the popularity and application of live-coding. By conducting a comprehensive literature review of 22 research papers, this study categorizes the papers based on their publication date relative to the pandemic: pre-COVID (2017 to 2019), during COVID (2020 to 2022), and post-COVID (2023 to 2024). The papers were selected using a specific search query for live-coding in introductory computer science education on Google Scholar. A systematic literature review was performed to determine their sentiment towards live-coding, categorizing the sentiments as positive, neutral, or negative. The sentiment data were then statistically analyzed using Fisher's Exact Test to assess significant differences across the three periods. Results from this manuscript indicate shifts in positive sentiment towards live-coding during and after the pandemic.

1 Introduction

The origin of live-coding as an educational tool has gained significant attention within the academic community, particularly in the fields of computer science and programming education. Live-coding involves instructors writing and explaining code in real time during lectures. [1] has mentioned the benefits of live-coding over static examples and power point slides where live-coding offers students a more richer and an interactive learning experience over other forms of learning.

The global spread of the COVID virus caused disruptions, in education systems around the world leading to the adoption of transitions to online and blended learning methods. In this manuscript, we investigate whether COVID increased positive sentiment towards live-coding in first-year programming courses in computer science. Our research question is: How has the popularity and application of live-coding in first-year computer science education changed before, during, and after the COVID pandemic?

We first perform a literature review, detailed in Section 2, classifying papers as positive or not positive (neutral or negative) towards live-coding across three time periods: pre-COVID (2017 to 2019), during COVID (2020 to 2022), and post-COVID (2022 to 2024). In Section 4, we state our hypothesis and perform the Fisher Exact Test (FET). Section 5 presents our discussion of the results, and Section 6 provides our conclusions.

2 Literature Review

For pre-COVID, the following papers were looked at: 1) The paper [16] evaluates the effectiveness of live-coding in introductory programming courses. It highlights the benefits, such as enhanced understanding of programming concepts and debugging through real-time code development. Positive feedback includes increased confidence and learning program design. However, some students expressed concerns about the pace and suggested more interactive involvement. Thus, the overall sentiment of the paper towards live-coding is **Neutral**. 2) The paper [3] introduces "Improv", an IDE extension that enhances live-coding in education by integrating PowerPoint slides with live code, reducing cognitive load and enhancing instructional clarity. Using Mayer's multimedia learning principles, Improv synchronizes code snippets with presentation slides, improving tutorial delivery. The authors highlight its benefits, stating that Improv combines the organization of slide presentations with the flexibility of live-coding, contributing positively to learning at scale. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 3) The paper [5] reviews the current state of computing education research, analyzing aca-

demic and policy literature to address challenges and opportunities in teaching computing, especially with new UK curricula. It explores various pedagogical techniques, including live-coding. The authors note that live-coding can be as effective as, if not better than, static code examples for teaching introductory programming, indicating their positive sentiment towards live-coding. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 4) The paper [2] explores using Jupyter notebooks as versatile educational tools across various disciplines. It discusses how Jupyter notebooks enhance student engagement, participation, understanding, and performance. Highlighting live-coding, the paper states that live-coding provides active learning by allowing students to complete code in notebooks before demonstrations. The paper asserts that notebooks increase student engagement and preparation for careers, indicating a **Positive** sentiment towards live-coding. 5) The paper [15] investigates factors influencing student performance in an introductory programming course at the Royal University of Bhutan. Using a mixed methods approach, data were collected through tests, surveys, and interviews. The findings suggest that live-coding, along with other methods like pair programming and independent coding, aids in learning. The paper highlights the benefits of live-coding, noting its effectiveness in engaging students. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 6) The paper [4] discusses developing a virtual laboratory to teach X-ray imaging principles to engineering students. It uses a creative approach integrating scientific simulations with modern computing techniques. The study combines qualitative and quantitative data to evaluate the virtual lab's impact on learning. The paper highlights the benefit of live-coding, noting that synchronous editing of shared documents enhances ensemble live-coding performance. Thus, the overall sentiment of the paper towards live-coding is **Positive**.

For during COVID, the following papers were looked at: 1) The paper [13] reviews academic and policy literature on using worked examples in computer science education, focusing on code-tracing and code-generation examples. It highlights that live-coding is appreciated by students more than standard lectures but also notes studies showing no significant differences between live-coding and static examples. Given these mixed findings, the overall sentiment of the paper towards live-coding is **Neutral**. 2) The paper [21] investigates challenges faced by near-novice programmers when learning a second programming language, focusing on a transfer pedagogy using live-coding. The study involves 62 second-year students transitioning from Python to Java, employing both quantitative and qualitative methods. Student feedback highlighted increased engagement, while lecturer feedback noted mixed results depending on student experience. Despite some challenges, the overall sentiment of the paper towards live-coding is **Positive**. 3) The paper [14] examines the

impact of active learning classrooms (ALC) on student performance in a second programming course at the University of Toronto Mississauga. Using a quasi-experimental design with 529 participants, the study compares active learning methods, including live-coding, in different classroom environments. The results do not indicate significant differences in student performance between room types, although ALCs stimulated interactivity. Thus, the general sentiment of the paper towards live-coding is **Neutral**. 4) The paper [12] investigates novice programmers' understanding of code before and after an introductory programming course. Using a mixed-methods approach, it assesses students' understanding through a "code as a story" method and evaluates code quality using SonarQube. The study suggests adding methods like live-coding to improve teaching. Despite not specifying live-coding's benefits, the overall sentiment of the paper towards live-coding is **Positive**. 5) The paper [20] investigates challenge-based and competency-based assessment strategies in a first-year programming course at McMaster University. Involving 207 students, the study evaluates performance through term tests and a final exam. The paper highlights live-coding as a key pedagogical approach that resulted in significant interaction and active learning. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 6) The paper [10] investigates the rapid shift from face-to-face to online learning during the COVID pandemic using a mixed-methods approach. The study includes surveys, focus groups, and continuous monitoring to assess student engagement. It highlights live-coding, reporting that students found it helpful for maintaining focus during screencasts. Despite challenges with course organization, the overall sentiment of the paper towards live-coding is **Positive**. 7) The paper [22] offers practical advice and strategies for teaching introductory programming courses, compiled from the experiences of four seasoned teachers. It presents 26 tips to enhance teaching effectiveness and student learning. The authors advocate for the 'Code Early and Often' approach, suggesting that live-coding be included in programming practices, noting its benefits for hands-on learners. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 8) The paper [9] studies the effectiveness of live-coding versus static code examples in a mid-level programming course. Using video analysis and student surveys, the study qualitatively evaluates teaching methods and student preferences. The authors note that live-coding helps students understand the development process, while static examples enhance code reading and tracing abilities. The paper highlights benefits of both approaches, resulting in an overall sentiment towards live-coding that is **Neutral**. 9) The paper [7] evaluates live-coding in computer science lessons from students' perspectives using semi-structured interviews with 10 first and second-year IT students at Westerdals Oslo ACT. The study identifies benefits such as ease of asking questions and thorough ex-

planations, but also challenges like fast pace and unsatisfactory explanations. Therefore, the overall sentiment of the paper towards live-coding is **Neutral**. 10) The paper [8] describes a new university course for teaching fundamental software engineering concepts to novice students, emphasizing maintainable and scalable code. Live-coding is used in five plenary lectures to introduce theoretical concepts with practical examples. The course's effectiveness was assessed through student feedback, homework performance, and final exam results. The authors highlight the importance of practical examples, favoring live-coding. Thus, the overall sentiment of the paper towards live-coding is **Positive**.

For post-COVID, the following papers were looked at: 1) The paper [18] investigates the impact of live-coding versus traditional static-code pedagogies on students' programming processes and performance in an introductory computer science course. Using two student groups with the same syllabus and instructor, the study collected data through exams, assignments, and feedback. The findings show no statistically significant differences between the groups, indicating that live-coding may not offer the perceived benefits over static code examples. Thus, the overall sentiment of the paper towards live-coding is **Neutral/Negative**. 2) The paper [11] examines how various editor features affect student experiences and perceptions in a creative coding course. Aiming to improve IDE design for educational purposes, the study uses a two-year longitudinal design with log analysis and surveys. The authors express a positive sentiment towards live-coding, noting its benefits for novices and creative contexts. Thus, the overall sentiment of the paper towards live-coding is **Positive**. 3) The paper [1] examines the effectiveness of live-coding in computer science education, comparing it with traditional PowerPoint-based methods. Using examples from courses at Boston University and California State University, Chico, the study highlights live-coding's interactive learning atmosphere and proposes a statistical method to evaluate its impact on student performance. Student feedback overwhelmingly supports live-coding, indicating a **Positive** sentiment towards this approach. The paper [19] explores the impact of live-coding versus static-code examples in introductory computer science, gathering data on student grades, programming processes, and lecture questions. The results show no significant differences in grades, with a slight advantage for static-code examples. Although live-coding improved adherence to programming processes, the static-code group performed slightly better on assignments and exams. Thus, the overall sentiment of the paper towards live-coding is **Neutral**.

3 Data

We used Google Scholar to find research papers on live-coding and alternative forms of teaching first-year programming in computer science. Specifically, we used the following query on Google Scholar: "comparative study between 'live-coding' and static example in 'computer science' 'first year"'. We used the above search query for the three different time ranges of 2017-2019 (pre-COVID), 2020-2022 (during COVID) and 2023-2024 (post-COVID).

For the search query and the time range 2017-2019, we received 27 results. Among these, 5 were books, 3 were music-related, and 1 each were related to food, love, chemistry, and arts. Additionally, there was 1 tutorial schedule and 7 unrelated results. One result was a duplicate. Consequently, 21 results were discarded, leaving 6 relevant results for our research. For the search query and time range 2020-2022, we found 44 results. Among these, 5 were books, 18 were unrelated to live-coding, 1 might be related but is inaccessible, and 1 was psychology-related. One link was broken, 1 was psychology and music-related, 1 was math-related, 2 were music-related, and 1 was physics-related. One paper mixed slides and live-coding but did not include any additional information for us to draw conclusions on the sentiment of live-coding. Consequently, 32 results were discarded, leaving 12 relevant results for our research. For the search query since 2023, we received 12 results. Among these, 4 were related to computer science but not to live-coding. Three were music-related, and 1 might be related to live-coding but is inaccessible. Consequently, 8 results were discarded, leaving 4 relevant results for our research. We summarize our findings in Table 1.

Table 1: The first row indicates the number of papers with positive sentiments towards live-coding. The second row indicates the number of papers with neutral or negative sentiments towards live-coding.

	Pre-COVID	During COVID	Post-COVID
Positive Sentiment on Live Coding	5 [3], [5], [2], [15], [4]	8 [21], [12], [20], [10], [6], [22], [17], [8]	2 [11], [1]
Non-Positive Sentiment on Live Coding	1 [16]	4 [13], [14], [9], [7]	2 [18], [19]

4 Results

The results of the FET are presented in Table 2. This table reports the p-value, odds ratio, and confidence interval for running the FET in three different comparisons: 1) Pre vs. During COVID, 2) Pre vs. Post COVID, and 3) During vs. Post COVID. FET is a statistical test used to assess whether there are nonrandom relationships between two categorical variables in a contingency

table. It is particularly useful for small sample sizes or when expected frequencies are less than 5. FET is a nonparametric test, making it suitable for small datasets. It is primarily used for 2×2 contingency tables (Table 1 presents the data with 2 rows and 3 columns. When conducting FET, we analyze two columns at a time to determine the association between the time periods and the sentiment of the papers) and provides the p-value, which represents the probability of observing a distribution of values as extreme or more extreme than the one observed, assuming the null hypothesis (H_0) is true. For each pairwise comparison (as seen in column 1 of Table 2), we test the following hypotheses:

- H_0: The odds ratio of positive sentiment about live-coding between the two time periods is 1 (no difference in odds).
- Alternative hypothesis (H_1): The odds ratio of positive sentiment about live-coding between the two time periods is not 1 (different odds).

In our study, we examine the association between the time period (pre, during, and post COVID) and the sentiment (positive or negative) of papers about live-coding. The characteristics of our data make FET particularly suitable. We have a relatively small number of papers, especially when split across different time periods and sentiment categories. FET is ideal for such scenarios where other tests might not be reliable due to small expected frequencies. Given the importance of accurately detecting any association between time period and sentiment, the exact p-value provided by FET offers a precise measure of significance without relying on large-sample approximations. The test helps us understand if the observed differences in sentiment across different time periods are statistically significant. By comparing the odds of positive sentiment before and during the pandemic, we can assess how the pandemic influenced the perception of live-coding.

Table 2: Fisher's Exact Test Results for Various COVID Period Comparisons.

Comparison	p**-value**	**Odds Ratio**	**Confidence Interval**
Pre vs. During COVID	0.6148	2.5	[0.2136 29.2543]
Pre vs. Post COVID	0.5000	5	[0.2732 91.5179]
During vs. Post COVID	0.6044	2	[0.2009 19.9137]

As seen in Table 2, the p-values for all comparisons are greater than 0.05, indicating that we fail to reject H_0 in each case. This suggests that there is no statistically significant association between the timeline (pre, during, and post COVID) and the sentiment of papers about live-coding.

5 Discussion

The analysis performed using Fisher's exact test reveals that there is no statistically significant difference in the sentiment of papers about live-coding when comparing the periods before, during, and after COVID. This conclusion is supported by p-values greater than 0.05 across all comparisons. Even though the results lack statistical significance, they offer several valuable insights and points for discussion. The odds ratios for all comparisons are greater than 1, suggesting a tendency towards an increase in positive sentiment about live-coding when compared across all three different rows of Table 2. Row 1 shows an odds ratio of 2.5, indicating that the odds of a paper being positive about live-coding during COVID are 2.5 times the odds pre-COVID. Row 2 shows an odds ratio of 5, suggesting that post-COVID odds are 5 times the pre-COVID odds. Row 3 shows an odds ratio of 2, indicating post-COVID odds are 2 times the odds during COVID. This increased positive sentiment may be due to the necessity-driven adoption and innovation in live-coding practices during the pandemic. Educators and developers may have found new, effective ways to utilize live-coding under the constraints of remote learning and working. The development and adoption of advanced online collaboration tools during the pandemic could have positively influenced the perception of live-coding. Even though the statistical significance is not achieved, the observed trends indicate a growing acceptance and potential benefits of live-coding. Educators and developers should consider integrating live-coding into their practices more extensively. Universities and colleges could invest in training programs and resources to help educators and developers effectively implement live-coding, leveraging the positive experiences reported during and after the pandemic.

6 Conclusion

While the lack of statistical significance suggests that observed trends should be interpreted with caution, the overall positive shift in sentiment towards live-coding pre, during, and post-COVID indicates a potentially valuable development in educational and professional practices. By further investigating and leveraging these trends, stakeholders can improve the effectiveness and acceptance of live-coding methodologies.

Acknowledgement

We gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks.

References

[1] Abbas Attarwala. "Live Coding in the Classroom: Evaluating Its Impact on Student Performance Through ANOVA and ANCOVA". In: *2023 International Conference on Intelligent Education and Intelligent Research (IEIR)*. IEEE. 2023, pp. 1–6.

[2] Lorena A Barba et al. "Teaching and learning with Jupyter". In: *Recuperado: https://jupyter4edu. github. io/jupyter-edu-book* (2019), pp. 1–77.

[3] Charles H Chen and Philip J Guo. "Improv: Teaching programming at scale via live coding". In: *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale*. 2019, pp. 1–10.

[4] Alberto Corbi et al. "X-ray imaging virtual online laboratory for engineering undergraduates". In: *European Journal of Physics* 41.1 (2019), p. 014001.

[5] Tom Crick. *Final draft: Computing education: An overview of research in the field*. 2017.

[6] Kristine L Grayson, Angela K Hilliker, and Joanna R Wares. "R Markdown as a dynamic interface for teaching: modules from math and biology classrooms". In: *Mathematical biosciences* 349 (2022), p. 108844.

[7] Tor-Morten Grønli and Siri Fagernes. "The live programming lecturing technique: A study of the student experience in introductory and advanced programming courses". In: *Norsk IKT-konferanse for forskning og utdanning*. 4. 2020.

[8] Markus Hofbauer et al. "Teaching software engineering as programming over time". In: *Proceedings of the 4th International Workshop on Software Engineering Education for the Next Generation*. 2022, pp. 51–58.

[9] Derek Hwang et al. "A qualitative analysis of lecture videos and student feedback on static code examples and live coding: A case study". In: *Proceedings of the 23rd Australasian Computing Education Conference*. 2021, pp. 147–157.

[10] Nikola Luburić et al. "The challenges of migrating an active learning classroom online in a crisis". In: *Computer applications in engineering education* 29.6 (2021), pp. 1617–1641.

[11] Andrew M Mcnutt, Anton Outkine, and Ravi Chugh. "A Study of Editor Features in a Creative Coding Classroom". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–15.

[12] Maria Medvidova and Jaroslav Porubän. "Program comprehension and quality experiments in programming education". In: *Third International Computer Programming Education Conference (ICPEC 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik. 2022.

[13] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. "A review of worked examples in programming activities". In: *ACM Transactions on Computing Education* 23.1 (2022), pp. 1–35.

[14] Ayesha Naeem Syeda, Rutwa Engineer, and Bogdan Simion. "Analyzing the effects of active learning classrooms in cs2". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 93–99.

[15] Mani Pelmo. "Teaching and learning introductory computer programming at the royal university of Bhutan: Factors affecting student performance". PhD thesis. Curtin University, 2019.

[16] Adalbert Gerald Soosai Raj et al. "Role of live-coding in learning introductory programming". In: *Proceedings of the 18th koli calling international conference on computing education research*. 2018, pp. 1–8.

[17] Marisha Rawlins and Pilin Junsangsri. "Refining Competency-Based Grading in Undergraduate Programming Courses". In: *ASEE-NE 2022*. 2022.

[18] Anshul Shah et al. "An Empirical Evaluation of Live Coding in CS1". In: *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*. 2023, pp. 476–494.

[19] Anshul Shah et al. "The Impact of a Remote Live-Coding Pedagogy on Student Programming Processes, Grades, and Lecture Questions Asked". In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 2023, pp. 533–539.

[20] Gaganpreet Sidhu, Seshasai Srinivasan, and Nasim Muhammad. "Challenge-based and competency-based assessments in an undergraduate programming course". In: *International Journal of Emerging Technologies in Learning (IJET)* 16.13 (2021), pp. 17–28.

[21] Ethel Tshukudu, Quintin Cutts, and Mary Ellen Foster. "Evaluating a pedagogy for improving conceptual transfer and understanding in a second programming language learning context". In: *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*. 2021, pp. 1–10.

[22] Xihui Zhang et al. "Teaching introductory programming from A to Z: Twenty-six tips from the trenches". In: *Journal of Information Systems Education* 31.2 (2020), pp. 106–118.

Developing an Enterprise Application Tool to Discover Midwest Job Trends*

Chandra Prakash Bathula and Maria Weber
School Of Professional Studies
Saint Louis University
St Louis, MO 63103
{chadraprakash.bathula, maria.l.weber}@slu.edu

Abstract

Significant growth in Computer and Mathematical Occupations can be seen in the coming decade while outpacing other fields with a double margin of median pay and a high demand for specialized skills driven by technological advancements. According to the Bureau of Labor Statistics (BLS), between 2023-2033, the United States is projected to see a 11% and 7% growth in the Computer and Mathematical Occupations sector with more than 393,000 jobs annually [5] [6] [3]. This paper examines national trends in Computer and Mathematical Occupations, including wages, growth, and a particular focus on the Central Plains region (Iowa, Kansas, Missouri, and Nebraska). The median national annual salary for these roles varies from $101,000 - $104,000, while their regional pay might vary based on geographical development [5] [3]. Among the compared states, Missouri has the highest growth rate in the region, with roles like Data Scientists and Software Developers leading the chart. While Nebraska and Iowa have a moderate growth rate for roles like Data Scientists and Information Security Analysts roles, Kansas is projected to see a high growth rate in Actuaries. In addition, this study underscores the variations in employment per 1000 jobs and growth rates, stressing the need for targeted skill development to increase opportunities in this

evolving and technology-influenced field. Descriptive statistics and visualizations are used to explore regional competitiveness and distributions by providing information to educators, students, and policy makers.

1 Introduction

In the last two decades, we have seen exponential growth in Technology, including Artificial Intelligence (AI), Augmented Reality (AR), and Quantum Computing, among others. Technology and the Internet have become an integral part of our daily lives. For example, global connectivity and communication are made simple thanks to IoT devices, mobile devices, and unlimited internet; shopping saw a shift from on-site to online due to the explosion of web and mobile applications. The total employment in the US is projected to increase from 167.8 million in 2023 to 174.6 million jobs in 2033 [5], with a growth 4%. Primarily, these statistics are at the national level; we will narrow down the article to the central plains, a subregion of the Midwest, comparing growth and pay [5].

During the next ten years, 2023-2033, Computer and Mathematical Occupations tend to see faster growth compared to other occupations according to the Bureau of Labor Statistics (BLS) [5][4]. There is a noticeable difference of approximately $65,000 in the median pay compared to the rest of the occupations as of 2023 [5][4]. Most of these occupations require a bachelor's degree as a minimum educational requirement. However, only a very few choose a master's degree, and the variance in pay will be relative to the degree and position with years of experience [7].

2 National Trends

Total employment in the US during the last two decades starting from 2003 saw an exponential growth, 141.6 million jobs and peaked in 2007 with 149.3 million jobs, as it is the internet era, in almost every sector, Computer Science being the primary focus [7]. After the great recession in 2008, employment growth saw a dip but increased steadily until the 2019 Covid pandemic [5]. The Pandemic saw an unprecedented impact, pushing the unemployment rate to its highest level since World War 2 [7]. The unemployment rate in the US was notably high in early 2021 but improved significantly by the following year [16]. After 2022, employment growth stabilized and has been growing exponentially with a total employment of 167.8 million. In recent years, there has been a noticeable employment shift towards careers in the Computer Science field.

According to the BLS statistics, we can expect substantial openings in Computer and Mathematical Occupations for the next decade reflecting the

growing demand for talent in these areas [5][3]. Even though the overall US employment growth rate is steady for the projected period until 2033, it is expected to have 1.3 million jobs in IT by 2026. Moreover, Microsoft's CELA predicted we will see about 13 million technology-related jobs in the US [18]. Compared with the other occupations, Computer and Information technology with Mathematical occupations tend to see 11% and 7% growth, higher than the nation's average on employment. Artificial Intelligence, the new hype in the market, is said to impact the job market positively [5] [4].

With a promising atmosphere of growing employment in the Computer and Mathematical Occupations, one more advantage is the higher salaries. Computer and Information Sciences and Mathematical occupations have nearly twice the difference in median annual wage with $101,000 – $104,000 compared to the other occupations [4] [3]. Employment growth, high paying roles and technological shift, all these have happened during the last two decades with technological advancements in every way possible, not limited to IoT, AI, Cloud Computing, Web Development, and Big Data. All the above factors constituted the people to choose these growing occupations [9]. These stats are on a national scale, but the growth in these groups, pay scales, and employment per 1,000 jobs varies from state to state.

3 Regional Analysis on Central Plains

In this section, a comparison will be made between the national level to the sub-regional levels to analyze the trends in localized opportunities and concerning challenges. Our primary focus will be on central plains, including Iowa, Kansas, Missouri, and Nebraska. As of 2023, the United States saw an average 3.6% unemployment rate [5]. However, some states vary significantly compared to the national average. States like North Dakota and South Dakota have notably lower unemployment rates, while a higher rate can be seen in Nevada. Even Silicon Valley in California, while primary a tech hub, saw a relatively high unemployment rate, reflecting boarder economic trends rather than just tech-related issues. States like Texas, California, New York, Virginia, and Florida play a crucial role in the growth of Computer and Mathematical Occupations, standing as the top states for employment growth in these fields [5].

Comparing the employment growth in 2023 and 2024, Missouri, Kansas, Nebraska, and Iowa saw varying growth rates. Among these, Missouri had the highest employment growth in Computer and Mathematical Occupations, with significant employment figures and a notable employment rate per 1,000 jobs. Kansas also showed strong performance, with higher numbers in terms of employment per 1,000 jobs. Iowa, though lower in overall growth, still displayed significant employment in these fields, while Nebraska had a higher

employment per 1000 jobs 13]. Excluding Iowa and Kansas, the remaining states in the central plains have growth rates of about 2.0% or more [11][15].

As the employment per 1000 jobs is relatively high in these states, it is highly recommended that students focus on developing competitive skill sets to capitalize on opportunities within Computer and Mathematical Occupations. States like California and Texas, which are at the forefront of growth in these fields, also exhibit strong employment rates, leading to increased competition for available positions [5].

4 Navigating Job Growth and Skills Demand

This paper will focus on Computer and Mathematical Occupations, as defined by the BLS, which includes a diverse range of roles such as computer analysts, data scientists, software developers, statisticians, and others. Approximately 20 positions are part of these Occupations, with multiple names and Standard Occupational Classification (SOC) codes ranging from 15-0000 to 15-2099, reflect the ever-expanding demand for professionals skilled in technology and data analysis [3]. As the shift of digital transformation continues to shape industries around the globe, Computer and Mathematical Occupations are at the forefront of this shift, offering opportunities for those with specialized skills in areas like data science, artificial intelligence, CyberSecurity, and software development. Table 1 shows some of the roles of Computer and Mathematical Occupations. Individual roles within this sector differ, the underlying demand for these skills remains consistent across various positions.

Although there is a continuous demand for technology-driven roles like data scientists, developers, information security analysts, and others, actuaries, Statisticians, and Business Analysts play a key role in Computer and Mathematical Occupations. The foundation for the roles of data scientists, business analysts, and data analysts are the core skills from the areas of advanced statistical analysis and data interpretation. Whilst statisticians use mathematical techniques for data analysis, business analysts apply data insights to guide business decisions and their growth, which are the responsibilities of a data scientist. These foundational roles are crucial as industries become data-dependent. Statisticians, Actuaries, and Business Analysts contribute to developing emerging technologies like Artificial Intelligence (AI), Machine Learning, and predictive analytics, underscoring the importance of acquiring skills in these areas. Their expertise remains in high demand across diverse industries, inspiring and motivating others to delve into these fields.

Table 1: Table of Job Data for Various IT and Statistical Occupations

SOC Code	Position Name	Annual Median Pay (2023)	Median Pay Hourly (2023)	Minimum Education Required	Estimated Jobs 2023	Projected Growth (2023-33)	Growth %
15-1221	Computer and Information Research Scientist	$145,080	$69.75	Masters	36,600	9,400	26%
15-1241	Computer Network Architects	$129,840	$62.42	Bachelors	177,800	23,900	13%
15-1212	Information Security Analyst	$120,360	$57.87	Bachelors	180,700	39,800	22%
15-2051	Data Scientists	$108,020	$51.93	Bachelors	202,900	63,400	34%
15-1211	Computer System Analysts	$103,800	$49.90	Bachelors	527,200	56,600	11%
15-2011	Actuaries	$112,030	$57.60	Bachelors	30,600	6,400	21%
15-1252/53	Software Developer, Software Quality Assurance Analysts	$130,160	$62.58	Bachelors	1,897,100	327,900	17%
15-2041	Statisticians	$104,860	$50.41	Bachelors	48,800	9,900	11%

5 Comparing National Growth Over Central Plains Region

It is evident that Computer and Mathematical Occupations tend to see a higher growth rate compared to the rest of the professions. Students can focus on any of these roles to have a good career for the next decade. From table 2, among all the roles in Computer and Mathematical Occupations, Data Scientist and Information Security Analyst (15-2051 & 15-1212) roles have higher growth with 36% and 33% over the 2023-2033 projected period. Furthermore, this trend continues in the central plains' region as well. Missouri has a projected growth of 37.63% overall for the 2022-32 decade. Iowa, in the next place, has a 3.1% annual growth rate, Nebraska has 2.1%, and Kansas has 1.1% [11][14] [15]. Aligning closely with the National growth rate, Missouri has a 33.97% growth rate for Information Security Analysts in 2022-2032 [13].

Central Plains region can serve as an insightful case study to look at national trends on a local scale. This region mirrors broader national trends in Computer and Mathematical Occupations, as the states exhibit a strong de-

mand for technology and data, reflecting the growing national emphasis. While there is a variation in growth rates, the consistent need for skilled professionals across these states underscores the broader trend of this sector's job expansion in the U.S. The Central Plains region offers a clear view of national trends in specific local economies.

Table 2: Roles Comparison Between National and Central Plains Region (Missouri, Iowa, Kansas, Nebraska)

SOC Code	National Growth Rate (2023-2033)	Missouri Annual Growth Rate	Iowa Annual Growth Rate	Kansas Annual Growth Rate	Nebraska Annual Growth Rate
15-1211	11%	1.02%	1.0%	1.1%	1.02%
15-2011	22%	2.48%	2.3%	2.0%	2.04%
15-1252/53	17%	2.64%	2.3%	0.7%	2.20%
15-2051	36%	3.76%	3.1%	1.2%	2.71%
15-1241	13%	0.72%	0.5%	1%	0.56%
15-1212	33%	3.39%	3.4%	1.1%	2.47%
15-1221	26%	3.25%	NA	1.1%	1.22%

5.1 State-specific job role comparisons:

Missouri:

- Students and Faculty can focus on Computer and Mathematical Occupations a bit more compared to other fields and from table 2, roles like Data Scientists, Information Security Analysts, Software Developers, and QA Analysts in MO will see significant growth in the 2022-2032 projected period [13]. And as per fig 3, MO stands first in rankings for higher annual median pay as well.

- Computer and Information Systems Managers are getting paid higher than the remaining states as per fig1.

- MO has a high growth rate in Software Developers and QA Analysts (15-1252;15-1253) roles, with 26.43%, compared to the nation's average growth rate of 17% as shown in table 2.

Iowa:

- From table 2, next to MO, Iowa has a moderate annual growth rate for Data Scientists with 3.1% and Information Security Analyst roles with 3.4% [15].

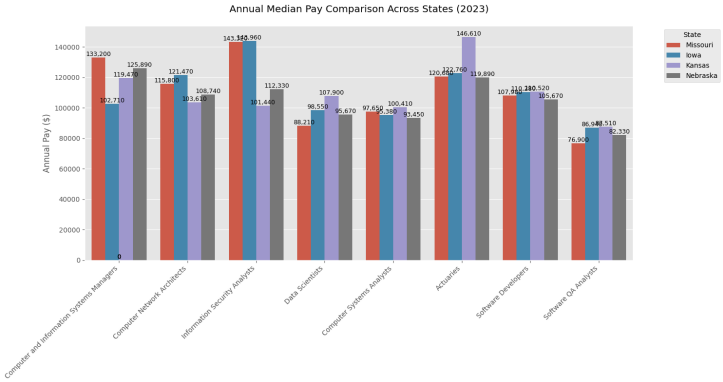


Figure 1: Annual Median Pay comparison over the Central Plains' based on job roles.

Kansas:

- Compared to other roles, Software Developers (15-1252) have lesser demand than other states, with only a 0.7% annual growth rate. Overall growth rates are moderate, and among them, Actuaries (15-2011) tend to show a high 2.0% annual growth rate [11].

Nebraska:

- The growth rate is steady. The roles of Information Security Analysts and Data Scientists tend to have good annual growth rates of 2.47% and 2.71% [14].

Takeaways:

- Missouri job seekers can focus on IT-related careers as they tend to have a higher growth rate. Missouri, Iowa, and Nebraska have good growth rates in Data Scientist and Information Security Analyst roles. Kansas job seekers can shift their focus to Actuaries and analytics-related careers.

All the roles that were discussed above revolve around diverse skill sets. Be it a Computer and Information Research Scientist, Computer Network Architect, or Data Scientist, they all rely on multiple programming languages, Cloud Computing, and Analytical Software and tools [9]. Some of the standard tools and technologies of these roles mentioned in table 3.

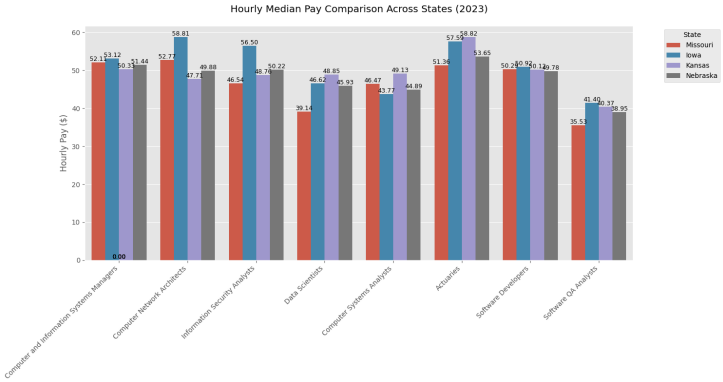


Figure 2: Hourly Median Pay comparison over the Central Plains' based on job roles.

Table 3: In Demand Skills

Category	**Skills**
Programming Languages	R, SQL, Python, C#, C++, Java, JavaScript
Cloud Computing	AWS, Microsoft Azure, Google Cloud Platform
Shell Scripting	Bash, Linux, UNIX
Analytical Tools	Tableau, Power BI, RStudio, Excel
Network & Communication	Wireshark, LANs, WANs, Packet Analyzers, Network Implementation
Development & Testing	SAS, JIRA, Jenkins, DevOps, Selenium, Git, Postman
Machine Learning & Data Science	Scikit-learn, NumPy, TensorFlow

6 Impact of AI on the Job Market

In the two years of the pandemic, from 2020-2022, the world saw a significant loss of lives and livelihood. That includes the IT sector as well, which introduced work from home. People started to work typically, and the job market seemed to be expected until the introduction of OpenAI's ChatGPT in November 2022 [2], followed by Bard, which later rebranded as Gemini and Claude in the very next year. All these AI Chatbots has versatility and functionalities that have shifted the workflow of software organizations. Even though AI was introduced decades ago in Product recommendations, Movie Suggestions, and Games, people did not notice it [1]. After these, Chatbot's introduction with

```
Summary Statistics:
          Annual_Pay                                Hourly_Pay
                mean        std    min     max        mean   std    min    max
State
Iowa     $110,256.25 $18,427.34  86940  143960     $51.09 $6.58 $41.40 $58.81
Kansas   $109,683.75 $17,508.07  87510  146610     $49.26 $5.00 $40.37 $58.82
Missouri $110,467.50 $22,429.04  76900  143320     $46.78 $6.35 $35.53 $52.77
Nebraska $105,496.25 $14,435.47  82330  125890     $48.09 $4.65 $38.95 $53.65

Roles with Highest Pay Variation Across States:
                                                    std        mean
Role
Information Security Analysts                 $21,682.80 $125,262.50
Computer and Information Systems Managers     $13,009.66 $120,317.50
Actuaries                                     $12,807.33 $127,485.00
Data Scientists                                $8,142.24  $97,582.50
Computer Network Architects                    $7,841.86 $112,405.00

State Rankings by Average Annual Pay:
State
Missouri   $110,467.50
Iowa       $110,256.25
Kansas     $109,683.75
Nebraska   $105,496.25
Name: Annual_Pay, dtype: float64
```

Figure 3: Descriptive Statistics for the Annual Pay, Hourly Pay, Highest Pay Variation and State Rankings

the capabilities that solve complex computing problems scared people from the job perspective [17].

Organizations started layoffs by leveraging the Large Language Models (LLMs) and Retrieval Augmented Generation (RAGs) on their work to automate and cut expenses. Even before reaching the peak potential of AI, the job market seems so challenging. Nevertheless, it will not be the same or result in a negative trend. On a positive note, even if AI is about to disrupt 40% of traditional job roles, it is about creating 60% of newer roles [9] [10]. The report of Microsoft's CELA said that the software field is expected to witness 13 million Data Science roles globally by 2030, and 20-50 million job roles are new in the software field, like AI Ethics Compliance, AI monitoring, AI consultant, Generative AI Developer and Prompt engineer, especially in the sectors like healthcare, green energy, and education [2] [10] [4] [18].

Due to the development of AI, we need the workforce to see the product or productivity in the software field. For example, if a research organization is working on a groundbreaking technology, it can be a product or software application that leverages machine learning. The path will include data collection, data cleaning, data engineering, data analysis, model building, application development with that ML model, deploying it on a server, be it a cloud or

physical server, and performance monitoring now and then as per traffic. Alternatively, if it is a product that is not software, we will have a few more steps of building the product, transportation, marketing, and sales [8]. All this work will be done in the above-mentioned roles, such as Computer and Information Research Scientist, Information Security Analyst, Data Scientist, Statistician, Computer Systems Analyst, Computer Network Architect, and Software Developer. Even a single AI wrapper creates multiple roles between the idea and the product.

7 Strategic Recommendations

This section of the paper highlights actionable strategies tailored for universities, students, and policy markets to address the exponential growth and shifting dynamics of Computer and Mathematical Occupations. It compares national and regional job trends, skill demands, and the impact of AI on the job market.

7.1 Recommendations for Universities and Faculty

As the demand is for Computer and Mathematical Occupations nationally and regionally, updating the existing curriculum to introduce more relevant concepts that reflect the roles of Computer and Mathematical Occupations can be helpful [5] [4]. Universities and Faculty should focus on this sector to train the students accordingly to meet the job market and be competitive. Below are suggestions that can help faculty to navigate students:

- Introducing specialized courses in Cybersecurity, Machine Learning, and AI to cater to roles like Data Science (15-2051) and Information Security Analyst (15-1212), which are expected to grow 36% and 33% by 2033.
- Incorporating hands-on training on practical tools and technologies, such as Analytical software like Tableau and Power BI, Big data software like Data Bricks, and Cloud Computing in Amazon Web Services, Google Cloud Platform, and Microsoft's Azure[18] [19].
- Encourage students to pursue specializations and certifications such as Associate of the Society of Actuaries (ASA), Certified Information Systems Security Professional (CISSP), Chartered Enterprise Risk Analyst (CERA), AWS Solutions Architect, and other cloud specialization certifications [18] [19].
- Equip students in central plains states to target localized job opportunities by highlighting state-specific opportunities, such as Software Developers in MO and demand for Data Scientists in Nebraska.

7.2 Guidance for Students

Students should work extra hard to survive this evolving software field. People choose IT for multiple reasons, including high-paying jobs, to create new software applications and innovate new products. To have employment in the Computer and Mathematical Occupations sector, students should focus on in-demand skills and tools. Some suggestions that can help students in entering this field are:

- From Table 3, we can see in-demand skills like cloud platform operations, machine learning frameworks (TensorFlow, Scikit-learn), programming languages (C++, Python, R), and certifications per the job market, and it is suggested that students focus on these skill sets.
- Build portfolios showcasing proficiency in skills and technologies, contribute to open-source projects to gain visibility and hands-on experience, and use platforms like LinkedIn, Twitter, and GitHub to connect with industry leaders [12].
- Target promising job roles based on growth rates nationally and regionally and acquire expertise in emerging roles like AI prompt Engineering and AI monitoring, which are expected to see 60% new job roles by 2030 [9] [10].

7.3 Suggestions For Policy Makers

Policymakers can bridge the gap between students and faculty by tackling the evolving job market. Some of the suggestions are:

- Designing interdisciplinary courses by involving Computer Science, Business applications, and Mathematical concepts in STEM education helps students understand the complete architecture of organizations and how to design a software solution. Online platforms with these interdisciplinary courses can help educate people who cannot afford tuition fees.
- Collaborating with research institutes and top companies to provide research or internship opportunities for students during the summer or after graduation to get Hands On experience.
- Providing Tax benefits for organizations that establish local research and training hubs, which provide job training and certifications.

8 Job Search Platform

This research paper proposes developing a tool to help students and job seekers find growing job roles based on education, pay range, and location. The application will be in a way in which a user will be first asked to create a profile with all the relevant personal and academically relevant information. That includes portfolio, GitHub, Medium, and Dribbble Hyper-Links, which showcase their coding skills and expertise in their field. There will be a home page that describes all the functionalities of the application. Then a Job Search Page will be available in which there will be a search option to look for a growing job role based on their education and location through filters. Filter to view the job roles growth from national-wide to regional-wide and can be further deduced to state. Next step is to choose a Job role, growth percentage locally and nationally, Job Description, Responsibilities, Relevant Skills, and Pay variance with the help of BLS data. Then, the page for job openings will appear based on the location applied in the filter option with the help of Google's Cloud Talent Solutions Job Search API and CoreSignal API. People's information based on the company can be pulled with the help of LinkedIn API, which can help network with concerned personnel. Moreover, a community discussion forum can also be created to notify people of updates outside the application. This platform, as shown in fig4, will be beneficial for STEM graduates, students, and people who are seeking jobs.

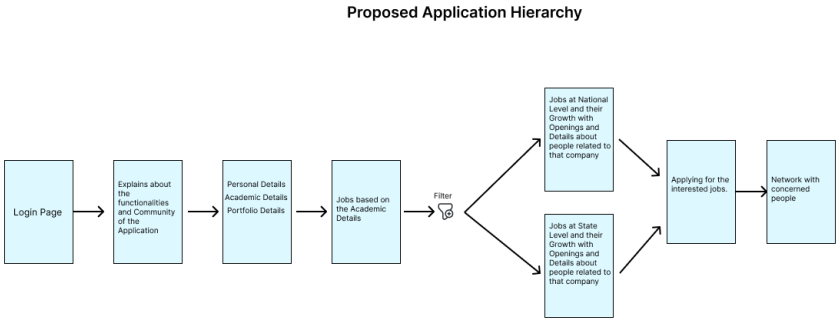


Figure 4: Hierarchical Diagram for Job Search Platform.

9 Conclusions

Analysis of this paper concludes that the Computer and Mathematical Occupations remain a faster growth sector in this evolving software job market, which offers a median average of approximately $105,000 nationally, which is more than any other occupation. Even though tech hubs like California, Texas, and Virginia lead in the growth of these occupations, the Central Plains region, Iowa, Kansas, Missouri, and Nebraska, has stable growth and affordability for professionals in fthese fields with slow growth annually and overall, in the projected period.

The emerging hype around newer technologies like Artificial Intelligence, Data Science, and Cyber Security highlights the importance of equipping the workforce with relevant skills. Policy Makers and educational institutions must prioritize technical advancements aligning with regional job growth to maintain competitiveness in evolving sectors.

While the modest growth rate is at 10% nationally, these discussed job roles and fields offer promising stability and rewarding careers. Success in this will depend on staying updated with emerging technologies and addressing the skill gaps to effectively meet industry needs and standards. Even though this paper examined based on the available statistics, this can change based on the innovations that enter the market each day.

Note:

- Annual and Hourly Pay Medians are taken instead of the mean because the mean is more prone to outliers, and the median will not be affected by one huge value.
- All the statistics of Central Plains states are from their respective state websites, and the national statistics are from the BLS website.
- Missouri's growth rate for the period 2023-2033 was taken as an annual average by dividing it by the total number of years to compare it with the rest of the states.
- Standard error is the smaller the value the closer to the actual value.

References

[1] Xavier Amatriain and Justin Basilico. "Recommender Systems in Industry: A Netflix Case Study". In: *Recommender Systems Handbook*. Springer US, 2015, pp. 385–419. ISBN: 9781489976376.

[2] Zhiqing Bian. "Research on the Impact of Artificial Intelligence on the Labor Market". In: *Highlights in Business, Economics and Management* 24 (2024), pp. 1036–1041.

[3] Bureau of Labor Statistics. *Computer and information technology occupations*. September 6, 2023. 2023. URL: https://www.bls.gov/ooh/computer-and-information-technology/home.htm.

[4] Bureau of Labor Statistics. *Computer and Mathematical Occupations*. April 4, 2023. URL: https://www.bls.gov/oes/2023/may/oes150000.html.

[5] Bureau of Labor Statistics. *Home : U.S. Bureau of Labor Statistics*. January 31, 2019. 2019. URL: https://www.bls.gov/opub/geographic-profile/.

[6] Bureau of Labor Statistics. *Math Occupations : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics*. September 4, 2019. 2019. URL: https://www.bls.gov/ooh/math/home.htm.

[7] Helen C. Connolly and Peter Gottschalk. *Differences in Wage Growth by Education Level: Do Less-Educated Workers Gain Less from Work Experience?* Tech. rep. 2331. IZA Discussion Paper, Sept. 2006. DOI: 10.2139/ssrn.937356. URL: https://ssrn.com/abstract=937356.

[8] Felix Dobslaw et al. "The Gap between Higher Education and the Software Industry—A Case Study on Technology Differences". In: *Proceedings of the 5th European Conference on Software Engineering Education*. 2023, pp. 11–21.

[9] Minghao Duan. "The Role of Innovation in Economic Growth and How Techno-logical Advancements Transform Industries and Employment". In: *Proceedings of the 2023 4th International Conference on Big Data Economy and Information Management*. 2023, pp. 658–661.

[10] H. Ekelund. *Why there will be plenty of jobs in the future - even with AI*. February 26, 2024. 2024. URL: https://www.weforum.org/stories/2024/02/artificial-intelligence-ai-jobs-future/.

[11] Kansas Department Of Labor. *Kansas Labor Information Center (KLIC)*. 2024. URL: https://klic.dol.ks.gov/vosnet/lmi/.

[12] Mehrdad Maghsoudi. "Uncovering the skillsets required in computer science jobs using social network analysis". In: *Education and Information Technologies* 29.10 (2024), pp. 12759–12780.

[13] Missouri Economic Research and Information Center. *Home page | Missouri Economic Research and Information Center*. 2024. URL: https://meric.mo.gov/.

[14] Official Nebraska Government Website. *Home*. 2017. URL: https://www.nebraska.gov/.

[15] Official State of Iowa Website. *Occupational Projections | Iowa Workforce Development*. 2024. URL: https://workforce.iowa.gov/labor-market-information/occupations/occupational-projections.

[16] R. Pandita. "Internet a change agent: An overview of internet penetration and growth across the world". In: *International Journal of Information Dissemination and Technology* 7.2 (2017), p. 83. DOI: 10.5958/2249-5576.2017.00001.2.

[17] Ayisha Tabbassum et al. "The Impact of AI on Future Employment Patterns". In: *International Journal of Global Innovations and Solutions (IJGIS)* (2024).

[18] Benneth Chukwuemeka Uzoma and Isokpehi Bonaventure Okhuoya. *A Research On Cloud Computing*. 2022.

[19] D. Wiershem, G. Zhang, and C. R. Johnston. "Information Technology Certification Value: An Initial Response from Employers". In: *Journal of International Technology and Information Management* 19.4 (2010). DOI: 10.58729/1941-6679.1095.

Domino Tilings: Projects and Assignments for Students*

Keith Brandt and William P Klasinski
Department of Mathematics, Analytics, and Technology
Rockhurst University
Kansas City, MO 64110
keith.brandt@rockhurst.edu

Abstract

We describe our work to solve several domino tiling questions. This setting provides a wide variety of questions that can be studied by students. Skills used include branching, loops, working with lists, recursion, and procedural programming.

1 Introduction

By *domino tiling*, we mean the following: Let m and n be positive integers where at least one is even. For brevity, we will often say tiling instead of domino tiling. Consider the $m \times n$ grid, which we will call "the board," that contains m rows and n columns. Our general task is to count the number of ways to place $\frac{mn}{2}$ dominoes on the board so that each domino covers two adjacent squares. Figure 1 shows an example tiling of a 3×4 board.

In this paper, we describe our solutions to a few domino tiling questions, and we pose a few more questions that could be considered. Skills used can be found in an introductory programming text such as [2].

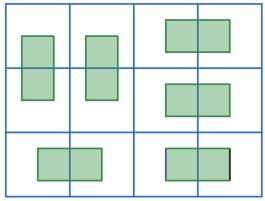


Figure 1: Example Domino Tiling

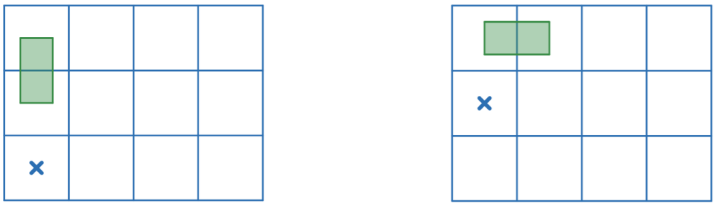


Figure 2: Placement of first domino on a board

2 Setup and the Algorithm

We view the board as an $m \times n$ table whose entries are positive integers indicating the order of placement of the dominoes.

2.1 Domino Placement

The algorithm to place dominoes follows a simple recursive structure: Each time a new domino is to be placed on the board, the algorithm considers placing it either vertically or horizontally. Once an orientation is determined, the domino is placed, the board is updated, and the recursion continues until no more dominoes can be placed. At that point, that algorithm has reached a dead end (backtrack) or the board is full.

Determine the location to place the new domino as follows: Scan the columns, in left-to-right order, from top to bottom, until we find an empty square. Using this approach, the first domino placed will lie in the upper-left corner of the board, in one of the two positions shown in Figure 2.

In each of the boards in Figure 2, the location to consider for the second domino is indicated with an "x". There is only one option, horizontal, for the second domino in the board on the left, whereas both horizontal and vertical

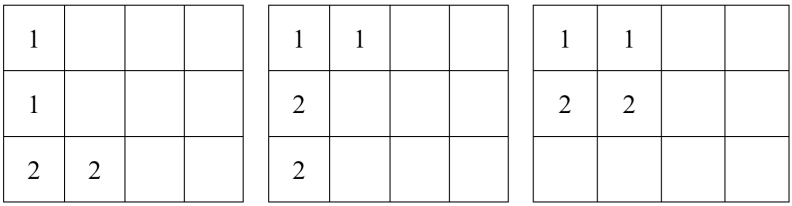


Figure 3: Placing the first two dominoes on a 3×4 board

placements can be made for the second domino in the board on the right. Of course, before attempting to place a domino in either orientation, the algorithm checks to see that there is sufficient space to make the placement.

2.2 Tiling Representation

When we place the k-th domino, we update the board by putting the integer k into the two cells corresponding to the squares covered by that domino. To illustrate the process, we return to the boards described in Figure 2. For the board on the left, the second domino must be placed horizontally. For the board on the right, the second domino can be placed in either orientation. By convention, we will always attempt a vertical placement first. The three possible ways to place the first two dominoes are shown in Figure 3.

3 The Questions

We now list questions we have studied and describe their solutions.

3.1 Square Boards

Our first task, the one that motivated this article, is to count the number of tilings of a $2n \times 2n$ board, where $n \geq 1$. To increase efficiency, we exploit the symmetry of the square and initialize the board to begin with a vertical domino. We can then double the final count. The solution for the first few values of n are shown in Table 1.

Table 1: Domino tilings of a $2n \times 2n$ board

$2n$	2	4	6	8
Tilings	2	36	6,728	12,988,816

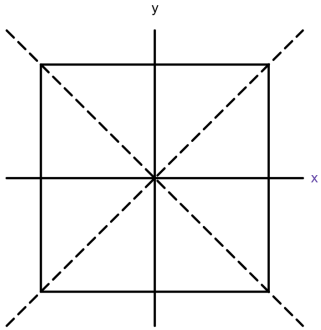


Figure 4: Symmetries of the square

The values shown in Table 1 count all possible tilings of a $2n \times 2n$ board. However, many of the tilings counted are simply flips and rotations of each other. Our next task is to produce a list of tilings that are distinct in the sense that no tiling on the list is a flip or rotation of any other tiling on the list.

The group of symmetries of a square is known as the dihedral group (see [1], page 31). Consider a square centered at the origin, as shown in Figure 4. The dihedral group consists of eight elements, which are listed in Table 2.

Table 2: Elements of the dihedral group

Element	Description
I	Identity
ρ	Rotation by 90 degrees clockwise
ρ^2	Rotation by 180 degrees
ρ^3	Rotation by 270 degrees clockwise
h	Flip across line $y = 0$
v	Flip across line $x = 0$
d_1	Flip across line $y = x$
d_2	Flip across line $y = -x$

Say two tilings are *equivalent* if one can be obtained from the other via a flip or rotation from the dihedral group. We now modify our algorithm as follows. We build a list of tilings we call the *essential list* that has the following property: No two elements in the essential list are equivalent to each other. The essential list will begin with the first tiling found by the algorithm. For all subsequent tilings found by the algorithm, we form all possible flips and

rotations and then check to see if any of these are equivalent to the tilings that are already on the essential list. If none of the flips or rotations of the tiling being considered are on the essential list, then it is added to the essential list.

There is one subtle point that must be addressed. It is easy to write functions that carry out the various flips and rotations. However, once an element of the dihedral group is applied to a tiling, the integers that indicate the order of placement become irrelevant (or worse, a distraction). We devised a tool we call the *signature* that allows us to compare tilings. The signature of a tiling is a nested list that indicates the structure of each row of the tiling. Determine the signature of a tiling as follows: Scan each row from left to right while comparing neighbors.

- If two neighbors are equal, we have come across a horizontal domino, so add a 2 to the list for that row and advance the index by 2 (to consider the next two entries in that row).
- If two neighbors are not equal, we know the first of these neighbors is part of a vertical domino. Add 1 to the list for that row and advance the index by 1.

For example, the signature of the tiling in Figure 1 is [[1,1,2], [1,1,2], [2,2]]. Note that if 2 tilings have identical structure (not paying attention to order placed), then their signatures will be equal. We use signatures to compare candidate tilings with the tilings in the essential list before adding them to the essential list. Our results for essential tilings of boards are given in Table 3.

Table 3: Essential tilings of a $2n \times 2n$ board

$2n$	2	4	6	8
Tilings	1	9	930	1,629,189

3.2 Rectangular Boards

Minor adjustments to our algorithm for square boards allow us to study tilings of more general $m \times n$ boards where at least one of m or n is even. Table 4 shows our results for $m < n \leq 8$.

The reader may recognize the values in the row for $m = 2$. It is an easy exercise to show that the number of domino tilings of a $2 \times n$ board correspond to Fibonacci numbers. Given $n \geq 3$, we form all tilings of a $2 \times n$ grid by appending two horizontal dominoes to all tilings of a $2 \times (n-2)$ board and by appending one vertical domino to all tilings of a $2 \times (n-1)$ board.

Table 4: Tilings of $m \times n$ boards

$m \backslash n$	**3**	**4**	**5**	**6**	**7**	**8**
2	3	5	8	13	21	34
3		11	*	41	*	153
4			95	281	781	2,245
5				1,183	*	14,824
6					31,529	167,089
7						1,292,697

We can also apply some flips and rotations to rectangular boards that are not square. We may flip a tiling both horizontally and vertically, and we may rotate it by 180 degrees. By applying these flips and rotations, we can count the essential tilings using the method described for square boards. Our results for essential tilings of $m \times n$ boards are given in Table 5.

Table 5: Essential tilings of $m \times n$ boards

$m \backslash n$	**3**	**4**	**5**	**6**	**7**	**8**
2	2	4	5	9	12	21
3		5	*	14	*	46
4			33	98	230	658
5				329	*	3,818
6					8,121	42,837

3.3 Boards with Holes

By removing one square from the board, we can study tilings of $m \times n$ boards where m and n are both odd. In our study, we restricted our attention to square boards.

If certain squares are removed, the resulting board will have no tilings at all. Define the parity of a square to be the parity (even or odd) of the sum of its indices. That is, the parity of the square that lies in the i-th row and j-th column is the parity of the sum $i + j$. For example, the squares along the main diagonal all have even parity since their indices are the same. When n is odd, an $n \times n$ board will have $\frac{n^2+1}{2}$ squares with even parity and $\frac{n^2-1}{2}$ squares with odd parity. In short, we must remove a square that has even parity, since each domino placed will cover neighboring squares with opposite parities.

The approach here is to place a special character in the cell corresponding to

	1	2	3
1	192	*	192
2	*	112	*
3	192	*	196

	1	2	3
1	96	*	100
2	*	56	*
3	100	*	30

Figure 5: Tiling 5×5 boards. Total tilings (left) and essential tilings (right)

	1	2	3	4
1	100,352	*	100,352	
2	*	60,032	*	66,304
3	100,352	*	103,984	*
4	*	66,304	*	75,272

	1	2	3	4
1	50,176	*	100,352	
2	*	30,016	*	33,152
3	100,352	*	51,992	*
4	*	33,152	*	9,466

Figure 6: Tiling 7×7 boards. Total tilings (left) and essential tilings (right)

the square that is removed. The algorithm can then be modified to work around that square. Not surprisingly, the number of tilings of such boards depends on the square removed. Figure 5 and Figure 6 give our results for 5×5 boards and 7×7 boards respectively. The number in each square indicates the number of tilings with that square removed. Due to symmetry, it suffices to show the results for the upper-left portion of the board. For essential tilings, the choice of square to be removed limits which flips and rotations may be applied.

3.4 Additional Questions to Consider

We restricted our study to counting domino tilings of boards with an even number of squares—and boards with an odd number of squares with one square removed. A first natural question would be to complete our study of boards with one square removed (we limited our attention to square boards). Other questions to study could be tilings of rectangular boards with multiple squares removed (paying close attention to parity). Furthermore, tilings of a variety of non-rectangular boards could be studied. The Wikipedia page for domino

tilings [8] gives an example tiling of an Aztec diamond. We could also allow dominoes to bend and then study tilings of surfaces such as a cylinder or a cube. Another challenge would be to develop more efficient algorithms to count tilings of larger boards. Considering shapes other than dominoes such as polyominoes [9] opens another world of possibilities.

4 Technical Aspects

We wrote our code in Python on the Cocalc platform [6]. We represent boards and tilings as nested lists, with a list for each row of the board. In our original implementation for square boards, the list is initialized to be empty. As dominoes are placed, integers are appended to the appropriate lists. When studying boards with a square removed, we initialize the board to have zeros in every location, with the exception of a special character for the removed square. The zeros are rewritten as dominoes are placed. On Cocalc, our code takes between 30 to 60 minutes to run for the largest boards listed in our tables. (Our program was not able to count all essential tilings of a 7×8 board in a reasonable time.) This paper grew out of a student project, where the student and faculty mentor wrote the various pieces of the code independently and then compared results. We are happy to share our code with any interested readers.

5 Conclusion

Domino tilings provide a wealth of questions to study. The tools used to solve them are well within the reach of students in first-year programming classes. The solutions apply many of the skills studied in such classes, and, for larger boards, illustrate the power of computers. Furthermore, students can check their work, as many of the values are known. The sequences in Table 2, Table 4, and the first row of Table 6 can be found in the Online Encyclopedia of Integer Sequences [5] (Sequences A00403, A099390, and A001224 respectively). Our values in Table 5 are contained in a larger table published by Klarner and Pollack in [4]. A closed-form formula for the values in Table 5 is given in [3] and [7]. Klarner and Pollack attribute a different but similar looking formula to Knuth.

References

[1] Joseph A. Gallian. *Contemporary Abstract Algebra, 8th Edition.* Pacific Grove, California: Thompson Brooks/Cole, 2013.

[2] Mark J. Johnson. *A Concise Introduction to Programming in Python, 2nd Edition*. Boca Raton, Florida: Chapman and Hall/CRC Press, 2018.

[3] Pieter Kasteleyn. "The Statistics of Dimers on a Lattice I: The Number of Dimer Arrangements on a Quadratic Lattice". In: *Physica* 27.12 (1961), pp. 1209–1225. DOI: https://doi.org/10.1016/0031-8914(61)90063-5.

[4] David Klarner and Jordan Pollack. "Tilings of Rectangles with Fixed Width". In: *Discrete Mathematics* 32.1 (1980), pp. 45–52. DOI: https://doi.org/10.1016/0012-365X(80)90098-9.

[5] OEIS Foundation, Inc. *The Online Encyclopedia of Integer Sequences*. https://oeis.org.

[6] Sage Math, Inc. *CoCalc – Collaborative Calculation and Data Science*. URL: https://cocalc.com.

[7] H.N.V. Temperley and Michael E. Fisher. "Dimer Problem in Statistical Mechanics–An Exact Result". In: *Philosophical Magazine* 6.68 (1961), pp. 1161–1163. DOI: http://dx.doi.org/10.1080/14786436108243366.

[8] Wikipedia Contributors. *Domino Tiling*. URL: https://en.wikipedia.org/wiki/Domino_tiling.

[9] Wikipedia Contributors. *Polyomino*. https://en.wikipedia.org/wiki/Polyomino.

Pedagogical Evaluation of Generative AI Course for Technologists*

Ajay Bandi
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
ajay@nwmissouri.edu

Abstract

Generative AI is a transformative technology that impacts various fields, including software development, data analytics, and cybersecurity. To address this, we have designed and developed a Generative AI course for technologists, integrating foundational knowledge of various Gen AI architecture models with hands-on practical experience using Python libraries, including HuggingFace. This paper discusses the detailed course structure and assessments. A pedagogical evaluation approach is followed to identify the challenges encountered in the course and how to overcome them. The results demonstrate that the Generative AI Course for Technologists effectively equips students with technical expertise and critical thinking skills through a balanced combination of theoretical concepts and practical exercises, such as chatbot development and prompt engineering. The course addresses challenges like hardware limitations and API integration by proposing future improvements, including a dedicated Python module and access to cloud-based GPU tools, ensuring learners are well-prepared to navigate and ethically apply Generative AI in real-world contexts.

1 Introduction

Generative AI represents a paradigm shift in artificial intelligence, transforming fields such as software development, data analytics, and cybersecurity. Table 1 shows the prescribed tasks and corresponding Generative AI tools utilized across the domains of software development, data analytics, and cybersecurity. As these technologies evolve, there is a growing need to prepare technologists [5] to navigate this landscape with a combination of theoretical knowledge, practical skills, and ethical awareness [6, 9].

Table 1: Prescribed tasks for using Generative AI tools across domains.

Prescribed Task	**Generative AI Tool**
Software Development	
Generate boilerplate code and API documentation	GitHub Copilot, Tabnine
Refactor and debug code with suggestions	Amazon CodeWhisperer, Codex
Create test cases and automate unit testing	Testim.io, Diffblue Cover
Generate user stories and task breakdown for agile workflows	ChatGPT, Atlassian AI
Data Analytics	
Generate synthetic datasets for model training or testing	Gretel.ai, ChatGPT
Automate exploratory data analysis and provide data insights	DataRobot, Tableau AI
Create natural language summaries of analytical findings	OpenAI GPT, BigML
Optimize query formulation for large-scale databases	ChatGPT, Perplexity AI
Cybersecurity	
Identify vulnerabilities in source code and suggest remediation steps	GitHub Copilot, DeepCode
Generate synthetic phishing emails for cybersecurity training	OpenAI GPT, ChatGPT
Automate the creation of security policy documentation	Jasper, ChatGPT
Detect and summarize malware behavior from logs or incident reports	Microsoft Azure OpenAI, Splunk AI

To address this need, we developed the course at the Northwest Missouri

State University Generative AI Course for Technologists, which aims to provide learners with a comprehensive understanding of the architecture, tools and applications of Generative AI.

The course curriculum emphasizes both foundational knowledge and applied learning. Students explore key architectural models, including Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), transformers, and Large Language Models (LLMs). These models emphasize a wide range of applications, from generative content creation to advanced natural language processing, providing a solid framework for understanding the internal architecture of Generative AI.

To ensure relevance and engagement, the course includes hands-on activities where students design and implement AI-based solutions. For example, participants build retrieval-augmented AI chatbots using tools such as graph databases and vector databases. These activities enable students to bridge theoretical concepts with practical applications, fostering skills that are directly transferable to industry contexts. Beyond technical expertise, the course includes significant emphasis on the ethical, privacy, and security dimensions of Generative AI. Learners critically assess the implications of deploying AI technologies in real-world scenarios, ensuring they are prepared to address challenges such as bias, misuse, and data protection. A capstone project further reinforces these objectives by requiring students to integrate their learning into a comprehensive, real-world application.

This paper discusses the pedagogical framework, instructional strategies, and outcomes of this course, highlighting how it prepares students to leverage Generative AI responsibly and effectively. The integration of technical rigor, practical application, and ethical considerations offers a model for equipping technologists with the skills necessary to lead in an increasingly AI-driven world.

2 Professional Advisory Team

The decision to offer Crafting a Generative AI Course for Technologists originated from insights provided by our professional advisory team, which consists of 40 industry professionals from the Midwest region, as well as feedback from technical speakers who have delivered guest lectures at our institution. These experts highlighted the growing importance of integrating Generative AI tools into software development workflows [1] to enhance efficiency and boost productivity.

However, they also emphasized the critical need for professionals to effectively evaluate the responses generated by these tools before implementing them in workplace settings. This requires not only technical expertise but

also the ability to design precise prompts that can elicit accurate and reliable outputs. As prompt engineering becomes a cornerstone skill in utilizing Generative AI effectively, equipping technologists with this knowledge is essential to maximizing the potential of these tools while ensuring their responsible use.

This course aims to bridge this gap by providing students with the skills to harness Generative AI tools effectively, evaluate their outputs critically, and design prompts that optimize results. By doing so, the course prepares future technologists to leverage Generative AI responsibly and productively in their professional environments.

3 Course Structure

This section provides details about the course materials. The hands-on worksheets and assignments are available in the GitHub repository[1].

Prerequisites: Students are required to successfully complete Programming II and Database Systems prior to enrolling in the Generative AI course. Proficiency in programming is essential, as this course is designed for technologists with an emphasis on the implementation of Generative AI applications rather than their usage alone. Foundational knowledge of databases is necessary, as the course leverages graph databases to manage extensive datasets for Generative AI applications. This course is primarily designed for junior, senior, or graduate students.

3.1 Module 1: Introduction:

This module provides a foundational understanding of Generative AI, focusing on its definition and explaining the basic principles of how Generative AI works. The module includes hands-on activities for content generation, such as text, images, and audio, while explaining the different input and output formats, where the source code is already provided, and students need to supply the input and observe the output in a Google Colab environment. Learners will explore Python libraries such as Optical Character Recognition (OCR) for extracting text from images, PIL for image processing, Google Text-to-Speech (gTTS) for converting text into speech, SpeechRecognition for processing audio input, cv2 for computer vision tasks, and rembg for background removal.

3.2 Module 2: Generative AI Models and Architectures

This module offers the foundational Generative AI models, their functionality, and diverse applications, offering a comprehensive understanding of the

[1] https://github.com/bandiajay/Generative-AI

underlying mechanics and capabilities of these technologies. It explores key architectures such as Variational Autoencoders (VAEs), which are used for unsupervised learning, Transformers that revolutionized natural language processing, Generative Adversarial Networks (GANs) known for creating realistic data, and Large Language Models (LLMs) that power conversational AI. The module discusses the unique features, advantages, and real-world applications of each architecture, such as content creation, image synthesis, and automated reasoning. This module does not include hands-on exercises, but details theoretical explanations supported by examples to help students appreciate the potential and versatility of these models in various domains.

3.3 Module 3: AI Open Platforms (ChatGPT and HuggingFace)

This module provides an overview of OpenAI, exploring its key milestones, developments, and the evolution of its products, including the role of ChatGPT within the GPT model series. It compares different versions of GPT, highlighting their features, improvements, and applications. Also, the module includes six hands-on worksheets covering key natural language processing (NLP) techniques: sentiment analysis, named entity recognition (NER), text generation, text generation using Hugging Face, text similarity measurement, and language translation. Students are encouraged to try different models from Hugging Face to deepen their understanding and explore various capabilities.

3.4 Module 4: Generative AI Chatbots

This module focuses on designing and testing chatbots, their architecture, practical implementation, and the role of the Neo4j database in enabling intelligent conversational systems compared to the rule-based systems. It provides hands-on experience through a worksheet in a Google Colab environment, where tools such as LangChain, OpenAI, HuggingFace, and Neo4j are used to design and implement chatbots. The module elaborates on the core building blocks of chatbot systems, including graph and vector databases for semantic search and augmented search retrieved (RAG) to provide accurate and context-sensitive responses [3]. It explains how the integration of Neo4j enhances the system's ability to manage complex relationships and structured data. This module covers both the conceptual and technical knowledge needed to create advanced chatbot systems.

3.5 Module 5: Prompt Engineering with Generative AI

This module focuses on Prompt Engineering with Generative AI, educating students on understanding what prompts are and how to design them effec-

tively to achieve the best results from AI models. It covers the different types of prompt engineering techniques and their appropriate use. It explains the common challenges faced in creating prompts and provides advanced techniques for crafting different types of prompts based on specific needs. It highlights Prompt Engineering strategies to address ethical concerns, sharing important ethical considerations and best practices to avoid potential issues and use the technology responsibly. We also focused on how to evaluate the re4sults from generative AI tools using an evaluation framework [2]. This module includes an exercise on using ChatGPT's Playground, where students will experiment with crafting prompts and adjusting its parameters. Microsoft GiHub Copilot is introduced to generate source code for demonstration.

3.6 Module 6: Security Risks and Privacy Concerns using Generative AI

This module addresses the security risks and privacy vulnerabilities associated with Generative AI, highlighting the critical importance of data validation and robust governance. It examines key issues such as data poisoning, where malicious inputs can disrupt model performance; data leakage, which can expose sensitive information; and privacy breaches, often caused by weak data anonymization protocols. The module also discusses contributing factors, such as the large size datasets, the increasing complexity of AI algorithms, and the lack of adequate human oversight during development and deployment. In addition to identifying these challenges, it provides strategic recommendations for mitigating risks, including stricter data handling protocols and enhanced transparency in AI workflows. While the module does not include hands-on exercises, it emphasizes building awareness and fostering responsible AI practices.

3.7 Module 7: Final Project

The project consists of five milestones, requiring teams to submit deliverables collaboratively. Teams must document their project idea, including team details, project title, and a concise description of their approach. They should outline the tools and technologies used, create a high-level architecture, and explain the workflow in detail. Teams will then provide a complete implementation of the project, along with citations, and save their work on a GitHub repository. The project also includes a presentation where all team members are required to upload audio and video recordings, demonstrating key parts of the project. Students are required to provide constructive feedback on other teams' presentations. Some of the sample projects are explained in the next section.

4 Assessments

The students completed a comprehensive series of tasks as part of their learning of the material, which included 13 worksheets, 6 assignments, a couple of in-class activities, 2 exams, a research paper, and the final project. These activities were designed to enhance their understanding of the concepts of Generative AI and its applications.

The worksheets focused on state-of-the-art Generative AI technologies, such as the conversion tools of various input and output formats, Image-to-Text, Audio-to-Text, and Retrieval-Augmented Generation (RAG)-enabled chatbots. Students were tasked with experimenting with various inputs on a given codebase, generating prescribed outputs, and analyzing their learning outcomes throughout the process. This hands-on approach provided valuable practical exposure to cutting-edge technologies. The assignments were primarily theoretical and centered on key topics such as security issues and prompt engineering. These tasks encouraged students to deepen their understanding of foundational concepts while critically analyzing potential challenges in the field.

Two in-class activities were conducted. The first focused on creating prompts to test the models for generating correct outputs, where students designed prompts to solve reasoning problems. The second activity explored factors such as Top-k sampling and penalty settings in ChatGPT. Students recorded their observations while adjusting these values, gaining insights into how these parameters influence the model's behavior.

Two exams covering all the topics ensured that students had a thorough grasp of the subject matter. This assessment also allowed them to provide feedback and suggest future directions for course improvement in the coming semesters. In addition, students collaborated on a research paper under the guidance of a supervisor. These papers focused on current trends in Generative AI, including topics like Fake News Detection, Ethical Concerns, and the role of Generative AI in Software Engineering. This activity fostered teamwork and research skills.

Finally, a team-based final project was developed, utilizing modern Generative AI technologies such as OpenAI and LangChain. This project allowed students to apply their learning to real-world applications, reinforcing their understanding of the field.

5 Pedagogical evaluation

The course has been offered at Northwest Missouri State University as a special project since Spring 2024 for three semesters, during which challenges were observed. Using the pedagogical approach evaluation framework [4, 7, 8], the

challenges faced by students and the strategies employed by educators to overcome them were analyzed.

Research paper:

The research papers focus on diverse applications, techniques, and ethical considerations of Generative AI across various domains. Topics include practical industry implementations of Gen AI in project management, healthcare, dynamic advertising, and agile development, highlighting how Generative AI enhances efficiency, innovation, and user engagement. Technical advancements are explored through areas like prompt engineering, automating technical documentation, software testing, and data curation, emphasizing tools and methods to improve workflows and productivity.

Ethical challenges and concerns are another central theme, with discussions on bias and fairness, AI hallucinations, and deepfake technologies, addressing the risks and implications of AI-generated content. Communication technologies, including chatbots, are examined for their current capabilities and future evolution. Overall, the papers present a well-rounded perspective on Generative AI, blending practical applications, technical insights, and ethical considerations to equip students with a comprehensive understanding of the technology's potential and challenges.

Challenges:

One of the primary challenges is students generating content directly using ChatGPT. To address this, students are required to work incrementally by submitting outlines or prompts for each milestone before submitting the actual paper. This approach encourages students to think critically about their paper topics in advance. More points are allocated to the process of creating the paper, such as outlining and generating prompts, rather than the final content.

Final Project: The final project features a range of generative AI chatbots designed to address real-world challenges across various domains. These include an e-commerce chatbot for Apple's website to enhance customer support, a story teller chatbot for creative writing and entertainment, and the GitHub GuideBot to assist with software documentation. In education, EDUBOT provides personalized learning support, while the interview bot, leveraging retrieval-augmented generation, helps users prepare for interviews. In healthcare, MediAssist offers AI-driven guidance for wellness and medical inquiries. Together, these projects highlight the versatility of generative AI in improving user experiences, automating workflows, and fostering skill development across industries.

Challenges in Final Project Implementation:

As part of their final project, students implemented RAG-enabled chatbots. They encountered two major challenges during the process. The first challenge

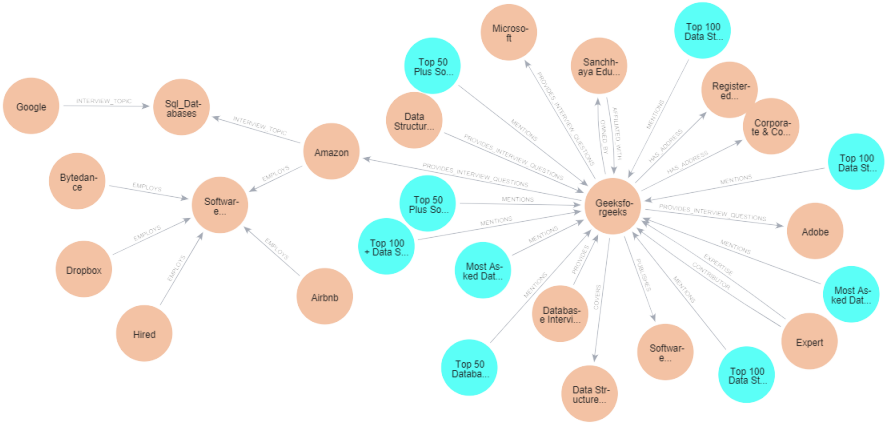


Figure 1: Knowledge graph of the data used to develop the interview bot.

involved converting data into knowledge graphs using Neo4j and LangChain. For intance, the knowledge graph of intervew bot is shown in Figure 1. For websites with large datasets, the process of extracting and converting data into knowledge graphs was particularly time-consuming and computationally demanding. This issue was exacerbated by the use of i5 Core processors, which lack the GPU capabilities required for generative AI chatbot operations. Also, the need to purchase OpenAI API keys further complicated the process due to resource constraints.

The second challenge was ensuring the chatbot's accuracy in providing answers. While these issues were not fully addressed this year, plans are in place to resolve them in the near future, including upgrading hardware and optimizing processes.

Worksheets and Assignments:

The assignments and worksheets in Generative AI using Python cover a range of topics focused on text, speech, and image processing. Tasks include text-to-speech conversion (Google Text-to-Speech), speech recognition, and sentiment analysis to explore natural language processing (NLP) techniques. Students also worked on named entity recognition, text similarity, machine translation, and text generation using tools like Hugging Face API. Optical character recognition (OCR) tasks involved Tesseract and EasyOCR for extracting text from images. Image-based projects included background removal and image colorization, showcasing generative techniques in computer vision. Also, students implemented chatbots, including a retrieval-augmented gener-

ation (RAG)-enabled chatbot, demonstrating the integration of advanced AI models for interactive applications.

Challenges:

Students in the course had understanding of Python programming but faced significant challenges when it came to using Python libraries for generative AI, such as Transformers and LangChain. These libraries, essential for building language model-powered applications, require advanced understanding and integration, which posed difficulties for many students. Also, students encountered issues with API usage, particularly when integrating models like Hugging Face and OpenAI into their assignments. Hardware limitations, such as the lack of GPUs, slowed down model training and processing, while dataset preparation and preprocessing for tasks like creating knowledge graphs or training natural language models were often challenging. Debugging errors in frameworks like LangChain and libraries like Tesseract and EasyOCR also time-consuming. Balancing the accuracy and performance of generative AI applications, especially when developing chatbots, added another layer of complexity. To overcome these hurdles, a dedicated module on Python for generative AI applications could be developed, focusing on the proper use of libraries, understanding APIs, and streamlining the debugging process. Providing access to cloud-based tools with GPU support would also help alleviate hardware constraints, improving the overall learning experience.

Exams and in-class activities Two in-class activities were conducted: the first involved designing prompts to test model outputs for reasoning problems, while the second focused on exploring factors like Top-k sampling and penalty settings in ChatGPT, allowing students to observe how these parameters affect model behavior. Two comprehensive exams ensured students mastered the material and provided feedback for course improvement. Also, students collaborated on research papers under supervision, addressing current trends in Generative AI, such as Fake News Detection, Ethical Concerns, and its role in Software Engineering, fostering teamwork and research skills.

Challenges in exams and in-class activities For the in-class activity, it was challenging for students to understand the various definitions of parameters and observe how the output changes when varying these parameters. Also, during exams, students were allowed to use generative AI tools, but many simply copied and pasted verbose answers without tailoring them to the proper format. As an educator, I emphasized the importance of redesigning prompts to generate clear and understandable answers. I also guided students in evaluating their outputs, including source code problems, by helping them better understand ChatGPT's parameters.

6 Conclusion

This paper presented the design, implementation, and outcomes of the *Generative AI Course for Technologists* at Northwest Missouri State University. The course combines foundational knowledge, hands-on activities, and ethical considerations to equip students with the skills needed to navigate the evolving landscape of Generative AI. By integrating practical exercises, such as chatbot development and prompt engineering, with theoretical concepts like VAEs, GANs, and LLMs, learners gain both technical expertise and critical thinking abilities. Assessments, including worksheets, projects, and research papers, reinforce this holistic approach, ensuring students can apply AI tools responsibly in real-world contexts. The course framework highlights the importance of balancing innovation with ethical awareness, addressing challenges such as bias, security risks, and misuse. As Generative AI continues to transform industries, this curriculum serves as a robust model for preparing future technologists. The paper also highlighted significant challenges in learning and implementation, such as hardware limitations, complexities in integrating Python libraries like Transformers and LangChain, and issues with API usage. To overcome these barriers, future iterations of the course will include a dedicated module on Python for Generative AI applications, emphasizing library usage, API integration, and debugging. Also, providing access to cloud-based GPU tools and streamlining the creation of knowledge graphs will further enhance the learning experience.

References

[1] Ajay Bandi and Hemanth Kagitha. "A Case Study on the Generative AI Project Life Cycle Using Large Language Models". In: *Proceedings of 39th International Confer* 98 (2024), pp. 189–199.

[2] Ajay Bandi and Ruida Zeng. "Evaluation of the Effectiveness of Prompts and Generative AI Responses". In: *International Conference on Computer Applications in Industry and Engineering*. Springer. 2024, pp. 56–69.

[3] Ajay Bandi et al. "Enhancing Generative AI Chatbot Accuracy Using Knowledge Graph". In: *International Conference on Software Engineering and Data Engineering*. Springer. 2024, pp. 157–167.

[4] Richard Hall. "Aligning learning, teaching and assessment using the web: an evaluation of pedagogic approaches". In: *British Journal of Educational Technology* 33.2 (2002), pp. 149–158.

[5] Antonie J Jetter et al. “Training Practitioners for Real-time Product Development Using Generative Artificial Intelligence”. In: *2024 Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE. 2024, pp. 1–11.

[6] Swapna Kumar et al. “The Role of Instructional Designers in the Integration of Generative Artificial Intelligence in Online and Blended Learning in Higher Education.” In: *Online Learning* 28.3 (2024), pp. 207–231.

[7] John M LaVelle, Chris Lovato, and Clayton L Stephenson. “Pedagogical considerations for the teaching of evaluation”. In: *Evaluation and program planning* 79 (2020), p. 101786.

[8] Jelena Maksimović. “Evaluation approach in pedagogical research”. In: *Social Context of education* (2009), p. 89.

[9] Patricia Santos, Keysha Urgel, and Verónica Moreno. “Generative Artificial Intelligence in Teaching and Learning of ICT Engineering Education: A Literature Review and Illustrative Scenarios”. In: *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*. IEEE. 2024, pp. 1338–1343.

The Impact of Course Modality and Size on Learning Outcomes: Applying IaC Principles in IS/Cyber Graduate Course Design*

Annamaria Szakonyi
Information Systems and Cybersecurity
Saint Louis University
St. Louis, MO 63103
annamaria.szakonyi@slu.edu

Abstract

This study explores the impact of class size and course modality on student learning outcomes, using Infrastructure as Code (IaC) principles to design Information Systems and Cybersecurity graduate courses. To address challenges like larger class sizes, diverse formats, and AI reliance, IaC principles such as scalability, modularity, and repeatability were applied to course design. Tools like LMS blueprints and modular templates ensured consistency across online and in-person formats. Analysis of student feedback and grades showed that smaller classes and individual assignments improved outcomes, while group work fostered collaboration. The findings suggest that IaC-inspired strategies can improve scalability and quality in graduate education.

1 Introduction: The Era of Higher Education Challenges

Saint Louis University's (SLU) School for Professional Studies (SPS) offers online applied education for working professionals both domestically and inter-

nationally[8]. In recent years, SLU has faced rising costs, fewer traditional students, and growing competition. As of 2018, 66% of U.S. adults had less than a four-year degree, and many who re-enroll in college prefer online programs, reflecting the growing shift away from the traditional student model[4][1][7]. The UPCEA[9] argues that higher education has reached "the end of the 'traditional students' as we know them."

In addition, according to the Institute of International Education[5] Open Doors 2024 report[3], U.S. colleges and universities saw a record 1,126,690 international students in 2023-2024, accounting for 6% of the total U.S. higher ed population. In this same period, 56% of international students majored in STEM fields. Among the top 10 states hosting international students, the Midwest has seen significant growth, with Missouri facing a 35% increase from the previous year. Clearly, the competition is staggering.

To meet these challenges, SLU has further expanded into global markets, introducing a global graduate program in 2022, welcoming global and non-traditional students on campus. With these changes, and the continuing requirements to cut costs, faculty have faced the need to adapt to teaching in various modalities and formats, both online and on-ground, with constantly growing class sizes and caps.

Another significant concern faced by faculty is the increasing reliance on AI by student populations, sometimes resulting in academic integrity issues. This creates significant risks for students to absorb knowledge and to grow their professional skills of critical thinking, problem solving and communication.

To respond to the need for flexible course modalities, growing class sizes and students' over-reliance on AI, faculty at SPS have been exploring innovative course designs and assessments to build adaptable and scalable courses that can be replicated in various settings, while maintaining some level of standardization.

2 Infrastructure as Code Principles in Course Design

Some of the technology challenges of businesses are similar to that of higher education: delivering to various types of consumers, fluctuations in user volumes, cost considerations, speed, and flexibility to adjust to these needs. DevOps has evolved as a key IS methodology to these business problems. Considering the similarities, as well as the author's expertise in Information Systems and Cybersecurity disciplines, the principles of Infrastructure as Code (IaC) used in DevOps deployments were considered in the design of graduate courses.

According to Amazon Web Services, Infrastructure as Code is the creation and management of environments using code instead of manual processes, enabling greater efficiency, scalability, consistency and automation while reducing

errors and speeding up deployment. This helps businesses scale, adapt, control costs and innovate faster in dynamic computing environments[2]. According to Morris[6], IaC relies on various important principles, including:

1. Repeatability and reproducibility: any action and process should be repeatable efficiently, without significant human effort.
2. Idempotency and consistency: running the environment multiple times should always produce the same result, assuring consistency.
3. Scalability and automation: easily adjust and scale environments up or down based on changing demands without manual intervention.
4. Modularity and reusability: creation of reusable pieces for common tasks.
5. Version control and continuous deployment: tracking changes and integrating testing into regular deployment for traceability and collaboration.
6. Reliability and accountability: ensuring predictable infrastructure performance and clear change oversight for transparency and trust.

Applying these principles can help organizations save time, reduce costs, and be more responsive to external challenges. This paper demonstrates that these principles of Infrastructure as Code can support course and assignment redesign to meet the current challenges of IS/Cyber graduate programs. The paper will discuss the author's practical application of these principles in course and assignment redesign and the related results in student learning and success for a specific course in the IS program.

3 Requirements for Course Design

3.1 Requirements for Class Modalities

To teach in various modalities, a crucial criteria was to make things repeatable yet flexible, easily replicating the same content in various formats. To move between online and in person lectures, it was important to provide the ability to hide and unhide content. Creating ways to auto-generate online and in-person formats based on predefined templates was crucial. Repeatable content and workflows assured this and provided consistency across formats and modalities.

3.2 Requirements for Class Size

To manage the increased work for faculty with the higher class sizes, it was important to automate assignment grading where possible. Assignments had to be scalable depending on student numbers by working in groups versus individual settings. However, considerations had to be given to the risks of group work, making group-based projects equitable and fair.

3.3 Requirements for the Over-Reliance on GenAI

Assignment formats had to support students to focus on their message, improve critical thinking and writing skills. Assignments needed to be less prone to full reliance on GenAI and more reliable on their own critical thinking. Various assignments for different formats had to be developed to encourage oral assessments that are generally more AI-resistant, despite growing class sizes.

To meet these challenges and needs, the various IaC principles were aligned with the specific challenge, as demonstrated in Figure 1. These principles then drove the design and implementation of new assignments, class activities and assessments to grow student success.

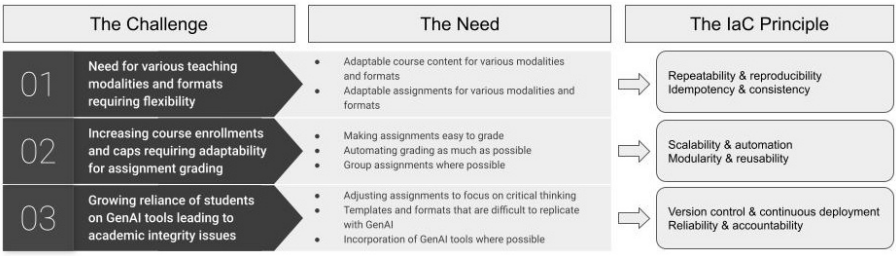


Figure 1: The challenges, needs and related IaC principles for course redesign.

4 Examples of IaC Principles in Course Design

Based on these requirements, various changes were implemented across different modalities and class sizes to assignments, class activities and assessments.

4.1 Repeatability and Reproducibility

Using the IaC principles, the author used Blueprints in the Learning Management System with various components (both assignments and course content) that can be easily hidden or published. As an example, case study based assignments allowed us to change the use case or problem statement, while maintaining standard requirements. Templates for both the course and the assignments helped faculty use relevant components and adjust according to specific class and student needs.

4.2 Idempotency and Consistency

Templates and Blueprints allowed faculty to stay consistent across various sections. Assignment templates also assured students had fairly standard submis-

sions, which made grading faster and easier for faculty, saving them time from grading that could be reallocated to student mentorship where it was needed.

4.3 Scalability and Automation

Assignment templates facilitated easier and faster grading by allowing faculty to recognize where to look for the answer, and limiting the space for students to provide too much information. This also forced students to think critically and grow their writing, both crucial skills in the industry. Incorporating project management tools in group work also helped faculty assess team contributions quickly, providing fair and equitable grades based on student involvement in each assignment. Critical thinking based quizzes were also developed that could enable faculty to auto-grade using the Learning Management System.

4.4 Modularity and Reusability

The author decided to test out mini-modules that can be easily replicated in other sections. This was done through various lecture videos and using the Learning Management System. Modularity also applied to how assignments were created, as depending on class size and student interaction, various modules could be included or excluded in the specific assignments.

4.5 Version Control and Continuous Deployment

Versioning student submissions with project management tools encouraged accountability. Students were required to follow PM best practices and tools to track their work, and provide reports on those. This not only helped them practice common industry tools, but also allowed them to grow their report writing capabilities that will be crucial in a corporate environment.

4.6 Reliability and Accountability

Reliable assessments can detect AI usage while maintaining academic integrity. This is best accomplished through oral assessments, where the immediate interaction between students and faculty allows the cross-checking of understanding. In-person students delivered presentations with strict criteria and time limits, while allowing creativity, mimicking real-life situations of industry presentations. This was a limitation in the online formats, where project management tools were used to assure accountability of team members. The templates used in various assignments forced students to think critically, as they had limited space and their message had to be summarized very thoughtfully to meet requirements. This required students to practice their prompt engineering skills

and to learn how to disclose their AI use and to transparently describe their process for delivering assignments.

5 Course Design Results

To evaluate the success of these principles in the practice of course redesign, class grade averages and course evaluation survey responses from students related to assignments, exams/quizzes and class discussions were analyzed for the same course in the IS graduate program between the period of Fall 2023 and Fall 2024. The results are summarized in Table 1.

Table 1: Results of various class modalities and sizes for the same course in the Information Systems graduate program between Fall 2023 and Fall 2024.

		Course	Course 1	Course 2	Course 3	Course 4	Course 5	Course 6	Course 7	Course 8	Course 9
Course information		Modality	Online	Online	On ground	On ground	On ground	Online	Online	On ground	On ground
		Class size (instruction)	12	12	96	98	92	26	26	99	100
		Section size (grading)	12	12	24	25	22	26	26	50+49	50+50
Assignment information		Assignment types	Individual only	Individual only	Individual at home, group in class	Individual at home, group in class	Individual at home, group in class	Individual only	Individual only	Group work only, individual exams	Group work only, individual exams
Grade averages		Median grade %	97.61	95.63	90.59	92.19	91.67	90	93.86	88.06	86.81
		Median grade	A	A	A-	A-	A-	A-	A-	B+	B
		Average grade %	96.85	93.46	86.82	89.2	87.79	88.35	90.64	87.21	85.84
		Average grade	A	A-	B	B+	B+	B+	A-	B+	B
Survey results	Class discussions/activities supported my learning	Mean	3.50	3.92	3.87	3.97	3.90	3.67	3.71	3.84	3.76
		Median	3.50	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
	Assignments/homework supported my learning	Mean	3.75	3.83	3.82	3.87	3.84	3.83	3.71	3.84	3.80
		Median	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
	Exams/quizzes supported my learning	Mean	3.75	3.83	3.89	3.83	3.94	3.50	3.86	3.89	3.76
		Median	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
	Survey response rates	Survey responses	12	12	45	30	31	6	7	19	25
		Survey response rate for course	100%	100%	46.88%	30.61%	33.70%	23.08%	26.92%	19.19%	25.00%
		Survey response rate for section	100%	100%	Unknown	Unknown	Unknown	23.08%	26.92%	19.19%	25.00%

Course evaluation scores ranged between 0 and 4 as follows: strongly agree (4), agree (3), disagree (2), strongly disagree (1), and not applicable (0). Based on the course data, several trends emerged regarding the impact of assignment types, teaching modalities, and class sizes on student grades and responses.

5.1 Impact of Teaching Modality

Online courses (1, 2, 6, 7) had higher median grades (A to A-) compared to on-ground courses (3 - 5, 8, 9), where median grades ranged from A- to B. Student satisfaction with assignments and exams was comparable across modalities, with most courses reaching scores of 3.80 or above. The exceptions are online courses (1, 7) with assignment/homework feedback, and online courses 1, 5 and

on-ground course 9 with lower score for exams. Online courses had lower mean scores for how discussions supported learning (3.50, 3.67 and 3.71 for courses 1, 6 and 7) compared to on-ground courses, which frequently exceeded 3.90.

5.2 Impact of Class Size

Smaller courses (1, 2, 6, 7) had higher average grades (88.35% - 96.85%) compared to larger classes (3 - 5, 8, 9) with average grades between 85.84% and 89.2%. Survey response rates were significantly higher in smaller classes, especially in courses 1 and 2, where response rates reached 100%, compared to response rates as low as 19.19% in larger sections (8 and 9). It is clear that smaller classes facilitate more personalized interaction, boosting both grades and engagement.

5.3 Impact of Assignment Types

Courses with individual-only assignments (1, 2, 6, 7) had slightly higher grades on average than those incorporating group work (3 - 5, 8, 9). For instance, courses 8 and 9, which used group work exclusively, had median grades of B+ and B, compared to A and A- in individual-only courses. Results showed minimal differences in student perceptions of assignments, regardless of assignment type, with most responses hovering around a mean of 3.80 to 3.87. It's notable, however, that the worst responses regarding assignments (means of 3.71 and 3.75 for courses 1 and 7, respectively) were received for 2 courses that required individual assignments only. This suggests that despite students performing better with individual assignments, they prefer group interactions more.

5.4 Student Feedback

In addition to the quantitative data, the verbal feedback provided by students in the course evaluations also supported some of these findings. As an example, course 2 respondents (online, small, individual assignments) shared that the course would benefit from group work, an offline format, and more discussions:

> "I think this course would be better done if this was conducted in offline because it would be easy to understand and improve skills."
> "This course could have incorporated a valuable group project. Each team could have been assigned a real–world company and tasked with identifying potential cyber threats while formulating effective prevention strategies. The teams would then present their findings collaboratively, fostering a practical understanding of cybersecurity challenges in a corporate setting."
> "[...]As previously suggested, incorporating a group project into

> the curriculum would enrich the learning experience by providing hands–on exposure. Such projects not only cultivate practical skills in preventing cyber threats but also enhance essential abilities in public speaking and team collaboration among students."
> "[...]More opportunities for open discussions and peer interaction would have added to the collaborative learning experience."

This shows that increasing class sizes for online lectures would be to the detriment of students, as discussions managed among more than 25 people will be cumbersome for faculty to engage with. In addition, students recognize the practical benefit of group work and presentations.

For course 3 (on-ground, large, mix of individual and in-class group work), the only feedback somewhat related to the course assignments and activities was as follows:

> "[...]just did a good job in making us understand the current situations by interlinking the recent situations and giving them as examples. The in-class activities and the final week's lab are the learning add-ons for me."

This course was co-located with other instructors' sections for grading, but the group activities in class were managed by the main instructor. This shows the importance of discussions and hands-on activities, which can be hindered by increased class sizes, as students will receive less individual attention.

Feedback for course 4 (on-ground, large, mix of individual and in-class group work) stated that:

> "the class environment and the teaching style of the course is absolutely good. The in class activities and quizzes are more interactive and made us to think the outcomes on our own. And the discussions helped us in improving our logical thinking capabilities."

This demonstrates the satisfaction of students appreciating the variety of activities and interaction with faculty. Despite being a large class, faculty did a good job connecting with students and making the interaction meaningful.

Feedback for course 8 (on-ground, large, group assignments and individual exams) suggested that students enjoyed the presentations and working in teams, helping them grow their professional skills. However, some comments warned about the risks of group work in case of conflict, which can create additional strain on faculty without appropriate resources to assist them in the resolution and tracking of these types of issues.

> "Nothing but the presentation work was good because it helps students to improve more in the speaking skills and knowledge"

"I like this course, this one is beneficial and showcases how we need to protect a corporation in terms of strong cybersecurity principles, and the professor is also so informative and helps the students with their respective doubts and gives valid clarification on it, and it is a good subject for maintaining teamwork. However, I observed some flaws in it, including the teammates are not co operative only one or two persons are actively participate in teamwork but it will impact for all teammates. We can't push everyone right so I think working in a team is not a good option, it will impact all."
"[...] The midterm assessment process was also excellent, as it provided meaningful feedback to track our progress. [...]"

6 Conclusions, Limitations

Students clearly enjoy group activities and related assignments with project management activities and presentations, though some risks were presented related to group conflicts and inequitable contributions of team members. Traditional solutions such as requiring students to describe contributions of team members are not efficient in large class sizes, as that requires faculty to individually review student submissions. Instead, using project tracking tools worked as a good compromise that helped students showcase their contributions.

Smaller classes yielded higher grades and engagement, suggesting that scalability and adaptability of teaching approaches are critical as enrollments grow. Individual-focused assignments may support higher academic performance, while group-based approaches could introduce challenges in larger classes. Mitigating these will require institutions to provide adequate resources to faculty, including TAs to help grade and manage situations in conflict in large classes.

To maintain consistency across formats and sizes, it is beneficial to develop modular and scalable teaching strategies that emphasize clarity, fairness, and student-centered learning approaches. These findings highlight the importance of flexibility in assignment design and teaching methods to address diverse student needs in various learning environments.

Certain limitations of this analysis exist. Student grades may not necessarily reflect actual student learning. However, student responses to surveys are also not consistent, ranging between 19% to 100%. Courses with lower survey response rates lack true representation of how successfully the various exams, assignments and discussions supported student learning. In addition, due to the cross-listing of multiple sections, some course surveys applied to multiple instructors. Due to the confidential nature of the surveys, it wasn't always evident which sections the student responses came from. Even though the majority of assignments were the same due to the combined in-person compo-

nent, some slight variations could still exist based on the section's responsible faculty. Overall, however, due to the similarity of the main artifacts across sections, this data provides valuable insights into how successfully the changes in this course contributed to student success across class sizes and course modalities. Creating a more targeted survey for student input on the effectiveness of assignments, quizzes, exams and activities could provide a more robust approach to measuring how well these IaC principles facilitated course redesign to meet the needs of various class sizes and modalities while assuring student success and learning.

References

[1] National Student Clearinghouse (NSC). *Some college, no credential student outcomes: 2024 report for the nation and the states.* https://nscresearchcenter.org/some-college-no-credential/.

[2] AWS. *What is Infrastructure as Code?* https://aws.amazon.com/what-is/iac/.

[3] Open Doors. *International students.* https://opendoorsdata.org/data/international-students/.

[4] Elise Gould. *Two-thirds of adults have less than a four-year degree.* https://www.epi.org/publication/two-thirds-of-adults-have-less-than-a-four-year-degree-policymakers-should-work-to-make-college-more-attainable-for-them-but-also-strengthen-labor-protections-that-help-all-workers/.

[5] Institute of International Education (IIE). *United States hosts more than 1.1 million international students at higher education institutions, reaching all-time high.* https://www.iie.org/news/us-hosts-more-than-1-1-million-intl-students-at-higher-education-institutions-all-time-high.

[6] Kief Morris. *Infrastructure as Code.* O'Reilly Media, 2016.

[7] Annamaria Szakonyi et al. "Non-traditional education to advance women in computing careers in the St. Louis metro region". In: *Proceedings of the 2021 IEEE Global Engineering Education Conference.* EDUCON2021. Vienna, Austria: IEEE, 2021, pp. 1093–1097. DOI: 10.1109/EDUCON46332.2021.9454075. URL: https://ieeexplore.ieee.org/document/9454075.

[8] Saint Louis University. *About Professional Studies.* https://www.slu.edu/professional-studies/about/index.php.

[9] UPCEA. *The Pulse of Higher Ed: The End of the "Traditional Students" as We Know Them.* https://upcea.edu/the-end-of-the-traditional-students-as-we-know-them/.

Remote Protocol Analysis Lab *

Nifty Assignment

Michael Ham
Beacom College of Computer and Cyber Sciences
Dakota State University
Madison, SD 57042
michael.ham@dsu.edu

Protocol analysis is a critical skill for cybersecurity engineers, especially as interconnected wireless devices expand the threat landscape. Non-routable communication protocols, such as Infrared, Zigbee, and Bluetooth, play a vital role in Internet of Things (IoT) applications and require specialized knowledge for effective threat hunting and vulnerability assessment. Academic standards, such as the NSA-CAE designations, emphasize the importance of networking and communication protocol analysis as essential knowledge units for cybersecurity education.

This nifty assignment requires remote learners to use a logic analyzer to analyze a custom-built IoT device and develop a software extension to decode the device's transmitted signals. The target device is a light-up tree made from a foam cone decorated with an RGB LED strip. The LEDs are typically controlled by an infrared (IR) remote using the NEC protocol. However, the LED strip IR receiver is attached directly to an Arduino microcontroller in this lab. The Arduino enables the programmatic injection of raw NEC IR commands that would otherwise be transmitted wirelessly. Direct command injection allows for unique functionality not seen in the LED strip, such as flashing "DSU" in Morse code with different colors representing each letter in the sequence. The device's programmable nature makes it a unique tool for students to explore beyond "standard" wireless communication protocols.

Applied labs like this provide students with experiential opportunities to practice capturing and analyzing real-time signals from target IOT devices; these learned skills translate directly to professional skill sets. Delivering such a lab on campus is more accessible than conducting the same lab for online students. Compared to campus students, remote learners face hurdles in accessing

similar applied lab experiences due to physical, regulatory, and technical barriers. Hardware costs, geographic limitations, and the need for accurate timing data in signal analysis are important considerations for remote lab participation.

A remotely accessible web-based platform is crucial for providing this and similar labs. A remote platform can connect theoretical learning with practical application by incorporating tools such as logic analyzers (e.g., Saleae Logic 8) with a target hardware device, emulating a physical presence. This environment ensures meaningful educational experiences in protocol analysis for every student, irrespective of the learning modality.

The web platform developed for this online lab is a Docker-based solution that provides students access to the hardware and tools needed for the assignment. It consists of three containers (nginx, logic2, and rpa-app). These three containers work in concert to provide a web presence through Nginx and a custom-built Flask application (rpa-app). This application allows students to control the target hardware device and work directly with a logic analyzer via the Logic2 software as if they installed it locally.

Upon logging into the web application, students interact with four central components on an easy-to-navigate page. First, the Logic2 software, which is attached directly to a logic analyzer, is embedded via NoVNC. Second, students have access to a live video stream of the target device, enabling them to observe their interventions as if they were locally present. This live view helps learners bridge the gap between virtual and physical interaction. Third, programmable buttons send raw serial commands to the Arduino, allowing students to perform certain tasks, such as resetting the device or transmitting the NEC IR sequence to light up the tree. The buttons are extensible, supporting various target devices and functionalities. Lastly, the platform has a web form for students to upload and test their Logic2 software extensions. This setup grants students complete control over hardware transmissions, live analysis, and immediate feedback from the device.

The Docker-based architecture makes this assignment scalable, easy to deploy, and accessible to other universities. Using Docker minimizes hardware requirements via horizontal scaling while maintaining the consistency of a hands-on learning environment, enabling institutions to replicate the experience for their students. The modular design makes adapting to other target devices or communication protocols easy, broadening its educational impact.

This assignment helps students progress toward learning outcomes, including analyzing raw communication signals and developing software extensions to decode protocols, creating functional software extensions using an API to translate theoretical knowledge into actionable tools, and demonstrating their understanding of how digital communication protocols control devices.

Dynamic Bracketology with C++ AI for NCAA March Madness*

Nifty Assignment

Roy Manfredi
Computer and Digital Technology
Westminster College
Fulton, MO 65251
roy.manfredi@westminster-mo.edu

Teaching C++ in CSA404: Data Structures, I often encounter students losing interest in repetitive coding exercises. To address this, I created an engaging midterm project in early 2024 that combined real-world problem solving with their enthusiasm for sports: developing an AI model to predict NCAA March Madness brackets. This spontaneous change from the planned curriculum captivated students while fulfilling course objectives.

The project began with research into college basketball teams, requiring students to analyze data and create C++ structs that represent players and teams. Data entry was divided among the six students, each researching a specific tournament region. Once data were compiled, students collaborated in teams to develop functions to calculate player scores, aggregate team scores, and dynamically visualize the bracket.

The assignment addressed key course goals by leveraging essential data structures such as structs and arrays for player and team data storage. The students created algorithms to process player statistics and dynamically calculate team scores. This required the implementation of modular functions and dynamic programming techniques, highlighting scalability and reusability. Performance metrics included the efficiency of score computations, adaptability of the bracket generator to changing data inputs, and accuracy of predictions in public bracket leagues.

After completing the core functionality, each student personalized their AI to reflect individual scoring priorities, incorporating unique algorithms to predict winners. The project culminated in submitting AI-generated brackets to a public league, where several performed exceptionally well in national

pools.There were three performance metrics evaluating technical efficiency and practical outcomes.

1: Algorithm accuracy:

Metric: How well the program's predictions aligned with real-world tournament outcomes. Relevance: While not expected to produce perfect brackets, the AI algorithms were evaluated by their ability to predict winners correctly, with results compared to actual tournament outcomes or rankings in public bracket leagues (e.g., ESPN).

2. Impact of Algorithm Customization:

Metric: Variability in predictions based on individual students' scoring criteria. Relevance: Students' unique weighting of factors (e.g., offensive vs. defensive strengths, player experience) influenced how their AI performed, showcasing the importance of thoughtful algorithm design.

3. Reusability and modularity of the code:

Metric: How easily functions and data structures could be reused or adapted for new scenarios. Relevance: The modular design allowed for future applications beyond basketball, such as applying similar logic to other tournaments (even ones like a favorite food tournament) or real-world prediction models. This assignment achieved 100 percent student buy-in and sparked continued interest in computational problem-solving. While this specific content resonated with the class demographics, the adaptable structure can be applied to other domains, making it a versatile teaching tool. Students continue to discuss and inquire about this project, demonstrating its long-term impact on engagement and learning.

Internship Experience Sharing *

Nifty Assignment

Bin Peng and Wen-Jung Hsin
Computer Science and Information Systems
Park University
Parkville, MO 64152
{bpeng, wen.hsin}@park.edu

Computer Science (CS) and Information Systems (IS) internships at the college or university level offer students opportunities to gain hands-on experience and apply their academic knowledge in a professional setting. These internships provide skill development, mentorship, industry insights, and enhanced job prospects, among many other benefits. In the CS/IS program at Park University, the internship is strongly recommended. Students can earn academic credits for completing an internship by following specific guidelines for converting the internship into course credits at Park University. These guidelines include: (1) the request must be approved before the internship begins, (2) the student must work in a professional environment, (3) the internship duties must be sufficiently complex to require the expertise of a senior-level CS/IS student, and (4) prior to enrolling in the internship course, the student and their job supervisor must jointly prepare an internship proposal, which must be agreed upon and supervised by an academic advisor.

An internship is regarded as a vital component of a student's academic experience in the CS/IS program at Park University. As a result, students are strongly encouraged to complete a survey after finishing their internship to share their experiences with fellow students in the CS/IS program. The survey is shown in Table 1: CS/IS Survey – Internship Experience Sharing.

Each year, these shared experiences are disseminated to all CS/IS students through various communication channels, such as emails, the departmental LinkedIn group, poster displays at the departmental bulletin boards, and presentations at the annual Student Research and Creative Arts Symposium at Park University to showcase students' achievements. Sharing the results of these experiences not only provides valuable insights into how peers navigate and handle internships, but also fosters discussions and idea-sharing among

students, ultimately enhancing the likelihood of supporting one another in securing and successfully completing internships. Encouraging students to share their experiences has led to a significant increase in recognition and participation among students in the CS/IS program at Park University.

Table 1: CS/IS Survey– Internship Experience Sharing

The Computer Science and Information Systems program invites you to share your internship experiences. Your insight and reflection will be compiled and shared with all CS/IS students to promote greater participation in internships.	
Demographic Information	(1) What is your name?
	(2) What is your current class standing? (i.e., junior, senior, etc.)
	(3) Are you an international Student?* (Yes/No)
	(4) How can the CS/IS program share your submission? (i.e., anonymously, both name and email, name but not email, others.)
	(5) What is your email address?
Internship Experience	(1) When did you start looking for internship opportunities? (We're interested in knowing when during your studies, such as midway through your sophomore year, although if it was more than a couple of years ago, feel free to mention the specific year.)
	(2) Describe your experience securing the internship position, including a) how you learned about this opportunity (e.g. through Park University's career services, personal connections, internet searches, etc.), b) the duration it took to receive an offer, and c) whether any prior experience played a role.
	(3) Briefly describe the company and its location. This can be specific such as "Facebook, Menlo Park, CA" or more general, like "an insurance company in the Kansas City area".
	(4) Describe your duties during the internship.
	(5) Describe what you learned from this experience. Feel free to mention technical skills, soft skills, career directions, career insights, or anything else you found meaningful.
	(6) Provide your advice to students on how to secure an internship.

* International students are required to complete additional paperwork for internships, therefore, the CS/IS program would like to recognize students who have successfully navigated this process.

Teaching Cellular Concepts: An Into With GSM

Nifty Assignment

Kyle Cronin, Michael Ham
Beacom College of Computer & Cyber Science
Dakota State University
Madison, SD 57042
kyle.cronin@dsu.edu, mike.ham@dsu.edu

Many excellent labs and curriculum exist for teaching TCP/IP concepts, ethernet, routing and switching, and 802.11. However, today one of the prime mechanisms for communication do not exist within traditional networked environments: enter the mobile phone. While our students need not be experts at cellular communication, a basic understanding of device and base station signaling is important when entering the cybersecurity industry. In this nifty assignment, students stand up a simple OpenBTS base station using a Raspberry Pi and a Software Defined Radio.

To get started, students require basic hardware: a Raspberry Pi, an USRP B200 software defined radio, and a set of Android phones and SIM cards. Blank SIM cards can be procured and programmed, however Android OS and the proper OpenBTS configuration will allow for SIMs from any carrier to work.

With the hardware online, students can explore the configuration options that exist within modern cellular infrastructure. It should be noted that OpenBTS, an older project, only supports GSM (2G communication), however the signaling characteristics of modern networks (LTE and 5G) carry over, while GSM can be easily operated off of commodity computing hardware with low processing power.

This lab environment exposes students to the configuration settings that are required for mobile networks to operate. In order to bring a base station online, students must demonstrate a knowledge of the configurations required: Absolute Radio Frequency Channel Numbers, Mobile Country Codes, Mobile Network Codes, and International Mobile Subscriber Identity registration. In order to complete the task, students must apply all appropriate values, connect a set of phones, and be able to communicate within the devices.

The activity exposes students to the necessary parameters, but also allows for exploration beyond. OpenBTS supports capturing all cellular signaling information in a PCAP format, which allows students to easily analyze the changes in their configuration options and how they may be observed in an offline modality. As is often done in TCP/IP exercise, students can prove configuration options are present and show how they impact the operation of the network.

Outside of basic configuration, several security challenges can be introduced and demonstrated within a network environment. With a necessary discussion of the legal state of operating test cellular networks, students may observe the connection process as well as what may interrupt or cause issues with the process of a device registration. Common topics such as network spoofing can easily be demonstrated between students, leading to a discussion as to how these attacks can be somewhat mitigated in modern cellular network environments.

In totality, operating a GSM network from scratch allows students a hands-on opportunity to explore a modality for communication that is often challenging or unavailable in a learning environment. Students can demonstrate the concepts, as well as compare them to modern networks, while not having a heavy computing environment. Additionally, this lab can be positioned to give students without a traditional electrical engineering background exposure to cellular environments in a hands on way.

Prompting Collaboration: Development of an Multidisciplinary Applied AI Minor Program*

Panel Discussion

Ajay Bandi[1], Benjamin Blackford[2], Aziz Fellah[1], Diana Linville[1], Trevor C. Meyer[3], Robert J. Voss[4]
[1]School of Computer Science and Information Systems
[2]School of Business
[3]Department of Language, Literature and Writing
[4]Department of Humanities and Social Sciences
Northwest Missouri State University, MO 64468
{ajay, blkfrd, afellah, dianar, tmeyer, robvoss}@wmissouri.edu

1 Summary

Artificial Intelligence (AI) has rapidly transformed industries and research, becoming a driving force for technological innovation and development [1]. As AI continues to grow and change, it is reshaping the way we approach problem-solving, decision-making, and creative processes across various sectors. Northwest Missouri State University is developing a new multidisciplinary AI minor open to all undergraduate students on campus. The program is tailored for students from any discipline who want to explore how AI can be utilized and integrated into their fields such as computer science, humanities, business, sciences, healthcare, agriculture, and education, among others. The curriculum integrates topics such as foundational AI concepts, prompt engineering and writing processes, ethical considerations in AI, AI in the workplace, and a capstone project. This program also promotes interdisciplinary collaboration and emphasizes the ethical use of AI.

By the end of the program, students will be able to use AI to enhance efficiency and accuracy in tasks, develop and evaluate effective prompts, apply

generative AI tools across various input formats, and assess the ethical considerations of AI in real-world applications. The panel members are experts from diverse fields, including management, humanities, technical writing, and computer science. The panel discusses the development of the AI minor curriculum and explores opportunities to extend the AI curriculum by offering AI certificates for undergraduate and graduate online professional students. By attending this panel, the audience will gain valuable insights into developing comprehensive AI programs, fostering cross-disciplinary innovation, and preparing students to use AI ethically and effectively across diverse fields.

2 Minor Courses

The Applied AI minor requires 24 credit hours. Of these 24 hours, students must complete 15 required hours and 9 elective hours. The descriptions of the required courses are provided in this section. The electives are flexible, and students can take them within their major or outside of the major. This multidisciplinary minor program will begin in fall 2025 at Northwest Missouri State University.

Introduction to Applied AI: A non-technical introduction to the fundamental concepts in Generative AI focusing on practical applications, societal impacts, and ethical considerations.

AI in the Workplace: This course provides an overview of how AI is currently impacting the workplace. Topics will include the workflow changes, social impacts, and training needs associated with the rapid advancements in AI. Ethical issues will also be addressed. Students will learn how to effectively apply AI to appropriate activities in the workplace.

Prompt Engineering: Bridging writing students and artificial intelligence, this course explores the art and science of prompt engineering. Students will learn to develop, evaluate, and refine prompts to effectively communicate with and guide generative artificial intelligence (Gen-AI) models to generate outputs tailored to specific audiences, purposes, and contexts. Building upon foundational AI concepts, students will learn key prompt engineering techniques, including zero-shot, few-shot, chain-of-thought, tree-of-thought, and the NOVA system, among others, and creatively repurposing those techniques to develop their own methods and methodologies. Through iterative practice and real-world scenarios, the course equips students with strategies to maximize the value of AI in creative, technical, and executive applications, while developing critical thinking, problem-solving, and communication skills crucial for leveraging Gen-AI.

Ethics and AI: This course introduces students to the intersection of digital humanities and artificial intelligence, focusing on ethical considerations, societal impacts, and the evolving role of AI in the study of human culture and history. Students will explore how AI tools are reshaping digital research, data visualization, and archival practices in the humanities. Special attention will be given to issues of bias in digital methods, the ethics of AI-generated content, and the implications of automation in historical interpretation. Through hands-on projects, students will critically assess the responsible use of AI in the humanities and develop ethical frameworks for engaging with digital tools in research and public-facing projects.

Capstone in AI: This capstone project course offers students from diverse disciplines the opportunity to explore and apply AI techniques and tools specific to their field of study. Through hands-on projects, students will use AI-powered tools to solve real-world problems while collaborating within multidisciplinary teams to develop practical solutions. The course emphasizes the integration of AI across various domains, equipping students with the experience and skills necessary to apply AI effectively to real-world challenges.

3 Biographies

Dr. Ajay Bandi is an associate professor of computer science and the graduate program coordinator for the MS in Applied Computer Science. He teaches a wide variety of both on-campus and online graduate courses in computer science and data science. He has integrated Generative AI into his teaching and research, publishing refereed papers in this area.

Dr. Ben Blackford is the Director of the Melvin D. & Valorie G. Booth School of Business at Northwest Missouri State University. He has a PhD from the University of Nebraska - Lincoln with an emphasis on Strategy. His research interests include humor in advertising and interpersonal relationships in the workplace. Prior to academia, he spent 25 years with Wal-Mart and also co-founded a management consulting and financial advising company.

Dr. Aziz Fellah is an Associate Professor in Computer Science with extensive experience at various institutions. His background spans several subfields of computer science across the discipline.

Ms. Diana Linville is a Senior Instructor and Assistant Director of the School of Computer Science and Information Systems at Northwest Missouri State University and is the program coordinator for the interdisciplinary degree, Digital Media: Computer Science, as well as co-teaching an interdisciplinary course, Knacktive.

Dr. Trevor C. Meyer in an Assistant Professor in the Department of Language, Literature, and Writing. He has taught courses in rhetoric (theory and

history), professional & technical writing, and composition. Generative AI has been a major feature of his teaching and scholarship for the past several years. **Dr. Robert Voss** is an Associate Professor of History, Secondary Social Science Coordinator, and MSEd Teaching History Coordinator specializing in the intersections of technology, history, and society. With expertise in digital humanities and AI ethics, he brings a critical perspective on how emerging technologies shape research, education, and our understanding of the past.

References

[1] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8):260, 2023.

An Instructor's Introduction to Codespaces & Development Containers

Conference Tutorial

Bill Siever[1], *Michael P. Rogers*[2]
[1]*Computer Science & Engineering*
Washington University in St. Louis
bsiever@gmail.com
[2]*Department of Computer Science*
University of Wisconsin Oshkosh
mprogers@mac.com

A containerized environment (e.g., Docker) provides a way to bundle a piece of software with all its requirements and dependencies into a single, easily reproducible and moderately portable "container". *Development containers* (dev containers) leverage containers to provide all the tools and libraries necessary for a specific *software development target*. In addition, dev containers specify other elements relevant to development, such as settings and extensions for Integrated Development Environments (IDEs) [1].

For example, a dev container could be a combination of: 1) a container that has specific compiler/interpreter and library versions for a Java or Python project, and 2) IDE extensions and settings to develop and debug the application. Unlike many other approaches to managing tools and libraries, the container typically includes appropriate versions of all dependencies in a single bundle that is isolated from other installations, thus avoiding conflicts.

Many of the biggest benefits dev containers bring to application development are equally valuable in educational environments, including:

1. A complete and consistent development environment with appropriate versions of all dependencies for a project can be launched in a couple of minutes. This can eliminate both student off-task time spent installing and maintaining software and confusion from inconsistent versions of tools.

2. Cloud-based environments can be used to mitigate the limitations or inconsistencies of local resources (e.g., student laptops with insufficient space or the limited capacity of IT departments to maintain lab software).

3. Support for collaboration and code review are readily available.

4. It is possible to limit integration of facilities that could undermine intellectual property and academic integrity, such as AI-assisted development tools.

GitHub currently provides free educational use of their cloud-based dev container environment, Codespaces [2]. A Codespace is a dev container in a web-hosted version of Visual Studio Code [3], an editor widely used for development. Several courses, like Harvard's CS50 [4], are leveraging Codespaces to ensure a broad range of students in large-scale computing courses are all working in the same environment. Codespaces can virtually eliminate student and faculty time setting up and debugging issues that arise from the myriad of computing platforms that students use for coursework.

Tutorial

This tutorial will start with an overview of the fundamental concepts behind dev containers and provide some examples of their flexibility and breadth of use, including a dev container developed by one of the authors for a course in Digital Logic and another used to maintain course websites. We will proceed to work through two hands-on examples (along with any/all participants): 1) Launching and exploring a preconfigured dev container that may be suitable for many introductory courses and 2) Customizing a dev container with modifications that are relevant to computing educators. We will conclude with a group discussion about potential uses of and experiences with dev containers.

Biographies

Bill Siever is a Teaching Professor of Computer Science and Engineering at Washington University in St. Louis, where he teaches courses in Computer Science and Computer Engineering. Recently, he has updated an introductory course in Digital Logic and Computer Design to leverage dev containers.

Michael P. Rogers is an assistant professor at the University of Wisconsin Oshkosh, where he teaches a variety of computer science courses, including Software Engineering and Mobile App Development, where CodeSpaces and dev containers are a frequent topic of conversation.

References

[1] Development containers. `https://containers.dev/`.

[2] Github codespaces. `https://github.com/features/codespaces`.

[3] Visual studio code. `https://code.visualstudio.com/`.

[4] David J. Malan, Jonathan Carter, Rongxin Liu, and Carter Zenke. Providing students with standardized, cloud-based programming environments at term's start (for free). In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2023, page 1183, New York, NY, USA, 2023. Association for Computing Machinery.

Fending off Gitastrophe: a Tutorial on Architecting Collaborative Projects and Giving Great Feedback

Conference Tutorial

Michael P. Rogers[1], *Bill Siever*[2]
[1]*Department of Computer Science*
University of Wisconsin Oshkosh
mprogers@mac.com
[2]*Computer Science & Engineering*
Washington University in St. Louis
bsiever@gmail.com

1 Git and GitHub

Employers expect that students graduating from a 4-year CS program will have familiarity with version control, and the ACM curricular guidelines [1] reflect that. Of all the version control systems in existence, Git is by far the most popular [2].

Git can benefit individual developers: it allows them to safely test out new ideas, and the ability to rollback changes if necessary. Its true strength, however, lies in collaboration. Participants will learn the fundamentals of how to use Git on a collaborative project via GitHub, a leading cloud-based Git platform. Beyond housing repositories, GitHub also has powerful tools for collaboration that will be demonstrated in this tutorial. Most GitHub features are free for students, including the ability to create an unlimited number of private repos, with an unlimited number of collaborators.

2 Tutorial

The tutorial will begin with an overview of Git and GitHub concepts and terminology (repositories, branches, pushing, fetching, pulling, pull requests and code reviews). It will be followed by a demonstration of a collaborative

workflow that works well when working with student teams: forking and cloning a repo, making pull requests, evaluating pull requests, and providing feedback.

Time permitting, participants will then walk through the same process: fork a repo, make changes to it, and then submit a pull request. Merge conflicts will also be covered.

The tutorial will wrap up with a discussion of other strategies for working Git and GitHub into the classroom, and questions.

Biography

Michael P. Rogers is an assistant professor at the University of Wisconsin Oshkosh, where he teaches a variety of computer science courses, including Software Engineering and Mobile App Development, where Git and GitHub are used extensively.

Bill Siever is a Teaching Professor of Computer Science and Engineering at Washington University in St. Louis where he teaches courses in Computer Science and Computer Engineering.

References

[1] Amruth N. Kumar and Rajendra K. Raj. Computer science curricula 2023 (cs2023): The final report. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2024, page 1867–1868, New York, NY, USA, 2024. Association for Computing Machinery.

[2] Stack Overflow. Stack overflow developer survey 2022: Technology - version control, 2022. Accessed: 2025-01-15.