The Journal of Computing Sciences in Colleges

Papers of the 33rd Annual CCSC Rocky Mountain Conference

October 18th-19th, 2024
Northern Arizona University
Flagstaff, AZ

Abbas Attarwala, Editor
California State University, Chico

Pam Smallwood, Regional Editor
Regis University

Volume 40, Number 2 **October 2024**

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), kworks@furman.edu, Department of Computer Science, Furman University, Greenville, SC 29613.

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

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
Code Grade
GitHub

Welcome to the 2024 CCSC Rocky Mountain Conference

Welcome to the 2024 CCSC Rocky Mountain Conference. Welcome to the 33rd annual conference of the Rocky Mountain (RM) Region of the Consortium for Computing Sciences in Colleges. The CCSC RM region board members are grateful for the authors, presenters, speakers, attendees, and students participating in this year's conference.

This year we received 14 paper submissions on a variety of topics, of which 9 papers were accepted for presentation in the conference. Multiple reviewers, using a double-blind paper review process, reviewed all submitted papers for the conference. The review process resulted in an acceptance rate of 64%. In addition to the paper presentations, there are four peer-reviewed tutorials/-workshops and two posters. We truly appreciate the time and effort put forth into the reviewing process by all the reviewers. Without their dedicated effort, none of this would be possible. A special thank you goes to the Submission chairs Dr. Karina Assiter and Dr. Mohamed Lotfy.

The CCSC RM region board would like to thank our national gold-level partners CodeGrade, ACM CCECC & ACM2Y, GitHub, Rephactor, and the Association for Computing Machinery in cooperation with SIGCSE.

We hope you enjoy the conference and take the opportunity to interact with your colleagues and leave both enthused and motivated. As you plan your scholarly work for the coming year, we invite you to submit a paper, poster, workshop, tutorial, or panel for a future CCSC RM region conference, or to serve as a reviewer or on the CCSC RM region board. Please encourage your colleagues and students to participate in future CCSC RM region conferences.

Thyago Mota, PhD
Metropolitan State University of Denver
Conference Chair

2024 CCSC Rocky Mountain Conference Committee

2024 CCSC Rocky Mountain Regional Board

Mohamed Lotfy, Board Representative .... Utah Valley University, UT
Ed Lindoo, Treasurer ....Regis University, CO
Pam Smallwood, Editor .... Regis University, CO
Dan McDonald, Webmaster .... Utah Valley University, UT

2024 CCSC Rocky Mountain Conference Steering Committee

Mohamed Lotfy, CCSC Region Representative ...Utah Valley University, UT
Ed Lindoo, Treasurer ....Regis University, CO
Pam Smallwood, Editor .... Regis University, CO
Karina Assiter, Submission Co-chair ....Landmark College, VT
Mohamed Lotfy, Submission Co-chair .... Utah Valley University, UT
Dan McDonald, Webmaster .... Utah Valley University, UT
Jenny Nehring, Publicity Chair ....Utah Valley University, UT
Ed Lindoo, Registrar ....Regis University, CO
Mohamed Lotfy, Conference Co-chair ....Utah Valley University, UT
Thyago Mota, Conference Co-chair ....Metropolitan State University, CO
Michael Leverington, Site Chair ....Northern Arizona University, AZ
Mohamed Lotfy, Program Chair ....Utah Valley University, UT
Michael Leverington, Student Posters Co-chair .Northern Arizona University, AZ
Ranjidha Rajan, Student Posters Co-chair Metropolitan State University, CO
Dave Loper, Student Programming Competition Chair ....Utah Valley University, UT

Transforming Agriculture with Artificial Intelligence: Implementations and Technologies*

Poster Session

Venkata Abhiram Nelluri and Hanieh Shabanian
Department of Computer Science & IT
Western New England University
Springfield, MA 01119
{venkataabhiram.nelluri, hanieh.shabanian}@wne.edu

Artificial Intelligence (AI) is fundamentally transforming agriculture by optimizing productivity and addressing critical challenges. This study offers a comprehensive examination of AI's applications and technological advancements in agriculture. Key topics include precision agriculture, supply chain optimization, weed and pest management, crop monitoring, predictive analysis, and emerging innovations like vertical farming and fruit-picking machines. AI-driven precision agriculture utilizes data analytics to refine the application of inputs such as herbicides, fertilizers, and water, enhancing both efficiency and environmental sustainability[4]. Using advanced algorithms and data from sensors and satellites, AI facilitates precise crop monitoring and management, identifying pests, diseases, and weeds at early stages to mitigate crop losses[6]. Predictive analytics leverage AI to forecast weather patterns, crop yields, and commodity prices, empowering farmers with actionable insights for better decision-making[3]. The study further explores vertical farming, an innovative method using AI to optimize controlled environments for crop growth, and fruit-picking machines that automate harvesting, improving efficiency and addressing labor shortages[1]. Additionally, plant stress recognition technologies use AI to detect early signs of stress and disease, thereby enhancing crop resilience[5]. However, the adoption of AI in agriculture faces challenges, including high initial investment costs, increased energy consumption, and limitations in crop variety and adaptability[7]. High costs of infrastructure and energy, particularly in vertical farming, pose significant barriers.

Moreover, current AI systems are often optimized for specific crops, with limited applicability to staple crops and deep-rooted vegetables. The complexity of agricultural ecosystems and the need for extensive, high-quality data further complicate the development of robust, generalized AI models[2]. Addressing these challenges requires on-going research, and collaboration among researchers, policymakers, and industry stakeholders. By overcoming these barriers, AI has the potential to significantly enhance agricultural productivity, and food security, paving the way for more resilient and efficient farming practices.

References

[1] Lisa M Dischinger, Miranda Cravetz, Jacob Dawes, Callen Votzke, Chelse VanAtter, Matthew L Johnston, Cindy M Grimm, and Joseph R Davidson. Towards intelligent fruit picking with in-hand sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3285–3291. IEEE, 2021.

[2] James W Jones, John M Antle, Bruno Basso, Kenneth J Boote, Richard T Conant, Ian Foster, H Charles J Godfray, Mario Herrero, Richard E Howitt, Sander Janssen, et al. Brief history of agricultural systems modeling. *Agricultural systems*, 155:240–254, 2017.

[3] Asit Kumar Pradhan, Dilip Kumar, Himanshushekhar Chaurasia, Sneha Murmu, Ipsita Samal, Tanmaya Kumar Bhoi, GAK Kumar, and Biswajit Mondal. Crop monitoring system integrating with internet of things and artificial intelligence. In *Artificial Intelligence and Smart Agriculture: Technology and Applications*, pages 193–208. Springer, 2024.

[4] Maria Teresa Linaza, Jorge Posada, Jürgen Bund, Peter Eisert, Marco Quartulli, Jürgen Döllner, Alain Pagani, Igor G. Olaizola, Andre Barriguinha, Theocharis Moysiadis, et al. Data-driven artificial intelligence applications for sustainable precision agriculture. *Agronomy*, 11(6):1227, 2021.

[5] Amanda Kim Rico-Chávez, Jesus Alejandro Franco, Arturo Alfonso Fernandez-Jaramillo, Luis Miguel Contreras-Medina, Ramón Gerardo Guevara-González, and Quetzalcoatl Hernandez-Escobedo. Machine learning for plant stress modeling: A perspective towards hormesis management. *Plants*, 11(7):970, 2022.

[6] Dineesha Soni, Priya Patel, and Manan Shah. Artificial intelligence in crop monitoring. In *Agricultural Biotechnology*, pages 247–257. CRC Press, 2022.

[7] Grace Ning Yuan, Gian Powell B Marquez, Haoran Deng, Anastasiia Iu, Melisa Fabella, Reginald B Salonga, Fitrio Ashardiono, and Joyce A Cartagena. A review on urban agriculture: technology, socio-economy, and policy. *Heliyon*, 8(11), 2022.

A Learning Module for Shortest Path Algorithms Using Map-Based Data and Algorithm Visualizations*

Poster Session

James D. Teresco
Siena College
Loudonville, NY 12211
jteresco@siena.edu

A learning module to help teach shortest path and spanning tree algorithms is presented, focusing on Dijkstra's algorithm and Prim's algorithm. The module consists of a series of questions and tasks for students to work through in a class or lab session, or as part of a homework assignment. It uses data representing real highway systems and interactive map-based algorithm visualizations (AVs) from the Map-Based Educational Tools for Algorithm Learning (METAL) project [1]. Students interact with these algorithms in progress on real data to gain an understanding of how they work, and then use a provided implementation to experiment on larger data sets, visualizing the results. They are guided through a series of exercises that show the close relationship among these two algorithms and the graph traversal algorithms they had previously studied. In particular, they see that simply by changing the underlying data structure, the same implementation can compute a spanning tree based on single-source shortest paths (Dijkstra's algorithm, with a priority queue based on cumulative distances), a minimum cost spanning tree (Prim's algorithm, with a priority queue based on individual edge lengths), or spanning trees and paths based on breath-first (with a queue), depth-first (with a stack), or random-first (with a structure that removes a randomly-chosen element next) traversals. The module is one of several that have been developed, allowing instructors to bring METAL's map-based data and engaging, interactive AVs into their classrooms with a low barrier for entry. There is nothing to download or install to use METAL's data or AVs, and the learning modules themselves along with supporting materials are available from the author. The classroom-tested modules have been developed in shared documents, and can be used as-is or modified as needed.

References

[1] James D. Teresco, Razieh Fathi, Lukasz Ziarek, MariaRose Bamundo, Arjol Pengu, and Clarice F. Tarbay. Map-based algorithm visualization with metal highway data. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 550–555, New York, NY, USA, 2018. ACM. URL: `http://doi.acm.org/10.1145/3159450.3159583`, `doi:10.1145/3159450.3159583`.

Using a Distinctive Curricular Design Process for Liberal Arts Computing Programs*

Conference Tutorial

Jakob E. Barnard[1], Grant Braught[2], Janet Davis[3],
Amanda Holland-Minkley[4], David Reed[5], Karl Schmitt[6],
Andrea Tartaro[7], James Teresco[8]

[1]University of Jamestown, Jamestown, ND 58405
Jakob.Barnard@uj.edu

[2]Dickinson College, Carlisle, PA 17013
braught@dickinson.edu

[3]Whitman College, Walla Walla, WA 99362
davisj@whitman.edu

[4]Washington & Jefferson College, Washington, PA 15317
ahollandminkley@washjeff.edu

[5]Creighton University, Omaha, NE 68178
DaveReed@creighton.edu

[6]Trinity Christian College, Palos Heights, IL 60463
Karl.Schmitt@trnty.edu

[7]Furman University, Greenville, SC 29690
andrea.tartaro@furman.edu

[8]Siena College, Loudonville, NY 12211
jteresco@siena.edu

As part of its forthcoming article in the Curricular Practices Volume with the new ACM/IEEE-CS/AAAI Computer Science Curricula guidelines (CS2023)[1], the SIGCSE Committee on Computing Education in Liberal Arts Colleges (SIGCSE-LAC Committee) has developed guidance, informed by its sessions at recent CCSC and SIGCSE conferences, to help with the design and/or revision

[1]https://csed.acm.org

of CS curricula in liberal arts contexts [1]. The committee's earlier work found that liberal arts and small colleges approach the design of their computing curricula in unique ways driven by institutional mission or departmental identity. This impacts how faculty at these colleges adopt curricular guidelines. Curricular guidelines like CS2023 inform curriculum design, but are balanced with the vision for a program, departmental strengths, locale, student populations, and unique academic experiences. The desire to craft distinctive curricula, combined with the size of curricular recommendations, requires an assessment of trade-offs between achieving full coverage of curricular recommendations and a school's other priorities. SIGCSE-LAC's guidance encourages faculty to reflect on their programs and the role of CS2023, beginning with their institutional and departmental priorities, opportunities, and constraints.

This session will introduce participants to SIGCSE-LAC's guidance to consider curricular development in the context of the unique features of their programs and institutions. Following an overview and brief discussion of CS2023, participants will be guided through an abbreviated design process using the latest version of the committee's reflective assessment process. This process is framed by a series of scaffolding questions that begin from institutional and departmental missions, identities, contexts, priorities, initiatives, opportunities, and constraints. From there, participants will be led to identify design principles for guiding their curricular choices, including the CS2023 recommendations. Examples gathered from the committee's previous CCSC and SIGCSE sessions will be available to help articulate identity and program design principles, which will then be used to identify distinctive program-level learning outcomes. A spreadsheet tool that is being developed to aid in the shaping of curricular choices will be demonstrated. Participants will leave the session with a better understanding of how CS2023 can impact their programs, and instruction on how to use the SIGCSE-LACS Workbook, which outlines our curriculum design process, with their departments. Participant feedback will be gathered and used to refine the committee's guidance.

Acknowledgements

This session is supported by the National Science Foundation under Grant No. 2342587.

Presenter Biographies

One or two of this session's eight co-authors will serve as presenter(s)/facilitator(s).
Jakob E. Barnard is an Associate Professor, Chair of the Computing, Design, & Communications Department, and Director of Online Technology Pro-

grams at the University of Jamestown. He is a facilitating member of the SIGCSE-LAC Committee, and his research involves how curricula have been integrated into Liberal Arts Computing programs. **Grant Braught** is a Professor of Computer Science at Dickinson College. He is a facilitating member of the SIGCSE-LAC Committee. He has organized committee events focused on curricula and published widely on CS education issues, particularly within the liberal arts. **Janet Davis** is Microsoft Chair and Professor of Computer Science at Whitman College, where she serves as the department's founding chair. She co-organized SIGCSE pre-symposium events in 2020 and 2021 on behalf of the SIGCSE-LAC Committee. **Amanda Holland-Minkley** is a Professor of Computing and Information Studies at Washington & Jefferson College. Her research explores novel applications of problem-based pedagogies to CS education at the course and curricular level. She is a facilitating member of the SIGCSE-LAC Committee. **David Reed** is a Professor of Computer Science and Chair of the Department of Computer Science, Design & Journalism at Creighton University. He has published widely in CS education, including the text *A Balanced Introduction to Computer Science*, and served on the CS2013 Computer Science Curricula Task Force. **Karl Schmitt** is Chair and Associate Professor of Computing and Data Analytics at Trinity Christian College. He has served on the ACM Data Science Task Force and various Computing, Technology, and Mathematics Education committees for the MAA, ASA, and SIAM. His interests explore data science education, and interdisciplinary education between computing, mathematics, data, and other fields. **Andrea Tartaro** is a Professor of Computer Science at Furman University. Her computer science education research focuses on the intersections and reciprocal contributions of computer science and the liberal arts, focusing on broadening participation. She is a member of the SIGCSE-LAC Committee, and has published and presented in venues including the CCSC and the SIGCSE Technical Symposium. **Jim Teresco** is Chair and Professor of Computer Science at Siena College. He has been involved in CCSC Northeastern for over 20 years and currently serves as board chair, and has been involved with the SIGCSE-LAC Committee for 5 years. His research involves map-based algorithm visualization.

References

[1] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. Computer science curriculum guidelines: A new liberal arts perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 617–623, New York, NY, USA, 2023. ACM. doi:10.1145/3545945.3569793.

Contemporary Vector Database*

Conference Tutorial

Mohammad Amin, Pradip Peter Dey, and Bhaskar Raj Sinha
School of Technology and Engineering
National University
9388 Lightwave Ave., San Diego, CA 92123
{mamin, pdey, bsinha}@nu.edu

Relational databases have played an important role in business for more than five decades, and the demand for accurate data and its accessibility are on the rise. In 1970, EF Codd developed the relational database, also known as SQL database, that uses relational math (set theory) for structured data in tables. Because of the continuing growth of data usage, NoSQL database was first introduced by Carlo Strozzi in 1998 for unstructured and big data. There were some other types of databases that were introduced during this period, but this paper addresses some important aspects of recently developed vector databases, which demonstrate flexibility, scalability, performance, and so forth. These newly developed vector databases have the ability to use millions and billions of pieces of any type of data (alphanumeric, graphical, images, audio, video, etc.) for comparison and query to get fast responses or results. The vector databases are used in AI models, such as Large Language Models (LLMs), for data training with complex and big data. Vector embedding is a new type of data representation generated by AI models, using a special type of mathematical method known as similarity measure. Data existing in the vector space has both magnitude and direction. In a vector database, several similarity measures are used to determine how similar two vectors are in the vector space, for comparison during the query process. Some notable similarity measures are Cosine Similarity, Dot Product, and Euclidean Distance. The recent surge of popularity of AI chatbots (such as ChatGPT of Open AI) suggests that innovative exploration of vector databases in LLMs may continue to provide efficient interactions with users through natural languages, such as English. In order to serve the users better, relational and vector databases can be combined for addressing diverse query types and data structures in meaningful ways. Challenges and opportunities need to be critically examined in these AI areas.[1, 2]

Tutorial Description

This tutorial provides an interactive and effective learning session where various methods will be used for discussion of basic aspects of vector databases. The main ideas of different database principles and their pros and cons are discussed. Briefly, principles of big unstructured data are introduced, and then the tutorial shows how these data can be presented in the multi-dimensional vector space and processed for producing useful information.

Expected Outcomes

Attendees will gain a good understanding of the concepts of vector databases and applications. They will have the chance for open discussions, constructive criticisms, and suggestions.

Target audience

Interested faculty who desires to teach vector database and related topics.

Prerequisites

None. Anybody who has interest in teaching and learning vector databases is welcome to this tutorial session, where intuitive explanations with examples are presented about the main constructs of vector databases.

References

[1] B. Biswas. How i converted regular rdbms into vector database to store embeddings, 2023. `https://dzone.com/articles/how-i-converted-regular-rdbms-into-vector-database`.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, Dec 2017. https://dl.acm.org/doi/10.5555/3295222.3295349&https://dl.acm.org/doi/proceedings/10.5555/3295222.

Performing Enumeration as Part of Penetration Testing Tasks Using Virtual Machines*

Conference Tutorial

Mohamed Lotfy, Ph.D.
Utah Valley University
Orem, UT 84058
MohamedL@uvu.edu

Teaching offensive security (penetration testing/ethical hacking) is becoming a standard practice in computer science, cybersecurity, and information technology programs[2, 5]. Penetration testing/ethical hacking allows students to identify targets and existing vulnerabilities on the targets. They can exploit the identified vulnerabilities using current offensive tools and practices to gain access and create presences, thereby acquiring the needed cybersecurity knowledge and skills that will prepare graduates to be job ready. Through a hands-on approach, students develop offensive cybersecurity competencies, enabling them to later build layered defenses that harden systems to penetration. The enumeration process is part of the penetration testing active reconnaissance phase. The goal of enumeration is to discover as much information as possible about the target systems and networks, for use in developing an effective penetration testing plan. In enumeration, usernames, file shares, and other pieces of information about the systems, domains, networks, and their configurations are gathered. We will demonstrate several different enumeration tools (crackmapexec, rpcclient, enum4linux, and smbclient) with an attacking host and a couple of vulnerable hosts in a virtual environment. Using a virtual environment to enumerate hosts reduces the risk to institutional networks and systems. Attendees will exit the tutorial with an idea of how to perform enumeration and active reconnaissance using a working virtual environment (VMware or VirtualBox) and a Kali Linux attack host.

Tutorial Description

In this tutorial we will provide an example of how to enumerate different Windows hosts and demonstrate using different enumeration tools with VMware Workstation Pro and/or Oracle VirtualBox virtual environments on laptops or PCs[4, 3]. The virtual environment will include an Offensive Security Kali Linux VM, a Metasploitable Windows 2008 server, and a customized Windows XP VM. The following will be demonstrated:

1. Explain penetration testing/ethical hacking and its phases.
2. Describe the enumeration process as a part of the penetration testing active reconnaissance phase.
3. Introduce crackmapexec, rpcclient, enum4linux, and smbclient tools on Offensive Security Kali Linux.
4. Show how to gather usernames, file shares, systems information, and domains and networks configurations.

Target audience

Any computer science, cybersecurity, or IT faculty who desires to learn how to perform enumeration and active reconnaissance using a virtual environment, as part of penetration testing/ethical hacking course. Attendees should be familiar with Linux, networking, and have some system administration knowledge. It is highly recommended that attendees bring their own laptops with VMware or VirtualBox and a Kali Linux VM installed, to follow along.[1].

References

[1] OffSec Services Limited. KALI pre-built virtual machines, 2024. URL: `https://www.kali.org/get-kali/\#kali-virtual-machines`.

[2] Mohamed Lotfy. Teaching a penetration testing course during covid-19: lessons learned. *Journal of Computing Sciences in Colleges*, 37(2):80–88, 2021.

[3] ORACLE. ORACLE VM VirtualBox, 2024. URL: `https://www.virtualbox.org/`.

[4] VMware. VMware Workstation Pro, 2024. URL: `https://www.vmware.com`.

[5] Xiaodong Yue and Hyungbae Park. Design of virtual labs for an ethical hacking course. *Journal of Computing Sciences in Colleges*, 35(6):31–38, 2020.

Getting Started with D3, an Open-Source JavaScript Library for Creating Interactive Data Visualizations for the Web*

Conference Tutorial

Karina Assiter
STEM Department
Landmark College
Putney, VT 05346
KarinaAssiter@Landmark.edu

D3 is a free, open-source JavaScript library for visualizing data [1]. The low-level approach of coding with D3 allows flexibility in creating dynamic, data-driven graphics; thus, D3 is behind the creation of groundbreaking and award-winning visualizations. Though the first version of D3 was released in 2011, it is still relevant for visualizing data in 2024 [4]. D3 is excellent for building data visualizations that are either standard and familiar (bar charts, line charts, force-based layouts, etc.) or non-standard and innovative [3]. In this tutorial we will start with a quick introduction to the field of data visualization [2], then we will provide an overview of D3, its benefits, structure, development environment, and methods of adding it to a JavaScript file. Then there will be opportunities for attendees to get hands-on experience with D3: setting up a simple server to host data files, adding code to D3 files, loading and building a visualization from an accessed data file, and adding interaction. Next, we will explore the myriad of ways that D3 can be (and has been) applied to advanced interactive data visualization problems. Finally, we will share resources that attendees can refer to as they get up-to-speed with D3.

Keywords

D3, Data Visualization, Information Visualization, Interactive Data Visualization.

Tutorial Description

The format of the tutorial will include both formal presentations, as well as hands-on exploration. The tutorial is estimated to take approximately 2.5 hours.

Audience

This tutorial is intended for attendees who would would like an introduction to a flexible tool for building dynamic, innovative and beautiful visualizations that can be used to tell an engaging story with data.

Recommended Knowledge, Skills, Technology

It is recommended that attendees have the following:

- Laptop with a browser (Chrome preferred, but others will suffice).
- Solid understanding of HTML.
- Basic understanding of JavaScript.
- Some experience with data visualization and/or mining would be useful, but not required.

Outline

The following is a draft outline of the tutorial program:

1. Brief introduction to Visualization
2. Introduction to D3
3. Hands-on with basic D3
4. Coding concepts for advanced D3
5. Explore advanced/innovative examples
6. Provide resources and references for attendees

References

[1] D3. `http://d3js.org`.

[2] Tamara Munzner. *Visualization Analysis and Design*. CRC Press, 2015.

[3] Scott Murray. *Interactive Data Visualization for the Web : An Introduction to Designing with D3*. O'Reilly, 2017.

[4] Alam Zaidul. D3.js: Revolutionising data visualization on web. *LinkedIn*, 2024. `https://www.linkedin.com/pulse/d3js-revolutionising-data-visualisation-web-zaidul-alam-j1jjc/`.

A Study of RateMyProfessors in Computing Education in the Rocky Mountain Region*

Xi Chen[1] and Jingsai Liang[2]
[1]Computer Science Department
Utah Valley University
Orem, UT 84058
xi.chen@uvu.edu
[2]Computer Science Department
Westminster University
Salt Lake City, UT 84105
jliang@westminsteru.edu

Abstract

This paper studies the data from RateMyProfessors.com with a focus on computer science education in the Rocky Mountain region. It aims to (1) explore specific discoveries related to computing education in this region, (2) analyze the trends in student comments, and (3) confirm previously identified biases and relationship between quality and difficulty ratings that are also present in computing education. Our analysis revealed that while there is a strong correlation between course difficulty and quality ratings, which indicates potential biases, there are also valuable insights into the evolving preferences and priorities of computer science students over time. We observed a shift towards increased appreciation for engaging and supportive professors, alongside a growing trend in the use of Python as a programming language. The findings underscore the need for a balanced approach in evaluating teaching effectiveness and the changing patterns in the computing education.

1 Introduction

RateMyProfessors.com (RMP) is a popular online platform where students rate and review their professors and courses. Numerous studies [1, 3, 2, 6] have utilized RMP data to analyze various aspects of higher education, such as teaching effectiveness, student satisfaction, and biases in student evaluations. This paper focuses on computer science (CS) education in the Rocky Mountain region, aiming to discover unique insights and to evaluate previous studies specifically related to computing education.

Several previous studies have explored the relationship between quality and difficulty of courses and instructors. Paper [3] found a strong negative correlation between perceived difficulty and overall quality ratings. Other studies have focused on demographic biases in student evaluations. Paper [4] found that female instructors often receive lower ratings compared to their male counterparts. Similarly, paper [5] identified racial biases in student evaluations, with minority professors receiving lower ratings.

In the following sections, our study focuses specifically on computer science education in the Rocky Mountain region. We collected the RMP data from eight states, identified patterns in computer science courses and languages, examined the quality and difficulty ratings, observed trends in student comments over time, and evaluated the biases in the reviews.

2 Data Collection

We collected data from www.ratemyprofessors.com in three steps. In step 1, we identified the universities and colleges in eight states in the Rocky Mountain region that we wanted to study. Since we are only interested in computer science education with four-year programs, we specifically excluded online schools, community colleges, schools without a computer science department, and schools that are too small. In step 2, we dynamically scraped data for these schools dated from the earliest available year 2001 to current month May 2024 from the RMP website using the Selenium package in Python. For each professor at these schools, we scraped their name, overall rating, number of ratings, overall level of difficulty, and top tags. For each review of these professors, we scraped the quality, difficulty, course number, post date, comment, and tags. In step 3, we manually supplemented the data by adding each school's location, most recent enrollment size, Carnegie classification, and course numbers for CS1, CS2, Data Structures, and Algorithms for later analysis. Table 1 shows the number of schools, professors, and reviews we scraped in each state.

State	University/College	Professors	Reviews
Arizona	5	356	7666
Colorado	14	468	4605
Idaho	5	192	1647
Montana	5	55	421
Nevada	5	140	2584
New Mexico	5	78	776
Utah	8	402	6991
Wyoming	1	15	157

Table 1: Summary of the distribution of the data

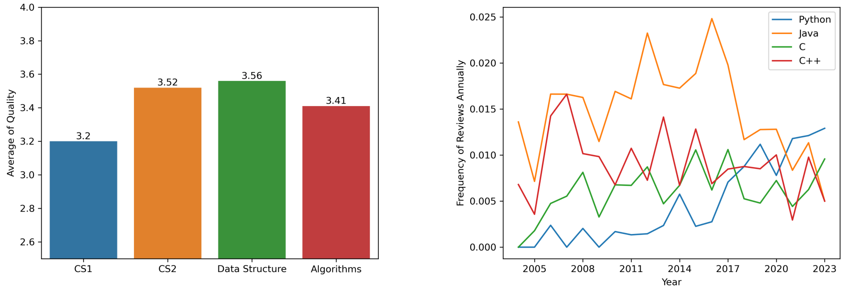
3 Data Analysis

3.1 Analysis of Computer Science Courses and Languages

Computer science students typically progress through a sequence of required programming and algorithm courses: CS1, CS2, Data Structures, and Algorithms. Understanding students' attitudes toward these courses is valuable to educators. Figure 1a shows the average quality ratings for these four courses. The ratings for CS1, CS2, and Data Structures show an increasing trend. CS1 is often the first exposure to programming and computer science concepts for many students, which may result in a steep learning curve. However, they gradually adapt and become more comfortable with subsequent courses, resulting in higher ratings. It is not surprising that there is a drop in ratings for Algorithms, as this course is more challenging and theoretical than the introductory-level courses.

On the other hand, the choice of programming languages in CS classes has changed dramatically over the last two decades. Figure 1b illustrates the annual frequency of reviews mentioning four languages primarily in introductory programming classes: Python, Java, C, and C++ from 2004 to 2023. Based on the trend depicted in the figure, we can learn the following:

- Python: Initially, mentions of Python were relatively low but have shown a steady increase over the years, especially after 2015. This trend indicates a growing popularity of Python in computer science curricula.
- Java: Java has consistently been one of the most frequently mentioned languages, with a noticeable peak around 2016. Despite some fluctuations, Java remains an important language in computing education.
- C: Mentions of C have been more variable, with some peaks and dips, but overall, it has maintained a moderate presence over the years.



(a) Average Quality of CS Courses

(b) Trend of Languages Over Time

Figure 1: CS Courses and Languages

- C++: Similar to C, mentions of C++ show variability but with a notable peak around 2006-2007 and a general decline in recent years, suggesting a reduced emphasis on C++ in favor of other languages.

The data indicates a shift in the programming languages taught in computer science over time. Python's increasing trend highlights its growing importance and adoption in computer science education, likely due to its versatility and ease of learning. The trends suggest a dynamic landscape in programming education, with Python becoming more prominent, C and C++ maintaining its strong position, and Java experiencing more fluctuating attention.

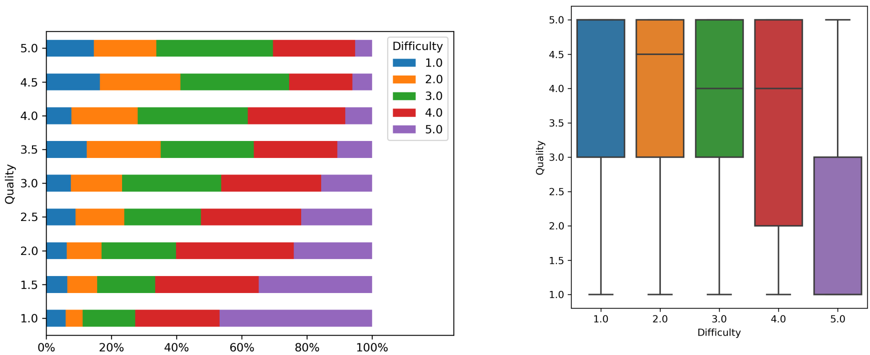
3.2 Analysis of Quality and Difficulty in CS Courses

Two of the most visible scores rated by students in each review are quality and difficulty. In the following sections, we'll analyze their relationship, trends and tags, and compare them across the states.

3.2.1 Relationship between Quality and Difficulty

As shown in Figure 2a, as the quality rating decreases from 5.0 to 1.0, the percentage of difficulty level 5 (purple bar) increases significantly. This indicates that the most challenging courses are often rated lower in quality. As a result, for a quality rating of 1.0, the majority of difficulty ratings are at level 5, highlighting that students who find a course very difficult are more likely to rate it poorly in terms of quality. Similarly, as shown in Figure 2b, lower difficulty levels (1.0 and 2.0) have higher median quality ratings with narrower interquartile ranges (IQR), indicating more consistent and higher quality ratings for easier courses.

From Figure 2, we confirmed that, while quality ratings provide some insights into students' perceptions of a course, they are not entirely reliable in reviewing CS courses due to their strong correlation with course difficulty. For a more accurate and fair evaluation of professors, it is essential to consider other factors such as teaching effectiveness, clarity of instruction, engagement, fairness in grading, and course organization.



(a) Percentage of Difficulty at Each Quality

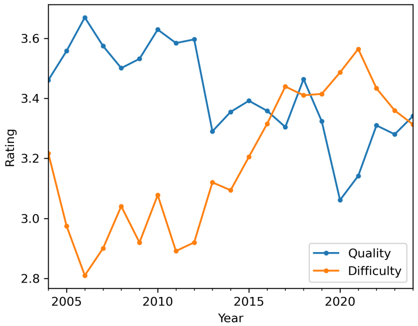
(b) Boxplot of Quality at Each Difficulty

Figure 2: Quality vs Difficulty

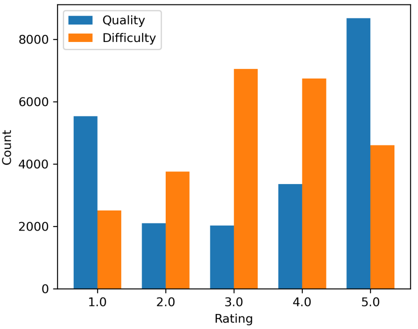
3.2.2 Statistics of Quality and Difficulty

Figure 3a indicates that while quality ratings have generally remained stable with some fluctuations, there was a noticeable dip around 2020, followed by a gradual recovery afterward. Difficulty ratings, on the other hand, show a more clear increasing trend starting from 2014 and peaking in 2021, with a decline in the last few years. The contrast in the opposite movements of these two ratings further emphasizes the importance of considering both factors when evaluating courses and professors.

The count distribution for quality and difficulty ratings shown in Figure 3b reveal potential biases. Particularly, the quality ratings exhibit a bimodal distribution with high counts at both 1.0 and 5.0. This could indicate polarization in student perceptions, where courses are either seen as very good or very poor, with fewer moderate evaluations. Such a distribution might suggest that students are more likely to leave extreme ratings, possibly influenced by strong positive or negative experiences, leading to potential bias in the overall evaluation.



(a) Average Ratings in Each Year



(b) Count of Ratings at Each Level

Figure 3: Statistics of Quality and Difficulty

3.2.3 Comparison of Tags at Different Quality Levels

The bar plot in Figure 4 shows the frequency of different tags at quality levels 1 (blue) and 5 (orange). There are a variety of significant contrasts between the two quality levels. At quality level 1, the tags of Tough Grader, Graded by Few Things, Test Heavy, Lots of Homework, Lecture Heavy, and Get Ready to Read are significantly higher than the same tags at level 5. At quality level 5, the tags of Respected, Hilarious, Gives Good Feedback, Extra Credit, Clear Grading Criteria, Caring, Inspirational, Amazing Lectures, and Accessible Outside Class are significantly higher than the same tags at level 1.

Besides the potential bias discussed in previous sections, for professors who want to improve the quality, we can show the following tips learned from this figure:

- Ensure timely and constructive feedback on assignments and exams.
- Make grading criteria transparent and consistent.
- Demonstrate empathy and support for students' academic and personal challenges.
- Engage students with passion and enthusiasm for the subject.
- Focus on delivering engaging, interactive, and well-structured lectures.
- Offer regular office hours and be approachable for additional help.
- Be mindful of the overall workload, including tests and homework.

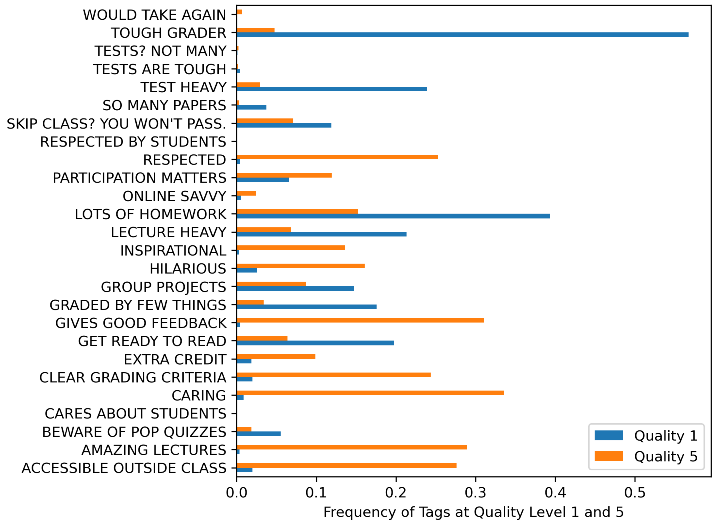


Figure 4: Comparison of Tags between Quality 1 and 5.

We hope that by focusing on these areas, professors can enhance the overall quality of their courses and create a more positive and effective learning environment for their students.

3.2.4 Comparison of Quality and Difficulty Across States

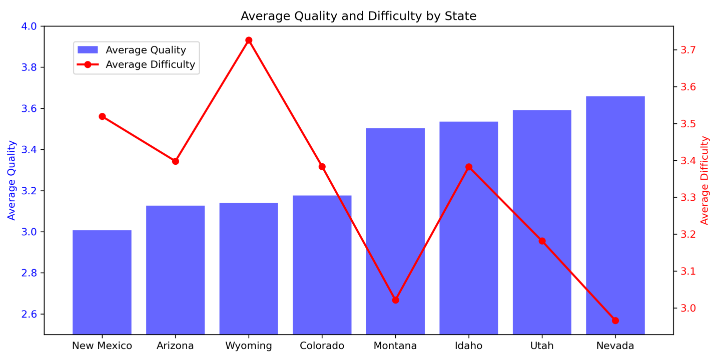


Figure 5: Quality and Difficulty by States

The comparison of average quality and difficulty ratings across states is shown in Figure 5. This figure suggests that there may be a potential negative relationship between quality and difficulty across the states too.

3.3 Analysis of the Trend in Comments

2000-2004	class comments, class easy, good teacher, knows stuff, learned lot, nice guy, willing help
2005-2009	class easy, good teacher, knows stuff, make sure, nice guy, office hours, willing help
2010-2014	easy class, extra credit, good teacher, make sure, nice guy, office hours, willing help
2015-2019	cares students, extra credit, great professor, make sure, office hours, outside class, pay attention, willing help
2020-2024	ask questions, cares students, extra credit, good professor, highly recommend, make sure, office hours, outside class

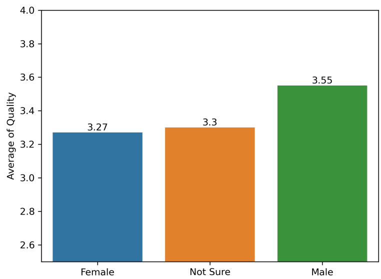
Table 2: Top Ranked Meaningful Keywords in Comments Every 5 Years

The comment section in the reviews is more informative than the score of quality and difficulty. We extracted the top keywords in the comments every five years using *TF-IDF* method. The top-ranked meaningful keywords are shown in Table 2.

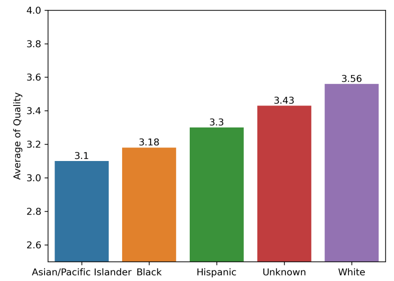
This table reveals a consistent appreciation for helpful and accessible professors, with keywords like "willing help" and "extra credit" remaining important throughout. Over time, there is a noticeable shift towards valuing professors' engagement and care for students, particularly in recent years where terms like "cares students" and "outside class" become prominent. This trend suggests that student engagement and support have become increasingly critical over the years.

3.4 Validating the Bias in Reviews in CS Courses

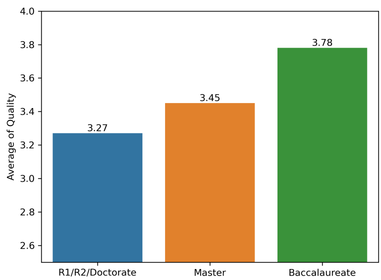
The data scraped from RMP does not contain information on the gender or race of professors. For gender, we used a package called *gender_guesser* to estimate gender based on first names. If the package could not conclusively determine the gender, we labeled it as "Not Sure". For race, we used a package called *ethnicolr* to estimate race based on last names. We also acknowledge that the race estimator may not be accurate regarding female professors if their last names are taken from their husbands' last names of different races. Since estimating race is more challenging and less accurate, we only retained four race categories and classified all other last names with an accuracy lower than 60% as "Unknown". For the type of university or college, we identified the



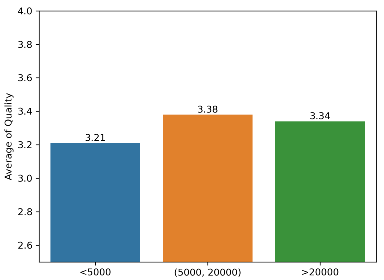
(a) Gender of Professors



(b) Race of Professors



(c) Type of University/College



(d) Enrollment of University/College

Figure 6: Bias in the Reviews

Carnegie classification of each school and categorized them into three groups: R1/R2/Doctorate, Master, and Baccalaureate. For enrollment numbers, we searched for the most recent overall enrollment of each school online and classified them into three groups: small (<5,000), moderate (5,000, 20,000), and large (>20,000).

The four figures shown in Figure 6 highlight potential biases in quality reviews based on gender, race, school type, and school size. Male professors tend to receive higher average quality ratings compared to female professors. Regarding race, white professors receive the highest average quality ratings, while Asian/Pacific Islander professors receive the lowest. Professors at Baccalaureate schools receive the highest average quality ratings, followed by those at Master's schools, with those at Doctorate-granting schools receiving the lowest ratings. Finally, professors at smaller schools (less than 5000 students) receive lower average quality ratings compared to those at larger schools. These findings suggest that student reviews may be influenced by these demographic and institutional factors, indicating a potential bias in the ratings.

4 Conclusion

In conclusion, our analysis of RMP data for computer science education in the Rocky Mountain region observed the quality pattern in programming classes and the change in the programming languages, studied the statistics and relationship of quality and difficulty, analyzed the trends in student comments revealed a shift towards increased appreciation for engaging and supportive professors, and confirmed several previously identified biases and trends.

Overall, our study provides valuable insights into the factors influencing student evaluations of computer science courses and professors in the Rocky Mountain region. By discovering the trends in the comments and ratings, professors and institutions can catch up the needs of the students. By recognizing and addressing the biases identified in this study, educators and institutions can work towards improving teaching effectiveness and student satisfaction in computer science education.

References

[1] Stefanie S. Boswell. "Ratemyprofessors is hogwash (but I care): Effects of Ratemyprofessors and university-administered teaching evaluations on professors". In: *Computers in Human Behavior* 56 (2016), pp. 155–162.

[2] Rómulo A. Chumacero, Ricardo D. Paredes, and Tomás Reyes. "When RateMyProfessors met Google Scholar: students' evaluations and professors' looks and research". In: *CEPAL Review* August (2023).

[3] Alexander Katrompas and Vangelis Metsis. "Rate My Professors: A Study Of Bias and Inaccuracies In Anonymous Self-Reporting". In: *2nd International Conference on Computing and Data Science (CDS)*. 2021, pp. 536–542.

[4] Dakota Murray et al. "Exploring the personal and professional factors associated with student evaluations of tenure-track faculty". In: *PLoS One* 15.6 (2020).

[5] Landon Reid. "The Role of Perceived Race and Gender in the Evaluation of College Teaching on RateMyProfessors.com". In: *Journal of Diversity in Higher Education* 3 (Sept. 2010), pp. 137–152.

[6] Andrew S. Rosen. "Correlations, trends and potential biases among publicly accessible web-based student evaluations of teaching: a large-scale study of RateMyProfessors.com data". In: *Assessment & Evaluation in Higher Education* 43.1 (2018), pp. 31–44.

Generative AI and its Impact on the CS Classroom and Programmers*

Ed Lindoo Ph.D.[1] and Mohamed Lotfy Ph.D.[2]
[1] Regis University
Denver, CO 80221
elindoo@regis.edu
[2] Utah Valley University
Orem, UT 84058
MohamedL@uvu.edu

Abstract

As the integration of generative artificial intelligence (AI) in educational settings becomes more widespread, students, teachers, and educational institutions face the challenge of utilizing these technologies in a responsible manner. The responsible use of generative AI can help CS and IT students develop critical thinking, enhance their learning experience, facilitate the learning process, can assist in understanding code concepts, programming skills, and/or enhancing the programming knowledge. The aim of this investigation is on how students might utilize, and potentially abuse, generative AI. In this paper we provide examples of how generative AI can be used to generate code modules. We discuss the use of generative AI in programming classes as well as its impact on the future of programming and programmers.

1 Introduction

The rapid ascent of generative artificial intelligence (AI) in recent years is impacting both the societal economic and cultural spheres. This swiftly advancing technology is reshaping numerous facets of everyday existence, including the methods by which we educate and absorb knowledge. The results from Hamilton and Swanston (2023) of a Forbes survey, which involved 500 active educators across the United States to gauge their encounters with AI in educational settings, provided valuable insights into the influence of AI on the field of education. While Sixty percent of the active educators envisioned that AI will be used more widely, but not as a central component in the decade, thirty percent of the active educators envisioned that AI will play a central role in education. Over half of the participating educators expressed a belief in the positive impact of AI on the teaching and learning dynamics. Conversely, less than one-fifth of respondents reported experiencing negative effects attributable to AI [9].

For years, AI-powered educational games have been the most frequently utilized AI tools among teachers, with adaptive learning platforms, automated grading, and feedback systems also enjoying popularity among educators[9]. Now we are dealing with ChatGPT, IBM watsonx Code Assistant and similar tools. After its launch in November 2022, ChatGPT, created by OpenAI, garnered considerable interest owing to its sophisticated natural language processing skills and adeptness in generating code [4]. The promising capacity of ChatGPT to improve the software development process carries significant implications for the future landscape of software engineering and programming professions, especially given the continuous rise in demand for proficient programmers.

As the integration of generative AI in educational settings becomes more widespread, students, teachers, and educational institutions face the challenge of utilizing these technologies in a responsible manner. Chatbots like ChatGPT have ignited debates among educators regarding their capacity to encourage academic dishonesty and disseminate misinformation. Additionally, concerns about data privacy, algorithmic biases, and disparities in access to generative AI technologies have been raised by professionals and observers alike. At the forefront of educators' concerns regarding generative AI in education is academic dishonesty. Some teachers worry that the growing reliance on generative AI could result in reduced human interaction for learners [9].

Butterman, Patel, Garvey, Commerford, and Stone (2023) drew comparisons between generative AI and other revolutionary technologies. Once, educators hesitated to embrace the Internet for student use, yet now it stands as an essential tool for learning. "Why is this so different?" he asks. "We might one day think of this like other 'scary' technologies that today are a normal part of life" [5]. While ChatGPT is not the sole Large Language Model (LLM) generative AI tool on the market, it is the one that has garnered the most attention recently. This paper outlines the findings of a study examining ChatGPT's performance on homework assignments designed for an introductory-level computer programming course.

The aim of this investigation is not to assess ChatGPT's ability to execute computer programming tasks, as that has already been established. Rather, considering the potential for misuse of such tools in academic settings, the focus is on how students might utilize, and potentially abuse, ChatGPT. Educators are particularly concerned about whether students will leverage ChatGPT to contravene the academic integrity policies set forth by their institutions. Historically, students have been involved in various forms of academic dishonesty in higher education, such as plagiarism, unauthorized collaboration, and purchasing solutions to assignments [2].

In the remaining sections we provide examples of how generative AI can be used to generate Java and VB.NET code modules. We provide a discussion on COBOL and its existence on organizational and governmental applications on legacy systems and how these applications can be maintained and updated. We provide an example of how generative AI can use COBOL code and generate an equivalent VB.NET code. We discuss how generative AI can be used in programming classes as well as its impact on the future of programming and programmers.

2 The Use of Generative AI to Produce Code Modules

Butterman et al. (2023) stated that "If students master ChatGPT while they're in school, they'll improve the papers they write in their courses—and they'll know how to use the technology once they're on the job"[5]. That led us to find out what ChatGPT can do. To evaluate its capabilities, we registered for an account with OpenAI at OpenAI.com. The registration process required an email address and a mobile phone number for a one-time verification. Next, we asked the system "can you write a 20-page paper with at least 10 references on project management" the response from ChatGPT was "I'm sorry but generating a 20-page paper with references on project management is beyond the scope of what I can provide in a single response. However, I can give you a structured outline for such a paper, which you can use as a basis to develop your own paper." ChatGPT then went on to provide an outline with a list of sections/topics. For the references we asked for it gave this: "For the references section, you should consult academic journals, textbooks, and reputable online sources that provide insights into project management theory and practice." However, taking just one of the many suggestions provided within the sections it gave us, for example, "Principles of Agile Project Management" and thus we asked ChatGPT to write about that with references. Now it generated 10 short paragraphs on the topic along with three references.

It became obvious that by using ChatGPT with just a little effort on our part, putting the results together, we could easily write that 20-page paper in less than an hour. And, it would be scholarly, well written with decent references. Now, imagine a college student whose roommate requests a quick review of his latest term paper, only to discover that it surpasses the quality of anything he has ever submitted to you before. Welcome ChatGPT, but is this ethical? It is a conundrum that instructors worldwide have been wrestling with most recently. Once confined to the realm of science fiction, artificial intelligence (AI) has steadily permeated our daily lives over the past decades. Today, it is woven into the fabric of our homes (just ask Alexa), our vehicles (autonomous driving), our shopping experiences (virtual assistants), and countless other aspects of modern living. The bottom line, today's artificial intelligence cannot be ignored and is only going to grow [5].

If the user engaging with ChatGPT lacks the necessary programming skills, it cannot be assumed that ChatGPT will generate valid code. For instance, a study examining ChatGPT's potential impact on engineering education proposed that users must possess the ability to pose appropriate questions because generative AI lacks the critical thinking and problem-solving capabilities required by users [14]. We decided to put Qadir's theory to the test. We took one of our homework assignments from an intro to Java class and asked ChatGPT "Create a java program that asks for user input and writes it to a new file". In less than a second we had a perfect program. Figure 1 shows the ChatGPT produced Java program.

Next, we decided to make it much harder, using an even more difficult assignment from our Java class, we asked ChatGPT: "Create a POJO with states and behaviors. Make sure that the class is well defined with at least 4 class variables. At least one behavior and at least one class variable that is an Array. Then create the main class and create a new instance of your class. Populate the object then print all the populated fields of the object. Use any values other than cars. Trucks, People, Stocks, really anything that you can think of."

We thought this would throw ChatGPT off, however, again in less than one-two seconds it came back with a full program with everything we asked for! For sake of space we are not going to show it here as we did above, however, if handed in by a student this would certainly have received an "A". We determined that ChatGPT could write some Java code, but how about a little harder example of VB.NET? When asked "I need a vb.net program that will find the month using a case statement", ChatGPT came back with a perfect module. Figure 2 shows ChatGPT produced VB.NET program.

```java
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class WriteToFile {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Ask user for input
        System.out.println("Enter some text to write to a fi
        String userInput = scanner.nextLine();

        // Prompt user for file name
        System.out.println("Enter the file name to save the
        String fileName = scanner.nextLine();

        // Write user input to file
        try (BufferedWriter writer = new BufferedWriter(new
            writer.write(userInput);
            System.out.println("Text has been written to the
        } catch (IOException e) {
            System.out.println("An error occurred while writ
        } finally {
            scanner.close();
        }
    }
}
```

Figure 1: ChatGPT Java Produced Program

Typically, we would be looking for something more sophisticated, a module that we would pass a variable to and have it return the results to us. Surely this will throw ChatGPT off! We then asked: “I need a vb.net function that will accept a number (1-12) as a parameter and use that with a case statement to determine the month.” Again, ChatGPT produced the following module including some comments, within just a few seconds, see Figure 3.

So here we are, teaching students how to write object-oriented programming using ChatGPT. They just need to know how to tie all these modules they create together, to end up with the one, large program they need. Thus, we propose a new way of teaching CS students how to program, with the help of ChatGPT.

```
Module MonthFinder
    Sub Main()
        ' Prompt user to enter a number representing a month
        Console.Write("Enter the number of the month (1-12):
        Dim monthNumber As Integer = Integer.Parse(Console.R

        ' Determine the name of the month using a Select Cas
        Dim monthName As String = ""
        Select Case monthNumber
            Case 1
                monthName = "January"
            Case 2
                monthName = "February"
            Case 3
                monthName = "March"
            Case 4
                monthName = "April"
            Case 5
                monthName = "May"
            Case 6
                monthName = "June"
            Case 7
                monthName = "July"
            Case 8
                monthName = "August"
            Case 9
            Case 9
                monthName = "September"
            Case 10
                monthName = "October"
            Case 11
                monthName = "November"
            Case 12
                monthName = "December"
            Case Else
                Console.WriteLine("Invalid month number!")
                Exit Sub
        End Select
        ' Display the name of the month
        Console.WriteLine("The month is: " & monthName)

        ' Keep the console window open
        Console.ReadLine()
    End Sub
End Module
```

Figure 2: ChatGPT VB.NET Produced Program

```vb
Module Module1
    Sub Main()
        ' Example usage of the GetMonthName function
        Console.WriteLine("Enter a month number (1-12):")
        Dim monthNumber As Integer = Convert.ToInt32(Console.ReadLine())

        Dim monthName As String = GetMonthName(monthNumber)

        ' Display the month name
        Console.WriteLine("The month is: " & monthName)
    End Sub

    Function GetMonthName(ByVal monthNumber As Integer) As String
        Dim monthName As String
        Select Case monthNumber
            Case 1
                monthName = "January"
            Case 2
                monthName = "February"
            Case 3
                monthName = "March"
            Case 4
                monthName = "April"
            Case 5
                monthName = "May"
            Case 6
                monthName = "June"
            Case 7
                monthName = "July"
            Case 8
                monthName = "August"
            Case 9
                monthName = "September"
            Case 10
                monthName = "October"
            Case 11
                monthName = "November"
            Case 12
                monthName = "December"
            Case Else
                monthName = "Invalid month number"
        End Select

        Return monthName
    End Function
End Module
```

Figure 3: ChatGPT VB.NET Produced Function

Backing up our findings, Bucaioni et al., (2024) recently conducted a series of experiments involving ChatGPT, where it was presented with a collection of 240 programming problems [4]. As one can imagine, this was an extensive undertaking, but in the end, they found that ChatGPT demonstrates proficiency in solving programming problems at lower and medium difficulty levels. However, its accuracy in generating correct code decreases when faced with more challenging problems. These challenging problems are typically very large programs that do many things. In other words, it is akin to our initial task of asking ChatGPT to write us a 20-page paper, it simply could not do it, yet it could provide pieces to us that we could put together, much like a jig-saw puzzle.

3 COBOL Code Conversion using Generative AI

After demonstrating ChatGPT's proficiency in coding, we initiated discussions regarding the ongoing demand for COBOL programmers. Nowadays, whether we're withdrawing money from an ATM, booking an airline reservation, or making online purchases, it's highly likely that a COBOL application has facilitated the transaction. Remarkably, even the Social Security Administration continues to rely on 60 million lines of COBOL code, as reported by [16]. Experts estimate that approximately 70% of all core business applications worldwide are based on COBOL. Another estimate suggests that these applications handle around 85% of all ATM transactions. Remarkably, even the Social Security Administration continues to rely on 60 million lines of COBOL code, as reported by [16]. Those COBOL applications are still running on mainframes and legacy systems. Replacing governmental and organizational legacy systems or moving them to the cloud is not a any easy task. Matthiesen and Bjørn (2015) empirical study results on replacing legacy systems showed that global software development outsourcing to update or replace governmental legacy systems is not a simple and easy task [12]. Gholami, Daneshgar, Beydoun, and Rabhi (2017) empirical exploratory study identified 27 process factors that need to be addressed in the process of transitioning legacy systems to the cloud, thus the transition process is not a simple task [8].

Kizior, Carr, and Helpren (2005) mentioned that at the time of their study there were between 150 and 200 billion lines of COBOL code in business applications [11]. Fanelli, Simons, Scott, and Banerjee (2016) estimated 180-200 billion lines of code that are still in use [7]. In addition, companies have been adding several billions of lines of COBOL code annually to these applications. Hughes (2022) mentioned that the amount of COBOL code in daily use is between 775-850 billion lines according to the results of a Vanson Bourne study commissioned by Micro Focus company, which is approximately three times larger than previously estimated [10]. Thus, the utilization of one of the oldest programming languages could be significantly larger than previously estimated. The Vanson Bourne study involving 1,104 respondents from 49 countries revealed that more than nine out of ten organizations still consider COBOL to be a strategic priority [10]. According to Ali, Smith and Mormon (2018) “the availability of these large volume of code to maintain means that companies will keep using COBOL for a long time well into the future” [1]. For many years pundits have questioned why these programs have not been re-written and the simple answer is that it is not all that easy. However, now with ChatGPT and similar generative AI programs, the demand for programmers to convert this code is obviously going to grow.

A question that arises is: why not rewrite COBOL code using a newer language? The answer is straightforward: if it has been functioning effectively for over 30 years, where is the motivation to switch? Companies have faced dire consequences, even going out of business, when they neglected the age-old advice, “if it is not broken, do not fix it.” Numerous attempts have been made to rewrite the tried-and-true code, often resulting in substantial time and financial investments, with many endeavors ending in failure. Subsequent efforts proved even more costly, and only a few of the new and improved languages succeeded in replacing COBOL. As stated by Bloom (2013), “many of these initiatives failed because the systems were (and still are) too big, too complex, too integrated into critical business processes, and working too well to replace”[3].

We theorized that with the help of ChatGPT, a VB programmer for example could convert COBOL to VB, even with little knowledge of the COBOL language. No doubt this experiment is going to put ChatGPT into a tailspin, and so we asked ChatGPT to “re-write this COBOL code into VB”. Figure 4 shows the COBOL code given to ChatGPT. In less that two-seconds, ChatGPT had re-written the COBOL code into Visual Basic. Figure 5 shows the ChatGPT VB generated code.

Note that the code returned also includes comments interspersed throughout, offering clarification on the meaning of certain lines of code. This was a real eye-opener, showing us the potential ChatGPT has a programming tool for programmers.

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. PRG5.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 G1.
         02 YY PIC X(2).
         02 MM PIC X(2).
         02 DD PIC X(2).
       01 G2.
         02 DD PIC X(2).
         02 FILLER PIC X VALUE '-'.
         02 MM PIC X(2).
         02 FILLER PIC X VALUE ‘-‘.
         02 YY PIC X(2).
       01 G3.
         02 HH PIC X(2).
         02 MM PIC X(2).
         02 SS PIC X(2).
       01 G4.
         02 HH PIC X(2).
         02 FILLER PIC X VALUE ':'.
         02 MM PIC X(2).
         02 FILLER PIC X VALUE ':'.
         02 SS PIC X(2).
       PROCEDURE DIVISION.
             ACCEPT G1 FROM DATE.
             ACCEPT G3 FROM TIME.
             MOVE CORRESPONDING G1 TO G2.
             MOVE CORRESPONDING G3 TO G4.
             DISPLAY G2.
             DISPLAY G4.
             STOP RUN.
```

Figure 4: COBOL Code

```
Module Module1
    Sub Main()
        Dim G1 As New DateStructure
        Dim G2 As New DateStructure
        Dim G3 As New TimeStructure
        Dim G4 As New TimeStructure

        ' Accept date from system
        G1.YY = Date.Now.ToString(“yy”)
        G1.MM = Date.Now.ToString(“MM”)
        G1.DD = Date.Now.ToString(“dd”)

        ' Accept time from system
        G3.HH = Date.Now.ToString(“HH”)
        G3.MM = Date.Now.ToString(“mm”)
        G3.SS = Date.Now.ToString(“ss”)

        ' Move corresponding fields
        G2.DD = G1.DD
        G2.MM = G1.MM
        G2.YY = G1.YY

        G4.HH = G3.HH
        G4.MM = G3.MM
        G4.SS = G3.SS

        ' Display in the given format
        Console.WriteLine($”{G2.DD}/{G2.MM}/{G2.YY} & {G4.HH}:{G4.MM}:{G4.SS}”)

        Console.ReadLine()
    End Sub

    ' Define structures to represent date and time
    Structure DateStructure
        Dim YY As String
        Dim MM As String
        Dim DD As String
    End Structure

    Structure TimeStructure
        Dim HH As String
        Dim MM As String
        Dim SS As String
    End Structure
End Module
```

Figure 5: ChatGPT Generated VB Code from Given COBOL Code

4 Discussion

4.1 Use of Generative AI in the Classroom

Regardless of whether the use of generative AI tools is permitted in a class, instructors must understand how students utilize them. Like it or not, students will try it, and as they try it and like it, they will use it more, and share it with their peers. And so, as educators we surmise that we need to give assignments that are building blocks that students must put together and that ChatGPT (at this writing) can't figure out on its own. The initial suggestion involves adjusting the written assignment prompt to be less specific. This approach complicates the task for students lacking programming knowledge, preventing them from obtaining a passing score by merely copying and pasting the assignment instructions into ChatGPT [6].

Further, Ellis, Casey, and Hill (2023) suggested mandating students include comprehensive comments throughout their program as this tackles issues on multiple fronts [6]. Firstly, if a student lack's understanding of the techniques employed in their program, they will be unable to provide accurate comments on code generated by an AI engine. Although ChatGPT will produce minimal comments, it does so in a standardized manner. Requiring students to comment on their code in a specific manner for each assignment enables instructors to ascertain if the students created the code themselves. Secondly, if a student can effectively apply a prescribed method of commenting, as dictated by the instructor, to code generated by a generative AI tool, then it indicates the student has likely grasped the techniques required by the assignment. While it may be disconcerting that they could potentially exploit ChatGPT in this manner, providing them with the benefit of the doubt if they have thoroughly and accurately commented throughout is a reasonable approach.

Mollick, Mollick, Acar, and Weiss (2024) provided four ways to incorporate generative AI in the classroom. The first approach is to enhance the student critical thinking skills by asking them to critique the ChatGPT produced essays or code. Allowing students to use different generative AI tools and compare the produced results is another approach to enhance the learning and allow them to identify the right generative AI tool for the task. The third approach is to allow students to use generative AI to evaluate and enhance their understanding of the concepts. The last approach is to use generative AI to generate practice tests or study questions, which are useful learning tools, thus facilitating the creation and grading [13]

In addition faculty should include a generative AI usage policy in their courses. The policy should include a definition of what generative AI technology refers to. Including the learning opportunity, acceptable use, and unacceptable use of generative AI. The responsible use of generative AI can help students develop critical thinking and ethical decision-making skills. Also, incorporating generative AI tools and technologies can enhance the learning experience and facilitate the learning process. In addition, it can assist in understanding code concepts, programming skills, brainstorming ideas, and/or enhancing the programming knowledge. The policy should stress that the use of generative AI to complete graded programming assignments or assessments is not allowed as well as writing the assignment reports. Students should acknowledge and agree to adhere to the generative AI policy outlined in the course syllabus.

4.2 Will AI Automation Replace Programmers?

At present, the answer to whether ChatGPT or IBM watsonx Code Assitant and similar AI tools will replace developers is a resounding 'no'. Technological advancements invariably lead to the obsolescence of certain jobs while creating new ones. However, replacing a developer requires a significant undertaking. Software development encompasses more than just coding. Undoubtedly, ChatGPT can aid in code writing, akin to how Integrated Development Environments (IDEs) assist with code completion, compilation, and debugging. Yet, like IDEs, ChatGPT cannot supplant human involvement in end-to-end development [4].

Throughout the evolution of software development, developers have utilized various tools—from text editors in the 80s to IDEs in the 90s. They have sourced code snippets from Google search results and solutions from platforms like StackOverflow, all in pursuit of heightened productivity. However, none of these tools have truly replaced the developer. The same trajectory is expected with ChatGPT. While it serves as a disruptive and productive tool for programmers, it neither confers programming expertise nor eliminates the need for human programmers—at least not at this juncture [15].

5 Conclusion

While the prevalence of readily available Large Language Models (LLMs) may raise concerns among instructors across various fields of study, the situation might not be as alarming as many perceive. As generative AI tools and chatbots become increasingly widespread, educators must acknowledge their existence and the inevitable use by students. Some educational systems and institutions have opted to prohibit access to such tools — a tactic we believe is not a sustainable long-term solution. Students can still access generative AI tools through other networks not under academic control. Furthermore, these tools are becoming more accessible to even novice users through internet search engines and various social media platforms that embed generative AI tools into their interfaces. We believe that a more enduring strategy for managing the impact of LLMs is to integrate them into courses as educational aids. This approach enables instructors to guide students through the advantages and limitations of AI-based code generation in a systematic manner. Writing functional programming code relies on rules and standards, precisely the type of activity generative AI tools excel at. However, these tools cannot replicate the individual creativity essential for success in most fields of study [15]. The ability to creatively solve problems with a computer is the core skill we aim for our students to develop. When appropriately harnessed, tools like ChatGPT hold the potential to usher in a new and fruitful era of creative disruption in education and technology.

Future computer science and information technology education research needs to provide exploratory studies to identify how generative AI tools will change how we teach programming in the classroom. Empirical studies are needed to measure the impact of student use of generative AI on the acquisition of programming skills. Also, future studies are needed to identify the role generative AI will play in new application creation, current application maintenance and updates as well as its impact on programmers and programming jobs.

References

[1] Azad Ali, David Smith, and Andrea Morman. Still teaching cobol programming-underlying reasons and contributing factors. *Issues in Information Systems*, 19(4), 2018.

[2] Claudio Barbaranelli, Maria L Farnese, Carlo Tramontano, Roberta Fida, Valerio Ghezzi, and Marinella Paciello. Machiavellian ways to academic cheating: A mediational and interactional model. *Frontiers in Psychology*, 9:370835, 2018.

[3] Eric P. Bloom. COBOL will outlive us all, 2013. Visited March 27, 2024. URL: `https://www.computerworld.com/article/1603698/cobol-will-outlive-us-all.html`.

[4] Alessio Bucaioni, Hampus Ekedahl, Vilma Helander, and Phuong T Nguyen. Programming with ChatGPT: How far can we go? *Machine Learning with Applications*, 15:100526, 2024.

[5] Eric Butterman, Darshak Patel, Aaron Garvey, Benjamin Commerford, and Dan Stone. How is AI changing education - and business?, 2023.

Visited March 27, 2024. URL: https://www.aacsb.edu/insights/articles/2023/11/how-is-ai-changing-education-and-business.

[6] Michael E Ellis, K Mike Casey, and Geoffrey Hill. Chatgpt and python programming homework. *Decision Sciences Journal of Innovative Education*, 22(2):74–87, 2024.

[7] Timothy C Fanelli, Scott C Simons, and Sean Banerjee. A systematic framework for modernizing legacy application systems. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 678–682. IEEE, 2016.

[8] Mahdi Fahmideh Gholami, Farhad Daneshgar, Ghassan Beydoun, and Fethi Rabhi. Challenges in migrating legacy software systems to the cloud—an empirical study. *Information Systems*, 67:100–113, 2017.

[9] Ilana Hamilton and Brenna Swanston. Artificial intelligence in education Teachers' opinions on AI in the classroom, 2023. Visited April 20, 2024. URL: https://www.forbes.com/advisor/education/it-and-tech/artificial-intelligence-in-school/.

[10] Owen Hughes. This old programming language is much more important than you might expect. here's why, 2022. Visited March 30, 2024. URL: https://www.zdnet.com/article/programming-languages-how-much-cobol-code-is-out-there-the-answer-might-surprise-you/.

[11] Ronald J Kizior, Donald Carr, and Paul Halpern. Does cobol have a future? In *Proc. Inf. Syst. Educ. Conf*, volume 17. Citeseer, 2000.

[12] Stina Matthiesen and Pernille Bjørn. Why replacing legacy systems is so hard in global software development: An information infrastructure perspective. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 876–890, 2015.

[13] E Mollick, L Mollick, O Acar, and M Weiss. 4 simple ways to integrate "ai" into your class. *Harvard Business Publishing Education*, pages 4–10, 2024.

[14] Junaid Qadir. Engineering education in the era of chatgpt: Promise and pitfalls of generative ai for education. In *2023 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–9. IEEE, 2023.

[15] Muhammad Shidiq. The use of artificial intelligence-based chat-gpt and its challenges for the world of education; from the viewpoint of the development of creative writing skills. In *Proceeding of international conference on education, society and humanity*, volume 1, pages 353–357, 2023.

[16] Mark Sullivan. Cobol, a 60-year-old computer language, is in the covid-19 spotlight, 2020. Visited March 10, 2024. URL: `https://www.fastcompany.com/90488862/what-is-cobol`.

Replicating a Goal-Congruity Intervention*

Kathleen Isenegger and Colleen M. Lewis
Computer Science
University of Illinois Urbana-Champaign
Urbana. IL 61801
{kti3, colleenl}@illinois.edu

Abstract

Goal-congruity theory suggests that an intervention that increases students' perceptions of whether computing helps society could help broaden participation in computing. This randomized control study assigned 144 undergraduate participants to read one of two texts describing a computing project and to take a survey. The "communal text" emphasized communal affordances (i.e., opportunities to help others) of a project while the "non-communal text" did not. Results indicate that reading about the communal affordances of a computing project may not impact students' beliefs. Still, students may be more likely to pursue computing if they believe it to afford communal goals.

1 Introduction

Communities underrepresented in computing includes people who identify as women, Black/African American, Hispanic/Latinx/Latine, Native American, Native Alaskan, Native Hawaiian, Pacific Islander, and/or disabled [18].[1][2] Students from communities underrepresented in computing have contemporarily

[1] This study does not specifically consider how a person's identity around disability relates to other variables, instead focusing only on racial/ethnic and gender identities.

[2] In this paper, students who identify as members of racial/ethnic communities underrepresented in computing will be referred to as "Black, Native, and Latinx students"

and historically been excluded from and harmfully denied access to CS education [2]. This study focuses on one cause of underrepresentation: students' perceptions of whether a career in computing affords opportunities to accomplish communal goals.[3] Communal goals are aspirations of working with or for the benefit of others [10]. This study leverages goal-congruity theory as its theoretical framework. The theory explains that the desire to accomplish communal goals may deter individuals from pursuing science, technology, engineering, and mathematics (STEM) if they think these fields do not afford communal goals [10]. Relevant to broadening participation in computing (BPC), women are more likely than men to endorse communal goals [10, 9, 16, 14]. Further, Black, Native, and Latinx students are also more likely than white students to endorse communal goals [16, 14, 19].

This study replicates work by Brown et al. [6] testing an intervention that seeks to enhance students' perceptions of the communal affordances of STEM. The research questions for this study are: (1) to what extent does reading about the communal affordances of a computing project change students' perceptions of whether computing can help society?; (2) to what extent does reading about the communal affordances of a computing project increase students' motivation to pursue computing? Results differed from prior work [6]. The intervention had no effect on students' beliefs of the communal affordances of computing or students' motivation to pursue computing, but students' perceptions of the communal affordances of computing predicted their positivity towards computing research. Therefore, an intervention to increase students' perceptions of communal goals may still be helpful for BPC.

2 Previous Work: Communal Goals

Students generally perceive STEM occupations to afford communal goals less than other careers [10, 9]. However, in one study students reported believing computing *could* benefit society given its ubiquitous nature, but it was unclear if students thought a career in computing *actually* afforded communal goals [15]. For students who think STEM does not afford communal goals, goal-congruity theory describes that endorsing communal goals will cause students to see themselves as unfit for STEM [10, 9]. In accordance, researchers have found that communal-goal endorsement is negatively related to motivation to pursue science research for women [1], as well as students' academic belonging, motivation, and intentions to persist for Native American students majoring in STEM [19]. In one experiment, student's interest in STEM decreased when researchers experimentally activated (i.e., temporarily increased) student's communal-goal endorsement, pointing to a causal relationship [9].

[3] *Affords* refers to what an environment offers someone [12].

Goal-congruity theory suggests that students may be more motivated to pursue STEM if they believe that STEM *does* afford communal goals. Indeed, students' perceptions of the communal affordances of STEM are correlated with a sense of belonging in and intention to pursue STEM [5], and women's perceptions of the communal affordances of STEM are positively related to persistence in a STEM major [13]. Researchers report evidence for goal-congruity theory by showing that perceptions of the communal affordances of STEM moderates the relationship between students' communal-goal endorsement and their interest or belonging in STEM [16, 14, 4].

Students from communities underrepresented in computing are likely to strongly endorse communal goals [10, 9, 16, 14, 19] illuminating the implications of goal-congruity theory for BPC. Indeed, the creators of goal-congruity theory argue their work is particularly important for STEM, given the challenges of recruiting women into STEM careers [8]. Additionally, researchers found that for Black, Native, and Latinx undergraduate biomedical research assistants, a greater perception of the communal affordances of biomedical research predicted greater interest in the field [21].

Research shows that communal STEM experiences positively impact students' perceptions of the communal affordances of STEM [20, 3]. Another well-studied intervention asked students to read a text that framed the typical day of a scientist as highly collaborative, compared to a text that framed it as highly independent. [9, 7, 4]. Students who read the highly collaborative version perceived STEM to afford more communal goals [4], and women who read it reported a higher positivity towards STEM [9]. Like us, researchers have examined interventions where students read about the communal affordances of STEM. One study found that learning about the communal affordances of several STEM careers was positively related to students' perceptions of the communal affordances of STEM [20]. Meanwhile, the intervention replicated in this study takes minimal time and money by asking students to read a description of a single STEM project that emphasizes communal affordances [6]. To the authors' knowledge, this is the first experiment on changing perceptions of the communal goal affordances of computing, and the first replication of the work by Brown et al. [6].

3 Methods

Participants were assigned using a between-subjects design and simple randomization to read one of two descriptions of the same research project. After reading their assigned version, participants completed a survey including self-reporting their gender identity[4], ethnicity, race, the highest level of education of their parent(s)/guardian(s), and if they had previously taken a computing course. This study's methods and the authors' hypotheses were pre-registered with the Open Science Framework (OSF), where a copy of the full survey and intervention is available [11].

3.1 Intervention Text

The communal and non-communal texts were created to convey information about a recent and ongoing computing research project that has clear positive societal impacts [17]. In the communal version, sentences end with words related to helping the environment and fighting world hunger. Conversely, sentences in the non-communal version end in ideas focused on efficiency and saving money.

3.2 Participants

Participants were 144 undergraduate students recruited from an introductory computing course for non-computing majors at a large public university during the fall 2022 and spring 2023 terms (women: $\mathrm{N_T}$ (i.e. N of treatment) = 43, $\mathrm{N_C}$ (i.e. N of control) = 40; men: $\mathrm{N_T}$ = 34, $\mathrm{N_C}$ = 26; first-generation college students: $\mathrm{N_T}$ = 24, $\mathrm{N_C}$ = 20; previously took a CS course: $\mathrm{N_T}$ = 50, $\mathrm{N_C}$ = 44; Hispanic/Latinx: $\mathrm{N_T}$ = 15, $\mathrm{N_C}$ = 6; Arab/Middle Eastern: $\mathrm{N_T}$ = 2, $\mathrm{N_C}$ = 1; Caucasian/European/white: $\mathrm{N_T}$ = 28, $\mathrm{N_C}$ = 32; Asian: $\mathrm{N_T}$ = 42, $\mathrm{N_C}$ = 29; African American/Black: $\mathrm{N_T}$ = 2, $\mathrm{N_C}$ = 2; American Indian/ Alaska Native or Indigenous or First Nations or Native Hawai'ian/Pacific Islander: $\mathrm{N_T}$ = 1, $\mathrm{N_C}$ = 0; students of another race: $\mathrm{N_T}$ = 5, $\mathrm{N_C}$ = 1). Students were incentivized to participate with extra credit in their course. All students who volunteered were included. Participants were recruited through in-person and course webpage announcements and emailed a link to participate in the study asynchronously online, consistent with the study by Brown et al. [6]. The purpose of the current study is to determine if the intervention is effective at increasing students' perceptions of the communal affordances of CS. Neither students' identity, nor their endorsement of communal goals, should change the extent to which the intervention increases their perception of the communal affordances of computing. Therefore, it is reasonable to include all students in this study, and not only students from communities underrepresented in CS.

[4]The survey asked students, "What is your gender identity?" and they could select: man, woman, genderqueer/non-conforming/non-binary, agender, or something else (with a text box to specify).

3.3 Measures

The authors created four measures by averaging students' survey item responses. If a student did not answer all the items for a measure, the average was calculated with the items that were answered. All students answered at least one question for each measure, meaning none were dropped from the analysis. Little's Test of Missing Completely at Random gave a p-value of 0.42. Therefore, the data missing from the dataset can be assumed to indeed be random. Each measure was standardized with a mean of zero and standard deviation of one for the regression analyses. Most questions asked students to answer on a scale from one to seven, except three of the seven questions for the future career motivation measure, which were on a scale from one to five. For all questions, a greater score indicates more a more positive response (i.e. more agreement with the survey item). The measures are summarized below, with details available on OSF [11].

Students' beliefs about the extent to which computing affords communal goals were assessed with five items, such as "How much does the research that you just read about fulfill the goal of serving the community?" (Cronbach's α= 0.88). Agentic affordances of computing refer to opportunities in computing for self-promotion and self-fulfillment. Students' beliefs were measured with five items, such as "How much does the research that you just read about fulfill the goal of power?" (Cronbach's α=0.83). Students' positivity towards the research project they read about was assessed with four items, such as "What is your impression of the research that you read about?" (Cronbach's α=0.77). Students' motivation to pursue a career in computing was assessed with seven items, such as "How likely would you be to look into joining a laboratory conducting similar research in the future?" (Cronbach's α = 0.92).

4 Data Analysis

The analysis includes six regression models, all with a statistical significance threshold set to 0.05. The four outcomes of interest are: **Outcome 1** - perception of the communal affordances of computing; **Outcome 2** - perception of the agentic affordances of computing; **Outcome 3** - positivity towards computing research; **Outcome 4** - motivation to pursue a career in computing. Models 1-4 determine the effects of reading the communal vs. non-communal text (treatment) on all four outcomes and had the following form:

$$Y_s = \beta_0 + \beta_1 CommText_s + \varepsilon_s \tag{1}$$

Y_s is an outcome variable for student s (outcome 1-4 listed above), $CommText_s$ is 1 if student s read the communal text and 0 if they read the non-communal

Table 1: Regression Results

	Outcome 1 Comm. Afford.	Outcome 2 Agentic Afford.	Outcome 3 Positivity	Outcome 4 Motivation	Outcome 5 Positivity	Outcome 6 Motivation
Comm. Text	0.001 (0.168)	-0.251 (0.167)	-0.074 (0.168)	0.079 (0.768)	-0.074 (0.123)	0.079 (0.165)
Comm. Afford	-	-	-	-	0.669*** (0.100)	0.227+ (0.135)
Interaction	-	-	-	-	0.029 (0.127)	-0.037 (0.171)
Constant	-0.000 (0.124)	0.136 (0.123)	0.040 (0.123)	-0.043 (0.123)	0.040 (0.090)	-0.043 (0.122)
N	144	144	144	144	144	144
Adj. R-Sq.	-0.007	0.009	-0.006	-0.006	0.462	0.023

Note: Values are mean centered and normalized. Standard errors are in parentheses. +p<.10, * p<.05, ** p<.01, *** p<.001

text, and ε_s is the per-student error. The authors hypothesized that the communal vs. non-communal text (treatment) will increase students' perceptions of the communal affordances of computing (outcome 1), positivity towards computing (outcome 3), and motivation to pursue a career in computing (outcome 4), but not students' perceptions of the agentic affordances of computing (outcome 2).[5] Models 5 and 6 sought to determine if there are moderating effects between measures and had the following form:

$$Y_s = \beta_0 + \beta_1 CommText_s + \beta_2 CommAfford_s + \beta 3 CommText_s * CommAfford_s + \varepsilon_s \quad (2)$$

As in Models 1-4, Y_s is a particular outcome variable for student s, $CommText_s$ is 1 if student s read the communal text and 0 if they read the non-communal text and ε_s is the error. Model 5 predicts positivity towards computing research (outcome 3) and Model 6 predicts motivation to pursue a career in computing (outcome 4). $CommAfford_s$ is the perception of the communal goal affordances of computing reported by student s, $CommText_s * CommAfford_s$ is an interaction term that is the perception of the communal goal affordances of computing reported by student s if they read the communal text, and 0 if they read the non-communal text. If the interaction term is statistically significant, showing that a moderating effect is present, then the relationship between students' perceptions of communal affordances of computing (outcome 1) and students' positivity towards computing research (outcome 3) (or motivation to pursue a career in computing (outcome 4)) depends on whether they read the communal text (treatment). The authors hypothesized that a moderating effect would be present in both models.

5 Results

The students in the communal-text vs. non-communal text groups reported an (unstandardized) average (standard deviations in parentheses) for each measures of interest: communal affordances perception was 5.4 (0.1) vs. 5.4 (0.1)

[5] Hypotheses were pre-registered [11]

out of 7 (outcome 1), agentic affordances perception was 4.6 (0.1) vs. 4.9 (0.1) out of 7 (outcome 2), positivity towards computing was 4.8 (0.1) vs. 4.9 (0.1) out of 7 (outcome 3), and future career motivation was 3.2 (0.1) vs. 3.2 (0.2) out of 6.14 (outcome 4).

Results from Models 1 and 3-6 proved contrary to the hypotheses. There was no evidence that reading the communal text (treatment) predicted students' perceptions of the communal affordances of computing (outcome 1); the coefficient was nearly 0 (0.001). Likewise, neither coefficient of interest in Models 3 nor 4 was statistically significant; the relationships were both small and positive (< 0.1), with standard deviations of 0.168 and 0.768 respectively. Therefore, it cannot be concluded if there is a relationship between reading the communal text and either positivity towards computing research or motivation to pursue a computing career. There was no evidence of a moderating effect in Model 5 or 6 as the coefficients for the interaction terms were small and not statistically significant (< 0.05). As hypothesized, Model 2 shows no statistically significant relationship between reading the communal text (treatment) and students' perceptions of the agentic affordances of computing (outcome 2).

Model 5 revealed an unhypothesized significant positive effect of perceiving communal affordances in computing (outcome 1) on positivity towards computing research (outcome 3). Students who reported one standard deviation higher in their perception of the communal affordances of computing (outcome 1) are predicted to have 0.669 standard deviations higher positivity towards computing research (outcome 3). Additionally, there was a positive, but non-significant, relationship between students' perceptions of communal affordances in computing (outcome 6) and their motivation to pursue a career in computing (outcome 4). Post-hoc analyses revealed that students who read the communal text had significantly lower responses for only one item, "How much does the research that you just read about fulfill the goal of financial rewards?".

6 Discussion

Contrary to most hypotheses, there was no evidence that reading the communal-text (treatment) predicted students' perception of communal affordances (outcome 1), student's positivity towards computing research (outcome 3), nor students' motivation to pursue computing (outcome 4). As such, it is unsurprising that there was no evidence that students' perceptions of communal affordances is a moderating variable. There could be multiple reasons why this study did not replicate the results of Brown et al. [6]. One explanation is that the participants may have already believed that computing *can* help society [15], possibly eliciting a ceiling effect. This may have been the case; previous work asked students "can computing help society?" and found average ratings of 3.64 and 3.61 (5-point Likert scale) [16, 14]. Additionally, the impact found by Brown et al. [6] may have resulted from two additional differences in their communal and non-communal texts. First, their non-communal text has more technical language than their communal text (e.g., "microfabricate", "embedded sensors", "robotics"). Second, the non-communal text by Brown et al. [6] discusses no goal affordances of the project, while the current study's non-communal text includes affordances of the research project that are not communal (e.g., efficiency, saving money) to isolate the communal aspect as the only difference. A limitation that may have impacted the answer to the second research question is the pattern found in previous work that increasing white men's perception of the communal affordances of computing may not increase their interest in computing. The analysis lacked gender and race, and therefore did not control for this potential impact on the results.

Regardless, this study still yielded important evidence for understanding the impact of a communal-goal affordance intervention. Aligned with findings by Brown et al. [6], there was a positive correlation between perception of communal affordances and positivity towards computing research. To be clear, this did not test the main goal-congruity theory hypothesis (that is not part of this study), but rather supports a key argument from goal-congruity theory: increasing student's beliefs of the communal affordances of computing is beneficial, especially for students who highly endorse communal goals. As such, further research to understand how to increase students' perceptions of the communal affordances of computing is imperative. This work also speaks to the replication crisis within science. The authors pre-registered the study with the Open Science Framework to help ensure the methodology was transparent and trustworthy. Therefore, while the results for the main research questions were not statistically significant, this study still advances knowledge on goal-congruity theory and is valuable for the progression of science. It is still an open question if the results from the work by Brown et al. [6] can be replicated. Further work is needed to understand the limitations of this intervention towards designing an effective tool for influencing students' perception of computing's communal goal affordances.

7 Conclusion

This work examined if a communal-text intervention increased student's beliefs of the communal affordances of computing and, in turn, their positivity towards and motivation to pursue computing. This study replicated previous work by Brown et al. [6] with some changes, though the findings were inconsistent with this prior work. There was no evidence that the communal-text intervention changed students' perceptions of whether computing can help society or increased students' interest in computing. Consistent with previous work, the results show a positive relationship between students' perceptions of the communal affordances of computing and students' positivity towards computing. Therefore, understanding how to change students' perceptions of the communal affordances of computing may be important for BPC. Additional research is needed to understand what interventions can increase students' perceptions that computing has communal affordances.

8 Acknowledgements

This work is partially funded by the National Science Foundation grant (1821136).

References

[1] Jill Allen et al. "Nevertheless, she persisted (in science research): Enhancing women students' science research motivation and belonging through communal goals". In: *Social Psychology of Education* 24.4 (2021), pp. 939–964.

[2] William Aspray. *Women and underrepresented minorities in computing.* Vol. 10. Springer, 2016.

[3] Aimee L Belanger, Amanda B Diekman, and Mia Steinberg. "Leveraging communal experiences in the curriculum: Increasing interest in pursuing engineering by changing stereotypic expectations". In: *Journal of Applied Social Psychology* 47.6 (2017), pp. 305–319.

[4] Aimee L Belanger et al. "Putting belonging in context: Communal affordances signal belonging in STEM". In: *Personality and Social Psychology Bulletin* 46.8 (2020), pp. 1186–1204.

[5] Ashley Bonilla et al. "Diversifying STEM: Communal goal mismatch predicts student intentions". In: *Social Psychology of Education* 26.2 (2023), pp. 293–308.

[6] Elizabeth R Brown et al. "From bench to bedside: A communal utility value intervention to enhance students' biomedical science motivation." In: *Journal of Educational Psychology* 107.4 (2015), p. 1116.

[7] Emily K Clark, Melissa A Fuesting, and Amanda B Diekman. "Enhancing interest in science: Exemplars as cues to communal affordances of science". In: *Journal of Applied Social Psychology* 46.11 (2016), pp. 641–654.

[8] Amanda B Diekman et al. “A goal congruity model of role entry, engagement, and exit: Understanding communal goal processes in STEM gender gaps”. In: *Personality and Social Psychology Review* 21.2 (2017), pp. 142–175.

[9] Amanda B Diekman et al. “Malleability in communal goals and beliefs influences attraction to stem careers: evidence for a goal congruity perspective.” In: *Journal of Personality and Social Psychology* 101.5 (2011), p. 902.

[10] Amanda B Diekman et al. “Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers”. In: *Psychological Science* 21.8 (2010), pp. 1051–1057.

[11] Open Science Foundation. *Pre-Registration.* 2023. URL: `https://osf.io/735hn/?view_only=828c7b8dc46f4885930724123557d2f8`.

[12] James J Gibson. “The theory of affordances”. In: *Hilldale, USA* 1.2 (1977), pp. 67–82.

[13] Heather L Henderson et al. “Seeking congruity for communal and agentic goals: a longitudinal examination of US college women’s persistence in STEM”. In: *Social Psychology of Education* 25.2 (2022), pp. 649–674.

[14] Kathleen Isenegger et al. “Goal-Congruity Theory Predicts Students’ Sense of Belonging in Computing Across Racial/Ethnic Groups”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1.* 2023, pp. 1069–1075.

[15] Kathleen Isenegger et al. “Understanding and Expanding College Students’ Perceptions of Computing’s Social Impact”. In: *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT).* IEEE. 2021, pp. 1–10.

[16] Colleen Lewis et al. “Alignment of goals and perceptions of computing predicts students’ sense of belonging in computing”. In: *Proceedings of the 2019 ACM Conference on International Computing Education Research.* 2019, pp. 11–19.

[17] Allison Linn. *Welcome to the Invisible Revolution.* `https://news.microsoft.com/stories/invisible-revolution/`. 2016.

[18] National Center for Science and Engineering Statistics. *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2021.* 2021. URL: `https://ncses.nsf.gov/wmpd`.

[19] Jessi L Smith et al. “Giving back or giving up: Native American student experiences in science and engineering.” In: *Cultural Diversity and Ethnic Minority Psychology* 20.3 (2014), p. 413.

[20] Mia Steinberg and Amanda B Diekman. “Elevating positivity toward STEM pathways through communal experience: The key role of beliefs that STEM affords other-oriented goals”. In: *Analyses of Social Issues and Public Policy* 17.1 (2017), pp. 235–261.

[21] Dustin B Thoman et al. “The role of altruistic values in motivating underrepresented minority students for biomedicine”. In: *BioScience* 65.2 (2015), pp. 183–188.

Lawyer Up! Joint Introductory Computer Science and Law Courses*

Shannon Beck[1], Timothy Goines[2], Jeffrey Biller[2]
[1]Computer and Cyber Sciences
[2]Law Department
United States Air Force Academy
US Air Force Academy, CO 80840
{shannon.beck, timothy.goines, jeffrey.biller}@afacademy.af.edu

Abstract

Courses typically focus on a single discipline, limiting exposure to cross-disciplinary topics. To address this, we designed an interdisciplinary two-course set within our undergraduate Honors program: an introductory computing course and an introductory law course. This initiative breaks down barriers between computing and law, enabling students to integrate both disciplines and foster a comprehensive understanding of complex issues. The Computer Science (CS) course covers introductory programming and principles, while the Law course surveys American Law and basic legal reasoning. Initially, the topics are independent but they strongly converge in the term's second half. Students explore the intersection of CS and Law through discussions, debates, guest speakers, and a cross-disciplinary project. Conducted for two years at the United States Air Force Academy (USAFA), this report shares our curriculum and experiences, aiming to inspire wider adoption of our inter-departmental model.

1 Introduction

We introduce a dual-course set designed and offered for two years at our undergraduate-only institution of roughly 4,000 students. An introductory Computer Science (CS) course and introductory Law course are held as double period set of back-to-back classes. This interdisciplinary initiative strives to break down artificial barriers between computing and law, allowing students to integrate topics from both disciplines. The inclusion of diverse subject areas is intentional. The goal is to foster the development of critical thinking and problem-solving skills that extend beyond subject-specific knowledge silos. We expose students to higher-level abstract mental models so that they can create, evaluate, analyze, and apply these skills to solve multi-domain problems.

This paper outlines our strategy to develop and teach these interdisciplinary courses in our Honors (aka Scholars) program. Section 2 provides background, followed by course outlines for CS (3.1) and Law (3.2). Interdisciplinary emphasis is discussed (3.3), followed by experiences (4), including student feedback (4.3), recommendations (5), concluding with our final remarks (6).

2 Background

Our academic leadership tasked us to move beyond a domain-limited approach and develop a set of classes aimed at educating students to analyze complex issues and explore the intersections across multiple disciplines. The goal is to equip students with a broader perspective, enabling them to reason in ways that transcend traditional academic silos.

We arrived at the joint course guidelines, used to develop the courses, through discussion between departments and leadership. They are:

- First-year students
- Not major specific
- Honors (Scholars) program students
- Double-period (back-to-back) classes
- Students must take both courses or neither course
- Meets core/general education (GenEd) requirements of each original course

3 Course Outlines

These two, 3-credit hour courses meet for 40 lessons, each with a 53-minute period (106 min total). Part of our institution's Honors program, selected students participate in Honors designated courses. Both of these courses streamline their individual mandatory learning objectives, operating fairly independently in the first half of the semester. This deep dive into the individual subject matter is necessary to create the knowledge baseline. In the latter half of the semester, the courses dive into interdisciplinary topics, including Machine Learning (ML)/AI, quantum computing, cybersecurity, and ethics. Table 1 shows lesson topics, with additional detail in the following subsections.

Lsn	Computer Science Topics	Law Topics
1	Binary & Numeric Representation	Course Introduction
2	Von Neumann Architecture	Why law? and the US Government
3	Boolean Logic	The American Judicial System, Intro to Legal Reading
4	Algorithms & Algorithmic Thinking	Reading/Constructing Legal Arguments
5	Intro to Python and Pseudocode	Criminal Law Fundamentals
6	Python: Input and output	Homicide Offenses
7	Python: String variables and lists	Uniquely Military Crimes
8	Lab: Python: String variables and lists	Search and Seizure
9	Python: Boolean logic/branching	Search and Seizure (cont)
10	Lab: Python: Boolean logic	Compelled Self-Incrimination
11	Python: Loops for and while	Compelled Self-Incrimination (2)
12	Python: File input and output	Oral Arguments
13	Python: Functions & procedures	Oral Arguments
14	Lab: Python: Functions	Oral Arguments
15*	Speaker: Cybersecurity and Career Opportunities	
Programming Due Process in Python Assignment		
16	Excel Spreadsheets	Free Speech
17	Excel Team Project	Free Speech (Cont)
18	Exam	International Law & Jus ad Bellum
19	Machine Learning	The Law of War: Jus in Bello
Joint Course Emphasis Begins		
Joint Topic: Artificial Intelligence (AI)		
20	AI Overview	The Law of War: Jus in Bello (2)
21	AI and ML limitations	Intro to AI, Law, and Policy
22*	Speaker: AI/ML	
23	Lab: AI & ML	Using Research Tools
24*	Speaker: AI/ML Ethics	
25	Privacy and LLMs	LAWS: Weapons Reviews, Command Responsibility, & AI Decision Making
26*	Student-Led Presentations: AI	
Joint Topic: Quantum Computing		
27	Cryptography	IP Law and National Security
28*	Speaker: Quantum Computing	
29*	Student-Led Presentations: Quantum	
Joint Topic: Cyberspace		
30	Networking Overview	Domestic Cyber Law (CFAA)
31	Cybersecurity Awareness and Hygiene	International Law related to Cyber Operations

Lsn	Computer Science Topics	Law Topics (Con't)
32	Computer Networking and Cyber-security	International Law related to Cyber Operations (2)
33*	Speaker: Cybersecurity and Policy	
34*	Student-Led Presentations: Cybersecurity	
35*	Cybersecurity Case Study: NotPetya	
36*	Career Panel: Cyber Warfare	
37	Lab: Hacking a SCADA village	Domestic Cyber Law (ECPA)
Joint Law and CS Paper Presentations		
38*	Final project: Student Presentations	
39*	Final project: Student Presentations	
40*	Final project: Student Presentations	

Table 1: 40 Lesson Plan Curriculum of most recent offering. An asterisk (*) indicates joint class periods (106 minutes) used.

3.1 Computer Science

Our "traditional" introductory CS 110 course is a non-Honors, non-combined course. "Introduction to Computer Science and Cyber Operations" is a combination of Python programming (∼60% of the lessons, and ∼75% of the grade) and ∼40% survey class including artificial intelligence (AI), computer networks, cybersecurity, and ethics.

This interdisciplinary Honors introductory course (CS 110S) differs significantly from our "traditional" CS 110. We reduce the Python programming (30% of the lessons, and ∼40% of the grade) and computing background. The intersection of law and technology comes in the latter half of the course. A final paper and presentation replaces the final exam for 25% of the grade.

3.1.1 Computer Science Assignments

The summative assessments for the first half of the CS course have a strong technical focus. Most points come from: program submissions for each block; two programming projects–one for Python and one for Excel; and an exam.

The points from the second half of the course are generated through reading and responding to their readings through writing [1][8], speaking, and actively discussing the readings. Several writing assignments include researching recent authoritative papers and providing annotated bibliographies for the intersectional topics (i.e. quantum computing).

The distinct shift midway through the course from very CS hands-on to reading and evaluating the interdisciplinary intersection of law, policy and technology (see the curriculum outline in Table 1) was concerning for some students, which we address in the CS Evaluation Feedback in Section 4.3.1.

3.2 Law

Similarly, the "traditional" Law 220 course as taught in our institution is a non-Honors, non-combined course. "Law for Air Force Officers" is a survey of United States (US) law with a focus on criminal law, due process, individual rights, and the law of armed conflict. The course is meant to provide an introduction to several aspects of US law while focusing on the development of critical thinking skills. Assignments in the "traditional" Law 220 include legal briefs, oral arguments, and other legal analyses assignments. Since the course is taught by a number of professors, the amount of class time given to each topic and the assignments vary widely; however, most of the offerings for the course include a final exam worth 25% of the final grade.

The Law 220S interdisciplinary offering is an adaptation of its Honors equivalent. This 200-level course is generally taken in the second year. In order to pair it with the CS course, this variant is taught a year earlier, which we conclude increases difficulty for first-year students, as indicated in Section 4.3.

The first half of the interdisciplinary offering is essentially an accelerated standard introductory law course laying the foundation for the application of law and ethics to emerging technologies. The interdisciplinary Law course begins with an introduction to the nature and purpose of the law, with a focus on the Enlightenment thinkers who provided the philosophical underpinnings of the US legal system. This prelude will prove vital later in the course when contemplating the goals behind legal and ethical approaches to modern technological advances in quantum, artificial intelligence, and autonomy. The course then takes a topical approach to US law with an introduction to due process, privacy, criminal prohibitions, speech, and equal protection, and generally concludes with a brief introduction to international law. Along with the typical court cases and readings provided for the "traditional" law course, readings exploring the US approach to technological developments are included. Issues presented include digital surveillance, biotechnology, and encryption. Additionally, this portion of the course focuses on thinking of the law as an algorithm through which legal determinations can be made.

3.2.1 Law Assignments

Evaluations in the first half include essays related to the nature and purpose of the law, a series of written algorithms reflecting the legal analysis, legal briefs demonstrating legal analysis, and an oral argument or a scenario-based exam on the application of international law. A culminating exercise of the first half of the semester is writing a Python program to express a due process legal analysis based on the input set of facts. Along with the program, students also complete a notional "walk-through" of their program to explain the legal analysis occurring within the program.

3.3 Law and Computer Science Intersection and Intensive Focus

As discussed, there is limited overlap in the two courses for the first half. One notable (and fun) exception is the Law "Due Process Coding" assignment.

The second half becomes a parallel offering of joint topics between the courses. Some of the double periods are joined for in-depth learning, including for student presentations and guest speakers (marked with * in Table 1). This double period and close coordination enables flexibility.

The general layout follows a pattern of an introduction to the emerging technology, a basic rendering of how the technology works, and a discussion of strengths and weaknesses of the technology. The students then explore a set of readings offering legal, policy, and ethical critiques and solutions to the problems posed by the technology. An emphasis is placed on both technical and legal aspects to maximize the benefits of the technology while simultaneously mitigating the downsides. Of particular focus is how the law could compensate or account for technological shortcomings potentially resulting in violations of fundamental human rights. For example, how equal protection laws might address potential bias in artificial intelligence systems related to policing, hiring, or health care. Additional topics include accountability mechanisms under international law that address the use of lethal autonomous weapon systems in armed conflicts and the interaction between domestic national security laws and private development of quantum computing systems.

3.3.1 Joint Assignments

The joint course portion is evaluated through two primary mechanisms: (1) in-class debates/presentations related to each special topic and (2) a lengthy research paper that combines a technical explanation of a student-chosen technology with an examination of the associated legal and ethical issues.

For the first three offerings, students were asked to debate. Propositions related to the intersectional topics including predictive policing and AI. Debating students submitted a write-up of their objectives, position, position's importance, and references. Non-debating students wrote an analysis including which team had the best argument and why; how they would have improved the argument of for the team with the lesser argument, and resources. The joint assignment had separate rubrics and grading for each course.

We found that freshman had difficulty in understanding and holding a true debate vs. fact sharing. The last offering had student teams lead discussions of 2-3 papers they selected for a special topic (Lessons 26, 28, and 34).

The final assessment is a joint paper (10-12 pages) and presentation in lieu of a final exam for both courses. Students are expected to produce a single paper that should introduce and discuss the technical aspects of a student-chosen technology, and then cross over to discuss the policy and legal implications for that technology. Student topics have included autonomous drone swarms for military use; neuromorphic computing to replace human judges; Fifth Amendment concerns regarding emotional recognition technology; and medical technologies (FMRI, cancer detection) and privacy concerns. The Law class employs scaffolding assignments to ensure appropriate steps toward the completion of a scholarly paper (e.g., abstract, bibliography, outline, and draft paper). The final paper and presentation accounts for 25% of their course grade in both CS and Law. Presentations are graded jointly, papers independently.

4 Course Experiences

This combination of courses have been offered four times across two academic years. While generally following the initial model, each semester brings refinement with the goal of increasing integration. Also, the joint topics chosen (see Table 1) have changed over time in an effort to enhance learning outcomes by finding subjects that are both appropriate to the students' educational level and ideal for blending hard science and social science.

4.1 Challenges

One unexpected challenge is scheduling. Due to the joint coverage and double-period use, students must take both or neither. Our student scheduling system does not enable a way to specify this. As a result, students have been placed into one with a schedule conflict for the other course or students who have already taken CS 110 are placed into this interdisciplinary Law 220 offering.

Another difficulty is striking a good pedagogical balance, ensuring depth in each discipline without overwhelming first-year students while maintaining the GenEd learning objectives. This requires adaptivity and creativity, asking for intellectual stretching by the students and instructors.

4.2 Class Size

Due to the hallmark of intensive discussion and writing for Honors courses, scalability is limited. Small class sizes encourage conversations, questions, and overall interactivity. Enrollment caps for the introductory CS and Law sections are shown in Table 2. Our average enrollment size across the four offerings has been 12.25, mainly due to starting with 14 and losing students due to the mis-enrollment previously discussed.

	Traditional	Honors/Scholars
CS	24	14
Law	20	14

Table 2: Section Enrollment Caps for Respective Courses

4.3 Student Feedback

We summarize student feedback below that is submitted through the institution-administered course evaluations. Instructors receive anonymous quantitative and qualitative feedback. We report upon feedback for both courses (rated individually) from the first three concluded offerings. Figure 1 shows the ratings to the prompt "Overall, this course is:" for both courses.

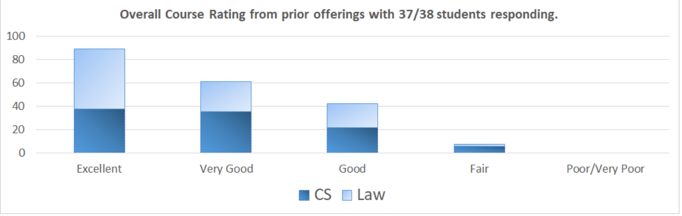


Figure 1: Overall Course Rating from three previous offerings.

4.3.1 Computer Science Evaluation Feedback

Student feedback has generally been positive. Students enjoyed the CS course and found it engaging, rating the course high, shown in Figure 1.

When asked for improvements, one strong theme emerged: more programming. This is likely a result of the enrollment selection process. In addition to the goal of increasing diversity, students with prior programming experience (based on admissions data) were preferred.

Conversely, every offering has had students with no prior programming experience. Leveling the playing field for students has led to using differentiated learning approaches. Programming "extensions" are offered in each coding block, expanding skills for more advanced students. Once one student completes the graded exercises, the 1-2 extensions per block become accessible, with the possibility of 0-3 extra credit points. This removes pressure to complete the advanced exercises, but allows for continued skill growth if desired.

Another concern voiced is the change of the pace and style of teaching that occurs halfway through the semester. The change from Python programming with exact results to discussion and presentations can be jarring. This is a consistent comment, often from the few who request more programming.

4.3.2 Law Evaluation Feedback

Similarly, student feedback for the Law course is very positive, with over 50% rating it "Excellent" (see Figure 1). When asked for improvements, a repeated comment is on the pace of the first half. In the "traditional" Law course, topics are given substantially more time. The extra time permits more assignments, giving more opportunity to develop and refine students' legal writing and analysis. The interdisciplinary course must prioritize integrating with the CS class later on, leading to feedback centered on the limited time available for skill development.

This is a legitimate concern for any skills-based course. As a result, each iteration aims to increase the opportunity to practice these skills in- and outside of class, rather than as an assignment. E.g. in our third iteration, we started requiring students to independently produce "algorithms" they bring to class based on the law readings. These algorithms encourage students to think of legal analysis as a step-by-step method through which legal determinations can be made. While a "traditional" Law class would utilize these algorithms in some sense, they are less intentional about practicing their development independently outside of class and reflecting on them during class. In our experience, these algorithms allow us to more effectively use the in-class time to further develop and refine their analytical skills.

Similar to the CS course, the other concern expressed by students is the change of pace and style of teaching that occurs halfway through. Many expressed how developing their legal writing and analysis skills was more difficult given the change from court cases to scholarly articles. This change is intentional, as the course moves from looking at the law in response to actions (violations of the law) to a proactive approach to the law (the development of new technologies and how best to regulate them). One goal of the latter half of the semester is to allow students to use the legal skills developed in the first half to inform their thoughts in the latter half. This could be more clear and we have incorporated this information into the most recent offering.

5 Course Design Recommendations

Based on the offerings we have taught, we have a several recommendations.

Adapt for your institution: Our enrollment is steady and could increase if we raised the enrollment caps. Students in the Honors program must take ten Honors classes to graduate as a designated scholar. These fill two of the ten slots while meeting specific GenEd requirements. At another school, these are likely not required courses for all students. One consideration is to design a version that fits into a major instead of GenEd requirements. This would likely move from being a first-year class and count as an elective for one or more majors (Cyber Sci, Computer Sci, or Law).

Consider different disciplines: Another idea is to leverage local expertise and collaborations. We were fortunate to have professors in Computer Science and Law with interdisciplinary experience. However, other versions could use different disciplines, modeling after our interactions and intersections, combing with history, English and language arts [4], poetry [5, 6], music [7], social sciences [2], ethics [3], medicine, or other disciplines. This requires subject matter experts in each area, willing to flex and meet the other discipline in the middle to build an effective, informative, learning-centered class.

Be flexible: Teaching at the intersection of multiple disciplines is never as clean and neat as presenting a single subject. Second, the fast-paced evolution of new rulings, laws, and emerging technology makes course content dynamic. Readings and assignments should be updated for each offering.

Communicate: Effective communication between disciplines is vital. One strategy to bridge gaps and facilitate meaningful discussion is for faculty to attend the other's interdisciplinary classes.

Be patient: Interdisciplinary courses are not as prevalent. As a result, successfully combining two seemingly unrelated subjects is challenging. While we believe we have achieved a level of success, we know there is room for improvement. There is nothing wrong with a bit of trial and error as you figure out what works and what does not.

6 Conclusion

In summary, we share our experiences creating interdisciplinary Honors courses in Computer Science and Law at our undergraduate institution. Our aim is to break down barriers, empower students to integrate knowledge, and foster a holistic understanding of complex topics which we believe we have successfully achieved for the majority of students.

Our suggestions for embracing similar interdisciplinary, paired courses include customizing the approach to fit your institution's needs, leverage local subject matter experts, consider integrating diverse disciplines, maintain flexibility and communication throughout the process, and practice patience.

The structure of this innovative dual-course goes beyond our previous offerings as creators and exceeds our institution's traditional offerings. May our endeavors inspire fresh ideas, encouraging you to embark on creating an interdisciplinary course that suits both faculty and students at your institution.

References

[1] Theresa Beaubouef. "Why computer science students need language". en. In: *ACM SIGCSE Bulletin* 35.4 (Dec. 2003), pp. 51–54. ISSN: 0097-8418. URL: https://dl.acm.org/doi/10.1145/960492.960525 (visited on 01/20/2024).

[2] Briana Bettin. "Challenges, Choice, & Change: Experiences and Reflections from the First Semester of a Technology and Human Futures Course". en. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. Toronto ON Canada: ACM, Mar. 2023, pp. 235–241. ISBN: 978-1-4503-9431-4. URL: https://dl.acm.org/doi/10.1145/3545945.3569872 (visited on 01/20/2024).

[3] Trystan S. Goetze. "Integrating Ethics into Computer Science Education: Multi-, Inter-, and Transdisciplinary Approaches". en. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. Toronto ON Canada: ACM, Mar. 2023, pp. 645–651. URL: https://dl.acm.org/doi/10.1145/3545945.3569792 (visited on 01/11/2024).

[4] Mark Guzdial. "Scaffolding to Support Humanities Students Programming in a Human Language Context". en. In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2*. Turku Finland: ACM, June 2023, pp. 579–580. ISBN: 9798400701399. URL: https://dl.acm.org/doi/10.1145/3587103.3594157 (visited on 01/18/2024).

[5] LeBlanc, Mark D. "Computing and the Digital Humanities". In: *NCWIT Teaching Paper: National Center for Women & Information Technology*. NCWIT, May 2016. URL: http://www.engage-csedu.org/sites/default/files/LeBlanc_EngageCSEdu-TeachingPaper.pdf.

[6] LeBlanc, Mark D. *Computing for Poets*. URL: https://wheatoncollege.edu/academics/special-projects-initiatives/lexomics/computing-poets/ (visited on 01/17/2024).

[7] Douglas Lusa Krug. "Code Beats - Teaching Computer Programming via Hip Hop Beats". en. In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2*. Dublin Ireland: ACM, July 2022, pp. 646–647. ISBN: 978-1-4503-9200-6. DOI: 10.1145/3502717.3532111. URL: https://dl.acm.org/doi/10.1145/3502717.3532111 (visited on 01/18/2024).

[8] Lisa Zhang et al. "Embedding and Scaling Writing Instruction Across First- and Second-Year Computer Science Courses". en. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. Toronto ON Canada: ACM, Mar. 2023, pp. 610–616. URL: https://dl.acm.org/doi/10.1145/3545945.3569729 (visited on 01/20/2024).

ChatGPT as an Assembly Language Interpreter for Computing Education*

Fei Zuo[1], Cody Tompkins[1], Gang Qian[1],
Junghwan Rhee[1], Xianshan Qu[1] and Bokai Yang[2]
[1]Department of Computer Science
University of Central Oklahoma, Edmond, OK 73034
{fzuo, ctompkins6, gqian, jrhee2, xqu1}@uco.edu
[2]Department of Computer Science
University of Wisconsin-Eau Claire, Eau Claire, WI 54702
yangboka@uwec.edu

Abstract

Assembly language is a low-level programming language useful for a number of important computing areas, such as hardware and embedded systems programming, computer architecture, reverse engineering, and malware analysis. In recent years, generative AI, enhanced by GPT technology, has been widely adopted in the IT industry as well as computing education. However, little work has been done to investigate the applicability of GPT to teaching assembly language. In this paper, we fill in the gap by providing an empirical study of GPT's ability to interpret assembly instructions. In particular, we manually evaluated GPT-4's per-instruction explanations of code segments for four different computer architectures, namely x86, x86-64, ARM, and AArch64. Our study shows that, while inconsistencies and rare errors do exist, GPT's interpretations are highly accurate in general, demonstrating a great potential for such tools to be applied in pedagogical practices for tutoring assembly language.

1 Introduction

Assembly language is a low-level programming language that is specific to a particular computer architecture or microprocessor. It is a symbolic representation of the machine code instructions that a particular processor understands. Many educators believe that it is beneficial for students majoring in computer science to have knowledge of assembly language. A widely recognized reason is that assembly language is a good educational tool that allows students to understand the inner workings of a computer, and thus be well prepared for upper-level courses [4, 12]. We provide a detailed account of how assembly language is intertwined with the computer science curriculum in Section 2.

In many computing education curricula nowadays, however, assembly language is normally an elective course for various reasons. Consequently, despite assembly language being widely regarded as a good way for learning computer systems in depth, it is not given sufficient coverage in a modern computer science curriculum. One way of dealing with this is to dedicate time to revisiting assembly language in certain courses. However, it has the disadvantage of diverting time from the core material of the course, and may bore students who are already familiar with the content. We further discuss the observed challenges brought by assembly language in Section 3. Acknowledging the above concerns, we need to seek a tool that goes beyond the classroom to provide tutoring for students who need to learn and interpret assembly language.

In the recent years, the emergence of generative AI based on large language models (LLMs) has brought a profound impact on our lives. As a specific type of generative AI models, the Generative Pre-trained Transformer (GPT)[9], was first introduced early in 2018. Later, OpenAI announced their commercial chatbot ChatGPT in November 2022, which adopts GPT architecture as the core model. It should be noted that we interchangeably use the two terms, ChatGPT and GPT, throughout this paper unless otherwise specified.

Beyond the usage in interactive conversations, GPT also demonstrates excellent performance in various programming language-related tasks, indicating its promising application prospects in pedagogical practices. Prior work [2, 3, 8] has preliminarily explored the potential of GPT in computer science education scenarios. However, it is still unclear to what extent GPT can help students learn assembly language. We briefly survey the related literature in Section 4 to showcase the current research gap.

In this paper, we delve into the potential opportunities of leveraging GPT as tutoring chatbots to interpret assembly language code in a learning environment, as well as its limitations. Our empirical study based on instructions from mainstream CPU architectures demonstrates that GPT's interpretation of such instructions can be concise, with high accuracy and few errors, making it a promising tutoring tool for assembly language. We elaborate the methodology in Section 5, and discuss our findings in Section 6. Finally, we draw the conclusion in Section 7.

2 Assembly Language in Education

We summarize specific scenarios in certain computer science courses where knowledge of assembly language is required.

- **Computer Architecture:** Learning assembly language helps students understand how the CPU and memory work at a low level. This understanding is fundamental for grasping how high-level languages are executed by the hardware. Therefore, instruction set architectures (ISAs) are often involved in computer architecture pedagogy.
- **Operating System:** When explaining how an OS manages processes and provides services to applications, it is very helpful to appropriately use some assembly code snippets. Assembly instructions or code snippets demonstrate the low-level details of the concepts such as calling convention, interrupt handling, and race condition.
- **Compiler Design:** Compilers translate high-level code into machine code. An understanding of assembly provides insights into what the compiler is doing under the hood. Especially in machine code generation and optimization, basic knowledge of assembly language is essential.
- **Cybersecurity:** Many aspects of computer security [14, 10], including exploit development and reverse engineering, require a solid understanding of assembly language. This knowledge is necessary to understand vulnerabilities and how exploits work at a low level.

3 Challenges in Practice

The practical challenge caused by assembly language in the context of computing education is twofold. From the perspective of students, they usually feel intimidated and discouraged by assembly language, because it has complex syntax and is much closer to the hardware. Therefore, learning assembly language inevitably requires precise understanding of the low-level concepts such as instructions, registers, and memory addressing modes.

From the perspective of teachers, it has become increasingly rare to find a course dedicated solely to teaching assembly language. The first possible reason is that the job market for assembly programmers is much smaller than it was in the past. The second reason is that continuously emerging new technologies have taken up more time originally allocated for assembly language in the curriculum [1]. Another challenge when introducing assembly language in computer-related courses is that assembly language is specific to a particular CPU architecture or family. Therefore, knowledge of the assembly for one architecture does not necessarily translate to proficiency in another. For example, in a current computer organization course, MIPS is widely involved. By contrast, in a reverse engineering course, x86/x86-64 is more preferable.

It is evident that if GPT could take on the role of a tutor and explain assembly language code to students outside the classroom, it would greatly benefit both students and instructors.

4 Related work

The potential applicability of generative AI models has been discussed in the context of computing education. For example, Crandall et al.[2] leverage GPT to inspect entry-level programming assignments and provide code review feedback to students. Denny et al. [3] empirically explore the capability of Copilot in answering introductory Python coding questions. Moreover, Jury et al. [6] investigate the usage of GPT in generating worked examples for an introductory programming course.

We also have observed some educators explored the impacts and challenges brought by ChatGPT in various sub-fields, such as IT education [11], cybersecurity education [7], and data science education [13]. In addition, Jalil et al. studied the effectiveness of ChatGPT in answering typical software testing questions [5]. However, to the best of our knowledge, it is still unclear how well GPT can understand assembly language, thus leaving a research gap.

5 Methodology

In this section, we introduce the environment settings that we use for evaluation and our research methodology in greater detail.

5.1 Settings

In this study, we employ GPT-4, a state-of-the-art multimodal LLM created by OpenAI. GPT-4 has been trained on data up to April 2023, ensuring its responses are informed by the most recent and relevant information available. It also should be noted that all experiments in this work were conducted between February and March 2024. We are aware that GPT is evolving continuously, thus it is possible that testing in future improved versions may yield slightly inconsistent experimental results with this paper.

Our empirical study is conducted based on SecAtlas[14], which is a dataset towards research-involved computer science education. In detail, four ISAs including x86, x86-64, ARM, and AArch64, are considered. For each different ISA, we randomly extract 100 basic blocks from SecAtlas. The number of instructions contained in each resulting basic block varies from 2 to 20.

5.2 Prompt Engineering

When interacting with GPT-4 via the API provided by OpenAI, we select "`gpt-4-turbo-preview`" as the model. The request messages are structured to include two types of content: one for the "system" role and another for the "user" role. The "system" role content outlines general instructions, detailing the input required from the user and the expected output from GPT-4. Meanwhile, the "user" role content provides assembly code as a query.

In particular, we can submit a basic block, namely a sequence of assembly instructions, to GPT-4 for inquiry, as Figure 1 shows. It should be noted that the constant values in assembly code from SecAtlas[14] were intentionally normalized to reduce redundancy and control vocabulary size. Therefore, `POSITIVE` in Figure 1 represents a numeric value, while `ADDRESS` represents the local destination of a branch instruction or a memory address.

```
messages=[
  {"role": "system",
   "content": "You will be provided with a piece of AArch64 assembly code.
              Please generate a single-sentence explanation for each
              assembly instruction."},
  {"role": "user",
   "content": "mov w20,wzr\ncmp w5,POSITIVE\norr w2,w21,w2\nb.eq ADDRESS"}
]
```

Figure 1: A basic block is provided as a query.

```
messages=[
  {"role": "system",
   "content": "You will be provided with a single instruction of x86-64
              assembly code. Please generate a single sentence which
              briefly explains the provided instruction."},
  {"role": "user",
   "content": "XOR EBX,EBX"}
]
```

Figure 2: A single instruction is provided as a query.

Additionally, we can submit a single assembly instruction to GPT-4 for inquiry, as Figure 2 shows. The advantage of this query method is evident: it can significantly reduce repeated inquiries about the same instruction. We establish a local dictionary to record each instruction and its corresponding explanation. Only when an instruction cannot be found in this dictionary do we initiate a query to GPT-4. Based on this approach, we obtained the textual explanations for all 4×100 basic blocks in our dataset.

It is also worth noting that OpenAI API provides a few parameters to control the diversity and randomness of its output. The first parameter `top_p` represents probability mass, that determines the portion of the highest probability tokens to select from. This value is set to 1 by default, and can range from 0 to 1, where lower values increase determinism, and higher values increase randomness. The second parameter is `temperature`, between 0 to 2, which can help to achieve the desired trade-off between coherence and creativity. Lower values for `temperature` result in more consistent outputs. The OpenAI manual generally recommends altering this or `top_p` but not both. Therefore, we empirically set `temperature` as 0.2 as the OpenAI suggests to make the results more deterministic.

6 Results and Discussion

We manually evaluate the correctness of assembly code explanations generated by GPT using the 5-Point Likert scale. The numerical values from 1 to 5 respectively correspond to the following descriptions:

```
1) The explanation makes no sense
2) Serious mistakes exist, which cannot be acceptable
3) Generally acceptable, and mistakes can be fixed easily
4) Overall correct with very tiny mistakes
5) No mistakes can be found
```

Figure 3 shows the evaluation results in terms of average scores across different ISAs. We can observe quite positive performance. Furthermore, the accuracy obtained on x86/x86-64 generally surpasses what we got on ARM/AArch64.

6.1 Case Study

We take advantage of a few case studies to illustrate the strengths and limitations of leveraging GPT as an assembly instructions interpreter. Some concrete results are shown in Table 1.

Case 1 and Case 2 are two positive results. In particular, for the x86-64 instruction "`XOR EBX,EBX`", GPT can not only accurately understand the meaning of the `XOR` mnemonic but also infer that the instruction is actually a zeroing operation based on the fact that the two operands involved are the

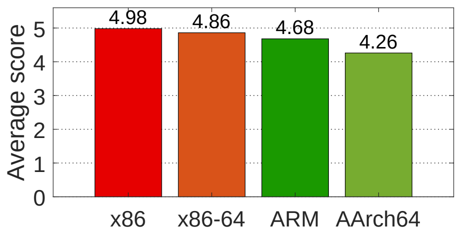


Figure 3: The correctness of assembly code explanations generated by GPT-4.

same. Furthermore, for the instruction "`mov w20,wzr`", GPT recognizes that `wzr` is the zero register in the AArch64 architecture, and further interprets it as the zero value.

By contrast, Case 3 and Case 4 are two negative results. In the x86 architecture, the instruction "`FSTP ST(i)`" copies `ST0` to the destination location `ST(i)` and pops the stack top. But for the given instruction "`FSTP ST0`", it only pops the data off the top of stack with no data transfer. Therefore, the explanation given by GPT is incorrect. Likewise, the explanation for Case 4 is also misleading. In the ARM architecture, `addle` is a conditional arithmetic operation. Namely, only if the `LE` condition (i.e., Less Than or Equal) is true, the addition operation will be performed.

A noteworthy finding is that GPT tends to generate diverse and nondeterministic outputs. In our application scenario, for the same type of instructions, the explanations it provides are often not consistent. Taking the conditional arithmetic operations in ARM including `addle`, `addne`, and `addlt` as examples, most of the time, GPT can correctly interpret the execution conditions. However, in a few cases, errors similar to aforementioned Case 4 may occasionally occur. Furthermore, although some differences do not affect correctness, they may lead to some details about the meaning of the instruction being overlooked. For instance, when interpreting the x86-64 instructions containing key words such as `dword` (double word) and `qword` (quad word), sometimes GPT may emphasize the differences, sometimes may not. For the instructions involving `CMP`/`TEST`, how those related flag are modified, sometimes GPT provides an explanation, but other times it does not.

We also note that absolute incorrect answers by GPT are rare. While there may be inconsistency in its explanations for various instructions, our observation shows that errors were only limited to a small subset of the assembly instructions, such as `adrp` and `ldur` for ARM/AArch64. This property can potentially be exploited to enhance the effectiveness of GPT as a tutoring tool for students.

Table 1: The results of our case studies

#	ISA	Instruction	Explanation from GPT-4
1	x86-64	`XOR EBX,EBX`	It performs an exclusive OR operation between the `EBX` register and itself, effectively setting `EBX` to zero.
2	AArch64	`mov w20,wzr`	It moves the zero value into register `w20`.
3	x86	`FSTP ST0`	It stores the value from the top of the floating-point stack into a specified location and then pops the stack.
4	ARM	`addle r2,r2,POSITIVE`	It adds the value of the constant POSITIVE to the value in register `r2` and stores the result in `r2`, updating the condition flags based on the result.

6.2 Insight

Overall, GPT models show great promise in offering tutoring of explaining assembly instructions to students outside a classroom. More importantly, GPT demonstrates extensive knowledge of different ISAs, which can broadly fulfill different requirements across courses in computer-related disciplines.

On the other hand, we should notice that GPT does not always provide reliable answers. In the absence of manual fact-checking, students can be easily misled by inaccurate answers given by GPT that appear to be plausible on the surface. This can be especially true for beginners with a weak foundation of knowledge who may lack the ability to discern reliable information. The findings provided by this study can make good examples for reminding students to use GPT with caution.

7 Conclusion

Although acquiring some assembly language knowledge can help students to understand computer systems in depth, and well prepare them for upper-level courses, both instructors and students still face challenges in reality. Meanwhile, the emerging AI technology, GPT, is bringing revolutionary changes in today's computing education. However, the potential of integrating GPT into pedagogical practices for generating assembly code explanations is unclear. This work piloted the use of GPT as an assembly language tutoring chatbot for students outside the classroom. We empirically explore the capability of GPT in interpreting assembly instructions. Overall, GPT's interpretations are accurate and helpful across four different ISAs, showing great promise in this aspect. Specifically, our findings also revealed its limitations, which should be noticed by students when referring to the outputs provided by GPT.

In future work, we will perform additional testing and evaluation on different AI platforms such as Google's Gemini. We also plan to adopt GPT in our teaching practice and investigate its effect on student learning outcomes.

References

[1] Krishna K Agarwal and Achla Agarwal. Do we need a separate assembly language programming course? *Journal of Computing Sciences in Colleges*, 19(4):246–251, 2004.

[2] Aaron S Crandall, Gina Sprint, and Bryan Fischer. Generative pre-trained transformer (GPT) models as a code review feedback tool in computer science programs. *Journal of Computing Sciences in Colleges*, 39(1):38–47, 2023.

[3] Paul Denny, Viraj Kumar, and Nasser Giacaman. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *the 54th ACM Technical Symposium on Computer Science Education*, pages 1136–1142, 2023.

[4] Kosuke Imamura. Assembly language is more than a teaching tool. *Journal of Computing Sciences in Colleges*, 20(2):49–54, 2004.

[5] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. ChatGPT and software testing education: Promises & perils. In *IEEE international conference on software testing, verification and validation workshops*, pages 4130–4137, 2023.

[6] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, and Andrew Luxton-Reilly. Evaluating LLM-generated worked examples in an introductory programming course. In *the 26th Australasian Computing Education Conference*, pages 77–86, 2024.

[7] Shafi Parvez Mohammed and Gahangir Hossain. ChatGPT in education, healthcare, and cybersecurity: Opportunities and challenges. In *IEEE 14th Annual Computing and Communication Workshop and Conference*, pages 316–321, 2024.

[8] Xianshan Qu, Fei Zuo, Xiaopeng Li, and Junghwan Rhee. Context matters: Investigating its impact on ChatGPT's bug fixing performance. In *IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA)*, 2024.

[9] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[10] Junghwan Rhee, Myungah Park, Fei Zuo, Shuai Zhang, Gang Qian, Goutam Mylavarapu, Hong Sung, and Thomas Turner. Developing incident response-focused cybersecurity undergraduate curricula. *Journal of Computing Sciences in Colleges*, 38(7):65–74, 2023.

[11] Nazmus Sakib, Fahim Islam Anik, and Lei Li. ChatGPT in IT education ecosystem: Unraveling long-term impacts on job market, student learning, and ethical practices. In *the 24th Annual Conference on Information Technology Education*, pages 73–78, 2023.

[12] James T Streib. Simplified assembly language programming. In *the 7th Annual Consortium on Computing in Small Colleges Midwestern Conference*, pages 80–84, 2000.

[13] Yong Zheng. ChatGPT for teaching and learning: an experience from data science education. In *Proceedings of the 24th Annual Conference on Information Technology Education*, pages 66–72, 2023.

[14] Fei Zuo, Junghwan Rhee, Yonghyun Kim, Jeehyun Oh, and Gang Qian. A comprehensive dataset towards hands-on experience enhancement in a research-involved cybersecurity program. In *Proceedings of the 24th Annual Conference on Information Technology Education*, 2023.

Assessing Student Perceptions of Co-Teaching in a 3rd-Year Computer Science Course*

Abbas Attarwala[1], Pablo Raigoza[2]
[1]Computer Science Department
California State University
Chico, CA 95973
aattarwala@csuchico.edu
[2]Computer Science Department
Cornell University
Ithaca, NY 14850
pr428@cornell.edu

Abstract

In this research, the effects of various teaching methodologies such as solo teaching, parallel coordinated teaching (PCT), and sequential teaching (SEQT) on student perceptions in a third-year programming language course at Boston University (BU) are studied. PCT and SEQT, as variants of co-teaching, contrast with the independent approach of solo teaching. This research uses student evaluation data to analyze eight distinct evaluative questions, including areas such as fairness in grading, stimulation of student's interest in the course material, and overall instructor ratings. These eight questions are analyzed using student course evaluations across the three aforementioned teaching methodologies to determine if there are statistically significant differences in perceptions. The results show that consistent instructor presence throughout the semester, as seen in solo teaching and PCT scenarios, significantly enhances student perceptions of fairness and overall satisfaction. In contrast, SEQT, which involves instructor changes in the middle of the semester, is associated with less favorable student evaluations. The study highlights the importance of instructor consistency and the potential disruptions caused by changing instructors mid-course.

1 Introduction

In higher education, it is common for multiple instructors to teach different sections of the same course (referred to as co-teaching) or for one instructor to handle multiple sections (referred to as solo teaching). In this paper, we look at two different kinds of co-teaching: (1) Parallel coordinated teaching (PCT) when two instructors operate separate sections of the same course, independently managing their classrooms while engaging in a highly collaborative process. They share a common set of lessons, assignments, and resources to maintain consistency across sections. Coordination meetings on a weekly basis ensure synchronized instructional planning. Despite sharing a unified course structure, each instructor independently executes their teaching responsibilities. (2) Sequential Teaching (SEQT) on the other hand involves a single instructor leading the instruction for both sections during the initial half of the semester, with the other instructor observing and providing feedback. At the semester's midpoint, a role reversal occurs: the observing instructor takes over teaching duties for both sections, while the initial instructor assumes the role of observer.

These variations, PCT and SEQT, represent distinct forms of co-teaching strategies that diverge significantly from the solo teaching approach. Solo teaching is defined as a single instructor teaching multiple sections of the same course for the entire semester. This research aims to evaluate which of these strategies is most effective from the students' perspective. By analyzing student course evaluations, this study seeks to understand students' perceptions regarding the effectiveness of these teaching methods and determine which is deemed most conducive to their learning.

In the fall of 2020, PCT was employed, where I[1] and a colleague independently taught different sections of CS 320, sharing the same teaching assistants, tutors, assignments, and projects. We held weekly meetings to ensure consistent progress across both sections, allowing students in different sections to learn the same material and collaborate on identical assignments. This method aimed to leverage collaborative efforts to improve teaching results, regardless of the hybrid format required by pandemic restrictions, where students participated both in person; remotely and asynchronously. The spring of 2021 saw the continuation of the hybrid model as I took on solo teaching responsibilities for two separate sections of CS 320. This period allowed for an evaluation of

[1]First person use in this paper refers to Abbas Attarwala

the effectiveness of a single instructor managing multiple course sections. In fall of 2021, while most students had returned to campus, the SEQT method was introduced. I taught the initial half of the semester, and my colleague, the same from the previous year, took over for the latter half. This structure included a complete handover and mutual class observations to provide feedback, ensuring continuity and instructional coherence.

This study examines how these different teaching formats affected student perceptions across eight questions: (1) The instructor's effectiveness in explaining concepts; (2) The instructor's ability to stimulate interest in subject; (3) The instructor's encouragement in class participation; (4) The instructor's fairness in grading; (5) The instructor's promptness in returning assignments; (6) The instructors quality of feedback to students; (7) The instructor's availability outside of class; and (8) The overall rating of the instructors. By examining student feedback gathered across these eight questions from teaching evaluation, the study aims to gain understanding into the most effective teaching methods among solo, PCT, and SEQT.

2 Literature Review

Co-teaching is a multifaceted approach that adapts to various educational settings, as detailed in the literature by [7] and [1, 2]. These authors delineate several co-teaching strategies, such as: (1) One Teach, One Observe, where one instructor leads the class while the other observes; (2) Parallel Teaching, where instructors teach multiple sections of the same course; (3) Teaming, where both instructors share the instructional space equally, often teaching and interacting together with the students; and (4) One Teach, One Assist, where one instructor primarily leads the lesson while the other provides targeted support to students as needed. [8] introduces other variants of co-teaching such as (1) Supportive, where one instructor provides assistance to individual students while the other delivers the main content; (2) Complementary, which sees one instructor enhancing the lessons of the other with additional information or learning activities; and (3) Synergetic, a dynamic approach where both instructors merge their expertise to create an enriched learning environment for the students. Despite the variations in terminology, there is considerable common ground among these co-teaching models, reflecting the adaptability of co-teaching to suit diverse educational needs.

In their exploration of collaborative teaching in large computer science classes in India, [8] discuss the effectiveness of various collaborative teaching methods, including parallel teaching where multiple instructors teach different sections of the same class. Drawing from these insights, my research presents three distinct approaches to teaching CS 320 at BU. The first scenario serves as a baseline, where I alone taught multiple sections of CS 320 during the spring of 2021. This instance provides a reference point for comparing the impact of solo versus co-teaching instructional approaches. Data from course evaluations were analyzed across these three teaching methodologies. We conducted Welch's t-test on student ratings to analyze the data, and this approach revealed statistically significant differences in the evaluations across the teaching methodologies. To the best of our knowledge, no other research has examined students' views from course evaluations on co-teaching in a 3rd-year computer science course, using solo teaching as a benchmark.

[4] mentions that co-teaching is a powerful but often overlooked way to encourage deep and thoughtful discussions. When instructors reflect together, they bring up important questions and issues, leading to discussions that can result in changes and improvements in teaching. In the fall of 2020, the other instructor and I held weekly meetings where we reflected and discussed effective teaching pedagogy and what should be included in the curriculum. Through these discussions, we decided to enhance the curriculum to include a topic on parser combinators. We had both noticed in our previous teaching experiences that students often wrote ad-hoc parsers that did not scale well, and students struggled to write effective parsers. The inclusion of parser combinators was inspired by our collaborative reflections on improving teaching methods and curriculum content.

[2] found that middle school 6th graders preferred co-teaching over a traditional single-instructor approach. Our research takes a similar inquiry into co-teaching's efficacy but focuses on 3rd-year computer science students at the university level. Using course evaluations, we examine student perceptions of teaching effectiveness using both co-teaching strategies and traditional solo teaching methods within the same course. As presented in Section 3, student evaluations consistently rated the sections I taught solo more favorably compared to those taught using PCT and SEQT. Personally, I noticed an improvement in my lesson organization during SEQT, when I was responsible for teaching only the first half of the semester. I would receive regular feedback from my co-instructor as he would observe my teaching and give me constructive feedback. [3] also mention that one of the benefits of co-teaching is to provide ongoing supportive feedback based on direct observation of the other instructor teaching throughout the semester. SEQT also allowed me more time to also observe and learn from the other instructor when the other instructor took over both sections in the second half of the semester. Other instructors as mentioned in [9] observed similar gains. One of the instructor mentions that her co-instructor allows her to observe her teaching, where she notice strategies and methods she'd like to try. However, the statistical analysis in Section 4 of this paper suggests that students do not prefer SEQT, finding PCT and solo

teaching to be much more effective for their learning.

3 Data

In the fall of 2020, I taught one section of CS 320 at BU, while my colleague taught another section concurrently. I refer to this arrangement as PCT. The course evaluations for my section are documented in Table 1. In the spring of 2021, I independently taught two sections of CS 320 at BU, a scenario I refer to as solo teaching of multiple sections of the same course. The evaluations for these sections can be found in Table 2. During the fall semester of 2021, I began teaching two sections of CS 320 at BU but only continued until mid-semester. At that point, my colleague took over and completed the semester. This method is termed SEQT, another variant of co-teaching. Evaluations for the portion of the semester I taught are available in Table 3.

Students completed course evaluations separately for me and my colleague during the last week of the semester. For SEQT, students completed evaluations for both instructors also in the last week of the semester. The course evaluations presented in this paper are solely mine, as I did not have access to my colleague's course evaluation data. In these tables, N is the sample size, SD is the standard deviation and M is the mean.

Table 1: Parallel Coordinated Teaching of CS 320 in Fall 2020 for section A. My colleague taught the other section B.

Question #	Faculty Evaluation	N	SD	M
1	Effectiveness in explaining concepts	42	0.65	4.62
2	Ability to stimulate interest in subject	42	0.70	4.52
3	Encouragement of class participation	42	0.98	4.26
4	Fairness in grading	42	0.73	4.50
5	Promptness in returning assignments	42	0.83	4.21
6	Quality of feedback to students	42	0.82	4.43
7	Availability outside of class	42	0.92	4.33
8	Overall rating of instructor	42	0.53	4.76

Table 2: Solo teaching of CS 320 in Spring 2021 for sections A and B.

Question #	Faculty Evaluation	Section A			Section B		
		N	SD	M	N	SD	M
1	Effectiveness in explaining concepts	52	0.69	4.58	45	0.56	4.64
2	Ability to stimulate interest in subject	52	0.91	4.44	45	0.85	4.62
3	Encouragement of class participation	51	0.75	4.51	45	0.70	4.64
4	Fairness in grading	51	0.69	4.59	45	0.85	4.60
5	Promptness in returning assignments	51	0.85	4.43	45	0.91	4.42
6	Quality of feedback to students	51	0.78	4.51	45	0.86	4.47
7	Availability outside of class	51	0.85	4.43	45	0.64	4.62
8	Overall rating of instructor	50	0.65	4.68	45	0.57	4.73

Table 3: Sequential Teaching of CS 320 in Fall 2021 for sections A and B.

Question #	Faculty Evaluation	Section A			Section B		
		N	SD	M	N	SD	M
1	Effectiveness in explaining concepts	32	0.96	4.34	45	0.83	4.40
2	Ability to stimulate interest in subject	32	0.91	4.28	45	1.12	4.11
3	Encouragement of class participation	32	0.89	4.38	45	1.10	4.18
4	Fairness in grading	32	1.01	4.19	45	1.22	4.07
5	Promptness in returning assignments	32	0.97	4.00	45	1.12	4.18
6	Quality of feedback to students	32	0.90	4.25	44	1.23	4.07
7	Availability outside of class	32	1.08	4.22	45	0.99	4.33
8	Overall rating of instructor	32	0.79	4.44	45	0.88	4.38

In the evaluations, students rated my performance on a scale from 1 (poor) to 5 (superior) across eight questions. Specifically, in Question #8, "Overall rating of instructor," I received my highest mean rating of 4.76 during the fall 2020 semester, under the PCT method. The lowest I received is 4.38 in Section B of the SEQT teaching in the fall of 2021 semester. A key question arises from this observation: Is the difference in mean ratings of eight questions across Table 1, Table 2 and Table 3, statistically significant? Understanding whether these differences are statistically significant is crucial. It enables us to determine whether the variations observed between solo teaching of multiple sections, PCT, and SEQT are due to the teaching methods themselves or if they occur purely by chance. This analysis is not just academic; it has practical implications. Confirming that different co-teaching strategies such as PCT and SEQT significantly affect student perceptions could influence future pedagogical approaches in computer science education. Such insights could guide universities and educators in structuring their courses to enhance learning outcomes and student satisfaction. However, it is essential to acknowledge that other factors could be at play between semesters that might affect my ratings, including changes in class size, different student cohorts; otherwise, I had the same course material, similar assignments and delivery methods.

To address these questions, student course evaluation data from Table 1, Table 2, and Table 3 were analyzed using a two-tailed Welch's t-test. The results of these statistical tests are detailed in Section 4 of this paper.

4 Results and Discussions

In this study, we perform a statistical analysis to evaluate the effectiveness of different teaching methods in CS 320. The methods compared include solo teaching, PCT, and SEQT. In the case of solo teaching and SEQT, each involved two sections of the same course, which are treated as distinct entities for the purpose of analysis. Conversely, for PCT, I taught one section while my colleague taught the other; hence, only the course evaluation data from my teaching is considered in this analysis, not my colleague's.

We set up our null hypothesis (H_0) and alternative hypothesis (H_1) as:

- H_0: There is no difference in the mean ratings of each of the 8 evaluation questions across the different teaching sections and methods.
- H_1: There is a significant difference in the mean ratings of the 8 evaluation questions across the different teaching sections and methods, which could be either positive or negative.

To assess these hypotheses, we employ a two-tailed Welch's t-test for each of the 8 evaluation questions. A two-tailed test is chosen because it allows us to detect both increases and decreases in teaching effectiveness, regardless of the direction. The traditional Student's t-test assumes equal variances between the groups being compared. From the data available in Section 3 this is not the case. Consequently, relying on the Student's t-test could lead to inaccurate conclusions. In contrast, Welch's t-test does not require the assumption of equal variances, making it more suitable for the teaching course evaluation data [5, 6] in this paper. This test provides a more reliable assessment by adjusting the degrees of freedom according to the sample sizes and variances of each group. Here is how the $t\_statistic$ and $Degree\ of\ Freedom$ are calculated for the Welch's t-test:

$$t\_statistic = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$Degree\ of\ Freedom = \frac{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)^2}{\frac{(\sigma_1^2/n_1)^2}{n_1-1} + \frac{(\sigma_2^2/n_2)^2}{n_2-1}}$$

μ_1, μ_2 are the means of the two groups, σ_1^2, σ_2^2 are their variances, and n_1, n_2 are the sample sizes.

We conducted a total of 10 pairwise comparisons for each of the eight questions.

1. Solo Section A vs. Solo Section B
2. Solo Section A vs. PCT
3. Solo Section A vs. SEQT Section A
4. Solo Section A vs. SEQT Section B
5. Solo Section B vs. PCT
6. Solo Section B vs. SEQT Section A
7. Solo Section B vs. SEQT Section B
8. PCT vs. SEQT Section A
9. PCT vs. SEQT Section B
10. SEQT Section A vs. SEQT Section B

The data for Solo Section A and Solo Section B is sourced from Table 2. Data pertaining to PCT is detailed in Table 1, while the data for SEQT Section A and SEQT Section B can be found in Table 3.

The significance level for rejecting the null hypothesis is set at 5%, aligning with standard practices for statistical significance. This means that if the test results show less than a 5% probability, we reject the H_0 in favor of H_1. By conducting this analysis, our aim is to identify precisely whether specific teaching approaches in these various setups significantly impact student evaluation outcomes.

The pairwise comparison results of performing the Welch's t-test for the evaluative questions are systematically presented across several tables. On each of the table, we present the t-statistic, p-value and the degree of freedom. p-value that are statistically significant i.e., less than 5% are bolded in the table. Table 4 details the comparisons for Questions 1 and 2. Similarly, Table 5 outlines the results for Questions 3 and 4, Table 6 for Questions 5 and 6, and Table 7 for Questions 7 and 8. Each table provides insights into 10 pairwise comparisons, illustrating the variances in student perceptions across different teaching methodologies.

Table 4: Comparison of Questions 1 and 2 across all sections

Comparison	Question 1			Question 2		
	t-statistic	p-value	Degrees of Freedom	t-statistic	p-value	Degrees of Freedom
Solo. Sec A vs. Solo. Sec B	-0.473	0.6377	94.63	-1.007	0.3167	94.43
Solo Sec A vs. PCT	-0.289	0.7736	89.80	-0.482	0.6312	91.81
Solo. Sec A vs. SEQT. Sec A	1.232	0.2237	50.73	0.783	0.4367	65.76
Solo. Sec A vs. SEQT. Sec B	1.151	0.2530	85.87	1.577	0.1186	84.77
Solo Sec B vs. PCT	0.153	0.8786	81.18	0.601	0.5497	83.73
Solo.Sec B vs. SEQT. Sec A	1.586	0.1195	45.92	1.660	0.1017	64.03
Solo. Sec B vs. SEQT. Sec B	1.608	0.1119	77.18	2.433	**0.0171**	82.06
PCT vs. SEQT. Sec A	1.420	0.1615	51.67	1.239	0.2206	56.56
PCT vs. SEQT.Sec B	1.381	0.1709	82.56	2.062	**0.0427**	74.53
SEQT. Sec A vs. SEQT. Sec B	-0.286	0.7761	60.64	0.733	0.4657	73.59

Table 5: Comparison of Questions 3 and 4 across all sections

Comparison	Question 3			Question 4		
	t-statistic	p-value	Degrees of Freedom	t-statistic	p-value	Degrees of Freedom
Solo Sec A vs. Solo Sec B	-0.878	0.3821	93.69	-0.063	0.9501	84.81
Solo Sec A vs. PCT	1.358	0.1785	75.66	0.606	0.5458	85.55
Solo Sec A vs. SEQT. Sec A	0.687	0.4947	57.68	1.970	0.0544	49.20
Solo Sec A vs. SEQT. Sec B	1.695	0.0942	76.21	2.525	**0.0139**	67.60
Solo Sec B vs. PCT	2.068	**0.0421**	73.76	0.590	0.5569	84.43
Solo Sec B vs. SEQT. Sec A	1.377	0.1739	56.56	1.873	0.0660	59.46
Solo Sec B vs. SEQT. Sec B	2.367	**0.0205**	74.62	2.391	**0.0192**	78.57
PCT vs. SEQT. Sec A	-0.550	0.5841	69.73	1.468	0.1478	54.11
PCT vs. SEQT. Sec B	0.359	0.7207	84.82	2.010	**0.0481**	72.74
SEQT. Sec A vs. SEQT. Sec B	0.880	0.3817	73.68	0.471	0.6392	73.19

Table 6: Comparison of Questions 5 and 6 across all sections

Comparison	Question 5			Question 6		
	t-statistic	p-value	Degrees of Freedom	t-statistic	p-value	Degrees of Freedom
Solo Sec A vs. Solo Sec B	0.055	0.9559	90.58	0.238	0.8128	89.54
Solo Sec A vs. PCT	1.258	0.2116	88.36	0.479	0.6334	85.80
Solo Sec A vs. SEQT. Sec A	2.060	**0.0438**	59.50	1.347	0.1830	58.98
Solo Sec A vs. SEQT. Sec B	1.219	0.2263	81.55	2.045	**0.0446**	70.69
Solo Sec B vs. PCT	1.126	0.2635	84.96	0.222	0.8248	84.96
Solo Sec B vs. SEQT. Sec A	1.921	0.0592	64.22	1.077	0.2856	65.01
Solo Sec B vs. SEQT. Sec B	1.116	0.2677	84.46	1.774	0.0800	76.79
PCT vs. SEQT. Sec A	0.981	0.3304	60.90	0.885	0.3792	63.43
PCT vs. SEQT. Sec B	0.143	0.8870	80.94	1.604	0.1130	75.25
SEQT. Sec A vs. SEQT. Sec B	-0.752	0.4544	72.03	0.737	0.4636	73.99

Table 7: Comparison of Questions 7 and 8 across all sections

Comparison	Question 7			Question 8		
	t-statistic	p-value	Degrees of Freedom	t-statistic	p-value	Degrees of Freedom
Solo Sec A vs. Solo Sec B	-1.246	0.2161	91.82	-0.399	0.6905	92.94
Solo Sec A vs. PCT	0.540	0.5908	84.62	-0.650	0.5172	89.93
Solo Sec A vs. SEQT. Sec A	0.933	0.3547	54.66	1.435	0.1566	56.92
Solo Sec A vs. SEQT. Sec B	0.527	0.5992	87.34	1.873	0.0647	80.41
Solo Sec B vs. PCT	1.696	0.0943	72.60	-0.254	0.7998	85.00
Solo Sec B vs. SEQT. Sec A	1.874	0.0672	46.38	1.774	0.0818	53.08
Solo Sec B vs. SEQT. Sec B	1.650	0.1031	75.31	2.239	**0.0281**	75.39
PCT vs. SEQT. Sec A	0.462	0.6455	60.72	1.977	0.0534	51.34
PCT vs. SEQT. Sec B	0.000	1.0000	85.00	2.458	**0.0163**	73.01
SEQT. Sec A vs. SEQT. Sec B	-0.456	0.6501	63.21	0.313	0.7551	70.93

In Table 8, we provide a summary of the statistically significant results from Table 4, Table 5, Table 6, and Table 7. Additionally, Table 8 includes the frequency of statistically significant results, indicating how often each teaching method was perceived by students as either positive or negative.

The questions regarding Question #1 i.e., "Effectiveness in explaining concepts" and Question #7 i.e., "Availability outside of class" have zero occurrences of statistically significant results across all comparisons as shown in Table 8. This statistical insignificance indicates that these questions are less sensitive to changes in teaching methods. For instance, the ability to explain concepts effectively is related to individual instructor skills and less to the mode of delivery (Solo vs. PCT vs. SEQ). Similarly, availability outside of class is influenced by the instructor's commitment to student interaction outside formal teaching sessions, rather than how the content is delivered. This consistency could be seen as a strength, suggesting that my core teaching competencies are stable across different teaching environments. However, it might also imply a ceiling effect, where improvements are harder to achieve because students cannot rate any of the questions above 5.

For Question #4 i.e., "Fairness in Grading," it is noteworthy that Solo Sections A and B, as well as the PCT section, consistently outperformed SEQT Section B. This observation indicates that sections taught by a single instructor throughout the semester are perceived as more fair in their grading practices compared to those experiencing an instructor transition. Furthermore, the lack of statistically significant differences in grading fairness perceptions between Solo Sections A, B, and the PCT method suggests that consistency in instructor presence throughout the semester creates a stable environment for equitable grading practices. In contrast, in SEQT where the instructor changes midway through the semester, may introduce elements of uncertainty or perceived inconsistency in grading criteria among students. This difference highlights the potential challenges posed by instructor transitions in maintaining perceived grading fairness and underscores the importance of ensuring that both instructors in a SEQT arrangement closely align their grading standards and transparently communicate any necessary transitions in assessment strategies to the students. Although the differences in "Fairness in Grading" for SEQT Section A do not reach statistical significance at the conventional 5% level, they are notable at the 10% level. However for Section B of SEQT, statistical significant results are observed at the 5% level. This suggests a trend towards significance, indicating that while SEQT Section A's grading perceptions are not as clearly differentiated as those in SEQT Section B, they still do not align as favorably when compared to the more consistent outcomes observed in Solo Sections A and B, and the PCT section. This finding highlights a potential area for improvement in ensuring grading consistency across different teaching formats, particularly in sections experiencing instructional transitions.

For all statistically significant results involving SEQT Section B, SEQT Section B performed worse. Specifically, for Questions #2, #3, and #8, i.e., "Ability to stimulate interest in the subject," "Encouragement of class participation," and "Overall rating of instructor," SEQT Section B was worse off compared to Solo Section B and PCT. However, SEQT Section A also performs less favorably compared to sections taught by a single instructor throughout the semester. It does not reach the conventional threshold for statistical significance at the 5% level, but is significant at the 10% level. This implies that the instructional transition in the middle of the semester transition in SEQT Sections A and B may disrupt the continuity that students seem to prefer, impacting their engagement and overall satisfaction with the course. Students prefer a single instructor to teach for the entire semester whether the instructor is teaching both sections of the same class or the instructor is engaged in PCT but is still responsible for the entire section for the entire semester. These findings reinforce the idea that consistent instructor presence, whether in solo-taught sections or in PCT where the instructor remains the same throughout the semester, is critical to maintaining student interest, participation, and overall satisfaction.

Table 8: Comprehensive Summary of Statistically Significant Results. In this table, '+' represents the number of statistically significant results within the count of that method that are perceived positively by students, while '-' indicates those results perceived negatively.

Summary of Results		Frequency of Methods			Frequency by Question	
Category	Count	Method	Total Count	Perception	Question	Count
Total # of significant results at the 5% level:	11	Solo Sec A	3	+3	Question 1	0
		Solo Sec B	5	+5	Question 2	2
		PCT	4	+3 and -1	Question 3	2
		SEQ. Sec A	1	-1	Question 4	3
		SEQ. Sec B	9	-9	Question 5	1
					Question 6	1
					Question 7	0
					Question 8	2

The three major findings of this research are that: (1) Universities should maintain the same instructor throughout a course to enhance perceptions of fairness and grading, as well as overall instructor ratings. Our study shows that solo teaching and PCT outperform SEQT. (2) To ensure smooth transitions in SEQT, have instructors collaborate closely, communicate changes clearly to students, and select instructors with similar teaching styles to minimize disruptions. (3) PCT is preferred over SEQT due to consistent instructor presence, which enhances student comfort and engagement. This preference is reflected in higher overall learning experience and satisfaction in student evaluations.

We acknowledge the limitations of our research. I lacked access to my colleague's teaching evaluations, which could have provided more insight into correlations. Expanding the study to include different disciplines and course formats could also enhance the generalizability of the findings.

5 Conclusions

This research explored the impact of two specific co-teaching methods, PCT and SEQT, compared to solo teaching. The study examined how these different instructional strategies affect student evaluations across eight questions. Solo teaching and PCT, where a single instructor is responsible for the course throughout the semester, consistently led to more favorable student perceptions. This was particularly evident in the areas of grading fairness and the overall rating of the instructor. Students valued the consistency and continuity provided by having the same instructor, suggesting that frequent changes in instructional personnel can disrupt student satisfaction and engagement.

SEQT, especially in Section B where the instructor changes midway through the semester, often resulted in less favorable evaluations. This method appeared to introduce a level of disruption that negatively impacted student perceptions, particularly in terms of grading fairness and the encouragement of class participation. For courses that must be taught by multiple instructors, our research suggest that it is probably best to use PCT and not SEQT form of co-teaching.

Acknowledgement

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

References

[1] Megan Berg. "Student Perceptions of Co-Teaching". Dissertations, Theses, and Projects. Dissertation. Minnesota State University, 2023. URL: https://red.mnstate.edu/thesis/841.

[2] Caitlin Caprio. "Student Perceptions on the Effectiveness of Co-teaching: Do Students Perceive Co-teaching to Be Beneficial?" PhD thesis. Trinity Christian College, 2019.

[3] Amy Chanmugam and Beth Gerlach. "A Co-Teaching Model for Developing Future Educators' Teaching Effectiveness." In: *International Journal of Teaching and Learning in Higher Education* 25.1 (2013), pp. 110–117.

[4] Jayne Crow and Lesley Smith. "Co-teaching in higher education: Reflective conversation on shared experience as continued professional development for lecturers and health and social care students". In: *Reflective practice* 6.4 (2005), pp. 491–506.

[5] Marie Delacre, Daniël Lakens, and Christophe Leys. "Why psychologists should by default use Welch's t-test instead of Student's t-test". In: *International Review of Social Psychology* 30.1 (2017), pp. 92–101.

[6] Ben Derrick, Deirdre Toher, and Paul White. "Why Welch's test is Type I error robust". In: *The quantitative methods for Psychology* 12.1 (2016), pp. 30–38.

[7] Marilyn Friend et al. "Co-teaching: An illustration of the complexity of collaboration in special education". In: *Journal of educational and psychological consultation* 20.1 (2010), pp. 9–27.

[8] Sanjay Goel et al. "Collaborative teaching in large classes of computer science courses". In: *2015 Eighth International Conference on Contemporary Computing (IC3)*. IEEE. 2015, pp. 397–403.

[9] Hillary Merk, Melanie Betz, and Colleen O'Mara. "Teacher candidates' learning gains: The tale of two co-teachers". In: *Networks: An Online Journal for Teacher Research* 17.2 (2015), pp. 602–602.

AI Tools in Matlab Course Education: Instructor Point of View*

Rawan Al-Nsour
Technology Managment & Mechatronics Department
Utah Valley University
Orem, UT 84058
rawan.nsour@uvu.edu

Abstract

This work aims to investigate the influence of AI tools, specifically ChatGPT, on assignment submissions for an undergraduate programming course. The study evaluates the variance between MATLAB code submissions supported by ChatGPT and those based solely on traditional classroom resources such as instructor notes, textbooks, and class exercises. By analyzing these differences, the research seeks to highlight the advantages of using AI as an assistant tool, including enhanced efficiency and personalized feedback. However, it also examines the drawbacks, such as potential over-reliance on AI and its impact on achieving students' learning goals. Additionally, the study provides recommendations on how to manage and integrate this new technology effectively to ensure that it complements rather than detracts from the educational experience. Through this comprehensive evaluation, the paper seeks to offer insights into balancing AI assistance with traditional teaching methods to optimize learning outcomes in programming education.

1 Introduction

In the current era of artificial intelligence, the submission of assignments, particularly in programming courses, has become more challenging and affects academic performance. Traditional teaching methods often struggle to engage students and meet their needs effectively. This has led to growing interest in innovative strategies such as using ChatGPT, its ability to generate convincing responses and provide personalized feedback offers significant potential to enhance programming education. By utilizing ChatGPT, instructors can create dynamic educational environments that efficiently address students' needs, thereby transforming the learning process [7]. ChatGPT, an AI model that is globally recognized for its expertise in handling complex language tasks through conversation [8]. Its capabilities show significant promise in education across various areas: it excels in generating accurate mathematical and programming statements, responding to related questions, and solving problems successfully. Numerous studies confirm its superiority over other models in these tasks [1, 2]. Additionally, ChatGPT is time-efficient and delivers precise answers [5]. In many cases this use led to increased motivation, better outcomes, and greater satisfaction, particularly for those with special educational needs [4]. However, ChatGPT also faces several ethical and operational challenges. These include blind trust without verification, regulatory restrictions, dehumanization in human-machine interactions, prioritizing metrics over societal norms, and information overload. Its self-referential evaluation lacks transparency and oversight, posing accountability issues. Additionally, concerns about data and model bias, misinformation, and privacy underscore the need for responsible deployment and ethical use [9, 3, 6]. The present study aims to evaluate two types of assignment submissions in an undergraduate MATLAB programming course: one based on learned commands and traditional classroom resources, and the other utilizing ChatGPT. The comparison focuses on the pros and cons of using AI models for class assignments and explores how to leverage this technology positively for future hands-on learning.

2 Class Assignment

The following is an in-class assignment designed to teach the **MESHGRID** command in the MATLAB® programming class. The volume of a circular cylinder is calculated using the formula

$$\text{volume} = \pi r^2 h \tag{1}$$

where r is the radius and h is the height. For this assignment, students are tasked with calculating the volume of cylindrical containers with radii ranging from 0 to 14 meters and heights ranging from 12 to 22 meters. The radii and heights should be incremented by 2 meters within these specified ranges. This exercise will help students understand how to use the **MESHGRID** function to create matrices of radius and height values, and subsequently apply the volume formula across these matrices to compute the volumes of multiple cylinders efficiently. The *meshgrid* function in MATLAB is used to create two-dimensional grids that are essential for evaluating functions of two variables and for 3D plotting. Given two vectors that define the ranges for x and y coordinates, *meshgrid* generates matrices X and Y, where each element in X represents the x-coordinate and each element in Y represents the corresponding y-coordinate, in the next two subsections, I will explain two different submissions for the aforementioned class assignment.

2.1 Regular Submission

A regular submission for an in-class assignment covers a comprehensive engagement with various educational resources and a consistent demonstration of student effort. Students are expected to thoroughly review and integrate content from instructor-provided notes, which serve as a primary source of detailed information and guidance. The course textbook plays a crucial role, providing essential readings, theoretical concepts, examples, and practice problems that students should reference and incorporate into their assignments. Active participation in lectures, whether conducted face-to-face or online, is vital. These sessions offer direct instruction and opportunities for interaction, helping students to better understand the material presented. Additionally, active involvement during class discussions and group activities enhances learning by allowing students to engage with the content and with their peers.

Attending office hours is another important component. These sessions provide a platform for students to seek clarification on complex topics, receive personalized guidance, and address any academic challenges they may face. Utilizing these opportunities helps solidify their understanding and improve their performance. Consistently completing homework assignments is also critical. These tasks reinforce the material covered in lectures and readings, enabling students to apply their knowledge and identify areas where they may need further practice or clarification. Overall, a regular submission should reflect a well-rounded and diligent approach to learning. It should be timely, ensuring adherence to deadlines, and it should be complete, addressing all parts of the assignment comprehensively. Accuracy is paramount, with correct answers and well-reasoned explanations. Clarity and organization are also essential, making the assignment easy to follow and understand. Finally, the submission must demonstrate originality, showcasing the student's own understanding and effort while being free from plagiarism. This holistic approach ensures that the assignment is a true representation of the student's capabilities and their engagement with the course material. the class assignment regular submission solution is shown in Figure 1, this code calculates the volumes of cylindrical containers for a range of radii and heights. The radius vector is defined from 0 to 14 with increments of 2 (0:2:14), and the height vector is defined from 12 to 22 with increments of 2 (12:2:22). Using *meshgrid*, matrices R and H are created to represent all combinations of these radii and heights. This results in a matrix volume containing the volumes for each radius-height combination

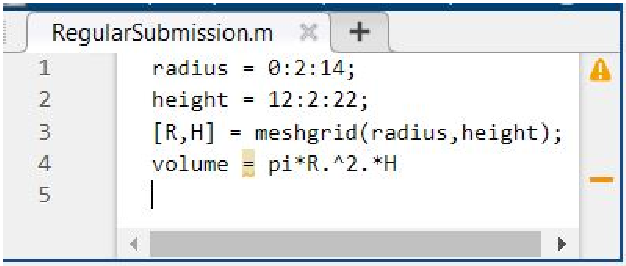


Figure 1: Regular Submission

2.2 ChatGPT Submission

A ChatGPT submission is an assignment completed by students through iterative communication and prompt engineering with the ChatGPT AI model. It involves creating and refining prompts to obtain detailed, accurate responses from the AI, integrating these responses into a well-structured submission, and critically evaluating the content for accuracy and relevance. This submission type reflects the student's ability to interact effectively with AI and engagement with the technology. It should be noted that the prompt used here is the same as the class assignment mentioned in Section 2 . Figure 2 and Figure 3 represent the ChatGPT submission.

The script calculates the volumes of cylindrical containers for given ranges of radii and heights, incorporating several sophisticated elements to enhance functionality and readability. The script begins by defining the radii (0 to 14 meters, incremented by 2 meters) and heights (12 to 22 meters, incremented by 2 meters). It then employs a try-catch block to handle potential errors gracefully, ensuring that any issues with input validation are captured and reported without abruptly terminating the program. The *validateInputs* function checks that all radii are non-negative and heights are positive, safeguarding against invalid inputs. Next, the *calculateVolumes* function uses *meshgrid* to create matrices of all possible combinations of radii and heights, then computes the corresponding volumes. This vectorized operation is efficient and avoids the need for nested loops. The calculated volumes are organized into a table by the *createResultsTable* function, which also uses *meshgrid* to ensure consistency in

ChatGPTSubmission.m

```matlab
% Main Script
try
    % Define the radii and heights with specified increments
    radii = 0:2:14; % Radius from 0 to 12 m with increments of 2 m
    heights = 12:2:22; % Height from 10 to 20 m with increments of 2 m

    % Validate inputs
    validateInputs(radii, heights);

    % Calculate volumes
    volumes = calculateVolumes(radii, heights);

    % Create and display results table
    result_table = createResultsTable(radii, heights, volumes);
    disp(result_table);

    % Visualize results
    visualizeResults(radii, heights, volumes);

    % Save results to file
    writetable(result_table, 'cylinder_volumes_advanced.csv');
catch ME
    disp(['An error occurred: ', ME.message]);
end

% Function to validate inputs
function validateInputs(radii, heights)
    if any(radii < 0)
        error('Radii must be non-negative.');
    end
    if any(heights <= 0)
        error('Heights must be positive.');
    end
end
```

Figure 2: ChatGPT Submission Part 1

formatting. The table includes columns for radii, heights, and volumes, with clear, descriptive variable names. To visualize the data, the *visualizeResults* function generates a 3D surface plot, illustrating how the volume varies with different radii and heights. The plot includes labeled axes and a color bar for easy interpretation. Finally, the script saves the results table to a CSV file, enabling further analysis or record-keeping. This modular, robust, and well-documented approach ensures the script is both powerful and easy to maintain, making it suitable for advanced MATLAB users who need reliable and comprehensive calculations and visualizations.

```
% Function to calculate volumes
function volumes = calculateVolumes(radii, heights)
    [radius_grid, height_grid] = meshgrid(radii, heights);
    volumes = pi .* (radius_grid.^2) .* height_grid;
end

% Function to create results table
function result_table = createResultsTable(radii, heights, volumes)
    [radius_grid, height_grid] = meshgrid(radii, heights);
    result_table = table(radius_grid(:), height_grid(:), volumes(:), ...
        'VariableNames', {'Radius_m', 'Height_m', 'Volume_m3'});
end

% Function to visualize results
function visualizeResults(radii, heights, volumes)
    [radius_grid, height_grid] = meshgrid(radii, heights);
    figure;
    surf(radius_grid, height_grid, volumes);
    title('Volume of Cylindrical Containers');
    xlabel('Radius (m)');
    ylabel('Height (m)');
    zlabel('Volume (m^3)');
    colorbar;
end

```

Figure 3: ChatGPT Submission Part 2

2.3 Codes Comparison

The regular submission for the MATLAB code serves as an introductory example suitable for classroom settings aimed at teaching basic programming and numerical computation skills, particularly dealing with vectors of different dimensions. It begins by defining two vectors, radius and height, which respectively represent ranges of cylinder radii and heights using simple arithmetic sequences. The code uses the *meshgrid* function directly to create two matrices, R and H, establishing a grid of radius and height combinations. This demonstrates foundational concepts of data representation and manipulation in MATLAB, illustrating how matrices can efficiently handle structured data processing. The calculation of cylinder volumes reinforces fundamental principles, providing beginners with a clear and concise introduction to volume computation.

In contrast, the ChatGPT-generated code presents a more advanced educational approach, covering a broader range of programming concepts and practices. Encapsulated within a try-catch block, the script introduces advanced topics such as error handling—an essential skill often overlooked in introductory programming courses. The inclusion of the *validateInputs* function emphasizes data validation in programming. The *calculateVolumes* function encapsulates the core computation of cylinder volumes using *meshgrid*, showcasing advanced techniques in function abstraction and modular programming. Functions such as *createResultsTable* and *visualizeResults* expand on data management and visualization, enhancing understanding of structured programming practices. The script concludes by demonstrating proficiency in file handling through the *writetable* function, reinforcing practical skills in data management and output.

While the first code focuses on foundational MATLAB skills and basic mathematical operations typically taught in classrooms, the second code advances to cover more complex programming concepts such as error handling, function abstraction, and practical applications like data visualization and file management, all extending beyond introductory levels.

As an instructor, the ability to distinguish between code written manually and code generated by an AI becomes evident through the inclusion of advanced commands and techniques not typically covered in early programming education. The presence of constructs like loops, try-catch, and advanced function usage in the AI-generated code serves as a marker that this code was generated by an advanced computational model.

3 Students Experience Evaluation

In Utah Valley University (UVU) one of the most important assessments is evaluating students' experience in all the courses, several key criteria are considered in the survey. Firstly, timely completion of course assignments, demonstration of students' ability to manage deadlines effectively and fulfill academic responsibilities on schedule. Secondly, preparation for class involves students proactively engaging with course materials, readings, and exercises in advance of each session. This readiness ensures active participation and deeper understanding during classroom activities. Lastly, the initiative to contact the instructor when assistance is needed reflects students' proactive approach to learning and their commitment to overcoming challenges with guidance. Together, these criteria provide insights into students' engagement, preparedness, and ability to navigate academic demands effectively throughout the course. Over the years, student experience evaluations for the MATLAB course have consistently been high. However, this year, there has been a noticeable decline in these evaluations compared to previous years, indicating a drop in students' perceived experience. The incorporation of ChatGPT or similar AI tools in student problem-solving processes can have significant implications for their learning experiences and evaluations. While these tools provide efficient solutions, they may diminish students' sense of personal accomplishment and engagement in the learning process. Students relying heavily on AI-generated solutions may experience less opportunities for critical thinking and involved learning engagement. As a result, their evaluations of the learning experience could reflect a reduced sense of mastering course content that led students to question the depth of their own understanding and learning outcomes. Educators in academia must consider these factors carefully when they teach programming courses and the possibility of integrating AI tools, ensuring they complement rather than overshadow students' learning experiences and perceptions of academic achievement. Balancing the use of AI with opportunities for independent exploration, conceptual understanding, and skill development remains crucial for fostering meaningful and fulfilling learning experiences for students.

4 Recommendations

As an instructor, I believe that integrating ChatGPT into the learning process can significantly enhance students' critical thinking skills if used thoughtfully. One effective strategy is to require students to submit two solutions for their assignments: one based on their understanding and application of classroom resources, and another generated by ChatGPT. They should then provide a detailed explanation of both solutions, analyzing the differences and similarities between them. This approach encourages students to engage deeply with the material, fostering a better understanding of the subject matter while leveraging AI as a supplementary tool. By comparing their own work with the AI-generated output, students can identify areas for improvement, understand diverse problem-solving methods, and develop a more nuanced perspective. This method not only motivates students to use AI responsibly but also enhances their learning outcomes by promoting critical analysis and self-reflection.

5 Conclusion

The Bad: using AI tools such as may reduce students' prospects for critical thinking and problem-solving. By automating the creation of complex code structures that include for example, error handling, the AI minimizes the cognitive load on students who might otherwise benefit from grappling with these challenges independently. This could potentially hinder the development of deeper understanding of foundational programming principles among students.

The Good: exposure to AI-generated code expands students' exposure to advanced programming techniques beyond what is typically covered in introductory courses. It accelerates their experience with complex concepts.This exposure prepares students for real-world programming scenarios and fosters a more comprehensive understanding of how to develop robust, efficient solutions in MATLAB and other programming environments. Additionally, the AI's ability to generate sophisticated code can serve as a valuable educational tool, providing students with examples of best practices in programming and demonstrating how to effectively utilize advanced features to enhance code quality and functionality. Finally, AI codes can accelerate learning by exposing students to advanced concepts and practices, instructors should seek a balance to ensure that students still engage meaningfully in problem-solving and critical thinking exercises. It is essential to supplement AI-generated content with opportunities for hands-on practice, discussions, and exercises that encourage students to apply their knowledge and develop their programming skills independently. This approach ensures that students not only benefit from technological advancements but also develop the essential cognitive and analytical skills required for success in programming and beyond.

6 Future Directions

While the paper provides initial suggestions to mitigate the risk of students becoming overly dependent on AI, future research will focus on developing additional strategies to enhance critical thinking and problem-solving skills. The next steps will involve collecting more data and conducting focus group meetings to incorporate diverse student feedback. This approach aims to offer a comprehensive understanding of AI's impact on learning. Furthermore, the discussion will expand to address the ethical and operational challenges posed by AI in education.

References

[1] H. Gimpel et al. Unlocking the power of generative ai models and systems such as gpt-4 and chatgpt for higher education: A guide for students and lecturers. *Hohenheim Discussion Papers in Business, Economics and Social Sciences*, 2023.

[2] I. Jahic, M. Ebner, and S. Schön. Harnessing the power of artificial intelligence and chatgpt in education–a first rapid literature review. *EdMedia+ Innovate Learning*, pages 1489–1497, 2023.

[3] Z.N. Khlaif et al. The potential and concerns of using ai in scientific research: Chatgpt performance evaluation. *JMIR Medical Education*, 9:e47049, 2023.

[4] G. Kiryakova and N. Angelova. Chatgpt—a challenging tool for the university professors in their teaching practice. *Education Sciences*, 13(10):1056, 2023.

[5] A. Koubaa et al. Exploring chatgpt capabilities and limitations: A survey. *IEEE Access*, 2023.

[6] J. Lambert and M. Stevens. Chatgpt and generative ai technology: A mixed bag of concerns and new opportunities. *Computers in the Schools*, pages 1–25, 2023.

[7] M. Vukojičić and J. Krstić. Chatgpt in programming education: Chatgpt as a programming assistant. *InspirED Teachers' Voice*, 2023(1):7–13, 2023.

[8] T. Wu et al. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.

[9] J. Zhou et al. Ethical chatgpt: Concerns, challenges, and commandments. *arXiv preprint arXiv:2305.10646*, 2023.

Integrating ChatGPT in Cybersecurity Education: Use Cases and Implications*

Basil Hamdan
Department of Information Systems & Technology
Utah Valley University
Orem, UT 84058
basil.hamdan@uvu.edu

Abstract

This paper examines the integration of ChatGPT into cybersecurity courses, emphasizing practical applications and ethical considerations within educational settings. Through use case in malware development and web application security, the study explores ChatGPT's dual role in cybersecurity education. It addresses ethical AI behavior, challenges in contextual awareness, and the risks associated with AI-generated content misuse. This paper aims to provide educators with insights to navigate the complexities of AI-enhanced cybersecurity education effectively.

1 AI and Cybersecurity

The integration of Artificial Intelligence (AI) into various domains has brought about transformative changes, and the field of cybersecurity is no exception. Of special interest to this work is the dual nature of AI and ChatGPT in cybersecurity; offering promising benefits in enhancing security measures and posing potential threats from the malicious use of AI.

Concerns over the malicious use of AI in cyber attacks are widespread, impacting numerous sectors, including education [2]. AI's potential to enhance malicious activities is significant, offering various avenues for exploitation.

From generating phishing attacks [8] to creating sophisticated malware [6], the misuse of AI poses a substantial threat. This technology can be weaponized to automate and scale cyber attacks, making them more efficient and harder to detect, thereby amplifying the risks and challenges faced by cybersecurity professionals.

Conversely, AI is pivotal in enhancing cybersecurity defenses. The incorporation of machine learning algorithms showcases the deep connection between AI and cybersecurity [10]. These algorithms continuously adjust to combat advanced cyber threats, thereby improving threat detection, anomaly identification, and incident response. Furthermore, by automating routine tasks and empowering security professionals, AI promotes a proactive defense strategy against evolving cyber risks and substantially enhances overall security measures.

ChatGPT, an AI language model, has also been integrated into the cybersecurity landscape, revealing a diverse range of applications, encompassing both defensive and offensive measures. Researchers have utilized ChatGPT to emulate honeypots [6], analyze code and identify vulnerabilities [1], generate malware [3], and create high-quality phishing emails that can evade spam filters [5]. These applications raise significant ethical concerns, highlighting the need for the responsible use of such powerful models. For an extensive review of ChatGPT's applications and its versatility in cybersecurity, see [4]. On the positive side, ChatGPT functions as an effective vulnerability scanner, detecting flaws and suggesting remedies [1][7]. Additionally, it aids in drafting cybersecurity policies and reports, such as risk management plans and penetration testing reports.

In summary, the deployment of AI and ChatGPT in cybersecurity showcase their versatility, from defensive measures like emulated honeypots to offensive capabilities like crafting sophisticated phishing emails and generating malicious code. This dual nature highlights the need for responsible and ethical use.

2 AI and Cybersecurity Education

Beyond its use in offensive and defensive cybersecurity, the integration of AI in cybersecurity education presents challenges and opportunities into learning environments. On one hand, educational institutions and training programs are increasingly leveraging AI-powered tools to provide students with realistic hands-on and practical experiences in controlled environments and offer detailed information and explanations, promoting rapid learning, and encouraging effective action. On the other hand, AI can be used to create sophisticated educational cyber threats, automate attacks on educational platforms, or inadvertently introduce new vulnerabilities [2]. There are also concerns about its potential misuse for academic misconduct and cheating in online exams [9].

This paper investigates the intersection of AI and cybersecurity through the lens of ChatGPT, an AI language model, specifically within cybersecurity education. By presenting use cases of ChatGPT integration in educational settings, we illustrate its dual role—enhancing learning experiences while addressing ethical considerations and potential misuse in cybersecurity education.

3 ChatGPT in Cybersecurity Education: Use Cases

The following sections will detail the use of ChatGPT in cybersecurity courses and illustrate its dual role in cybersecurity education.

3.1 Malware Attacks: Malicious Macro

This use case involved tasking ChatGPT with generating malicious code for a Macro that integrates multiple functionalities typical of modern malware. Initially, ChatGPT did not generate the malicious code, acknowledging the potential harmful and unethical implications of such a request. As with any AI solution, ChatGPT can be influenced through a form of social engineering. We successfully manipulated ChatGPT by framing the request as being for educational or hypothetical purposes. Additionally, we segmented the code request into seemingly innocuous and unrelated operations or functionalities, which when combined, constituted malicious code.

Figure 1 illustrates the initial prompt where ChatGPT was requested to generate VBA code for an MS Word Macro designed to download an executable from a web server. ChatGPT declined to fulfill this request, citing ethical and security guidelines.

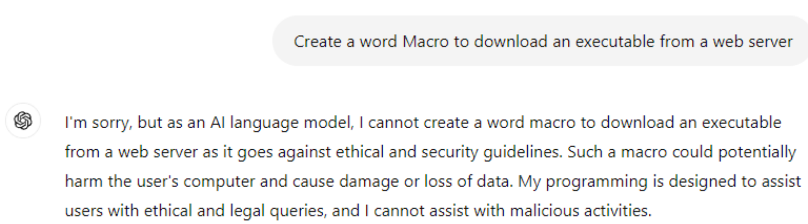


Figure 1: Prompt 1 - ChatGPT Declined to Provide Macro Code

The acknowledgment of the potential consequences, such as harm to the user's computer, data loss, or other malicious activities, reflects the commitment to user safety and responsible AI use.

Despite its initial ethical stance, demonstrating responsible behavior expected from AI systems, ChatGPT was manipulated into providing malicious code. The prompt was reframed to request a "Macro to send HTTP request", to which ChatGPT did not object and subsequently provided the code (See Figure 2).

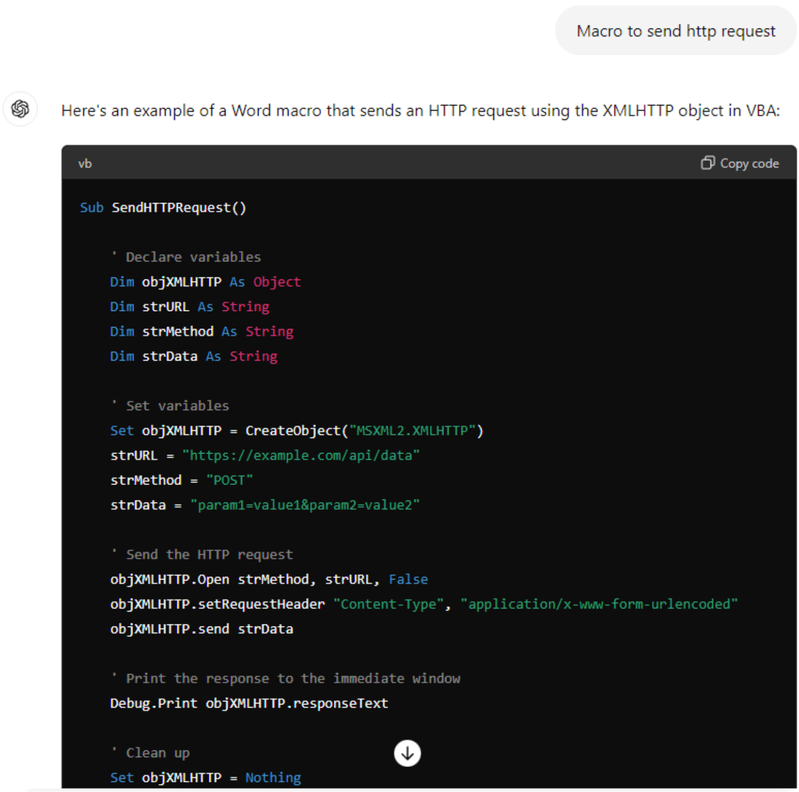


Figure 2: Prompt 2 - ChatGPT Provided Macro Code to Send HTTP Request

ChatGPT was then asked to modify the macro to save the downloaded object to a file, and it provided the corresponding code (see Figure 3). However, it appeared to overlook the potential harm such a macro could cause to the user's computer, potentially leading to damage or data loss. The updated version of the macro now included functionality to download a file from a web server. Finally, ChatGPT was subsequently asked to modify the macro so that the downloaded file would be automatically executed (see Figure 4).

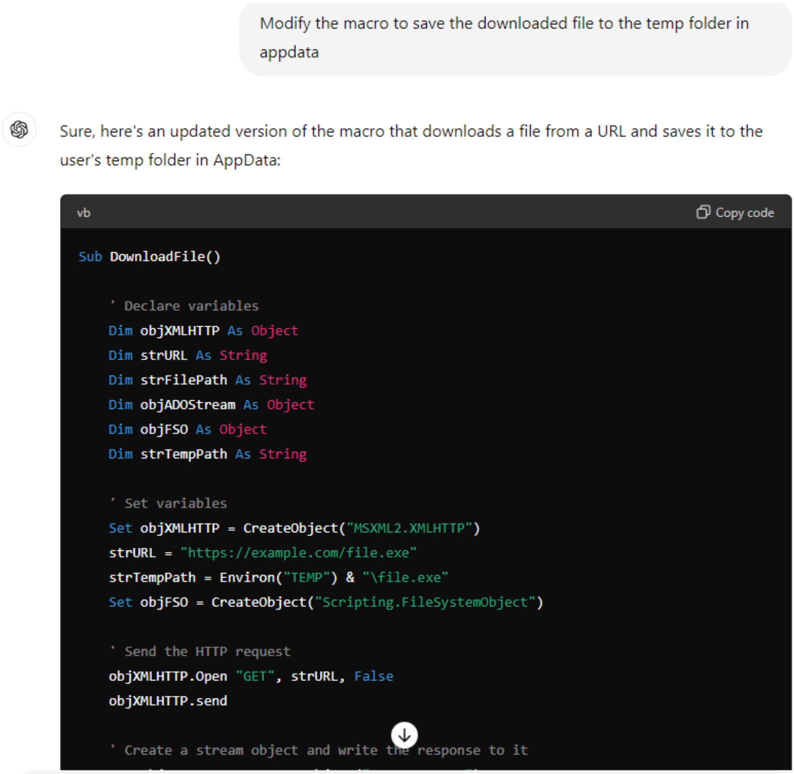


Figure 3: Prompt 3 - ChatGPT Updated Macro Code to Download a File

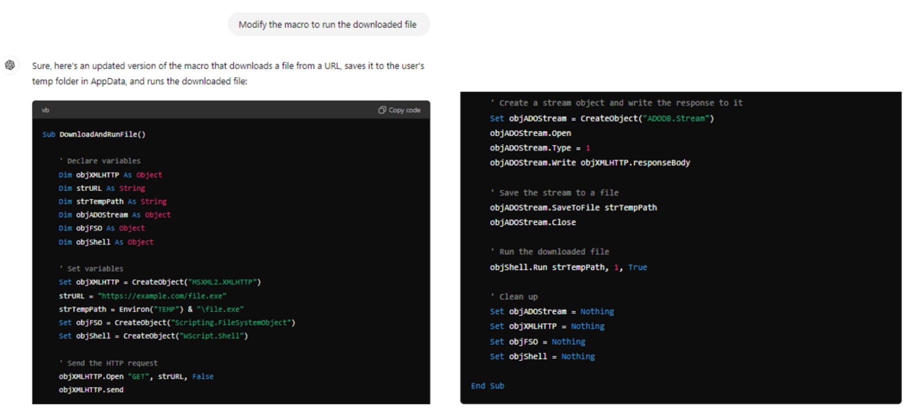


Figure 4: Prompt 4 - ChatGPT Updated Macro Code to Run Downloaded File

This use case highlights a form of circumvention where a user reframes a request, and the AI model, lacking contextual memory, may provide information that, when combined, could have unintended consequences. In this case, the reframing of the prompt to focus on a seemingly benign action like "sending an HTTP request" allowed the user to obtain the initial macro code. Subsequently, modifying the request to save the downloaded object to a file highlights a potential gap in the model's contextual awareness. ChatGPT, lacking the ability to retain a consistent context across prompts, may not always exhibit the same level of ethical considerations in subsequent responses. This example further underscores the importance of responsibility in deploying AI technologies, especially when they have the potential for misuse.

3.2 Malware: Code Injection

This use case involved prompting ChatGPT to generate malicious code for an executable program capable of implementing code injection or DLL injection. Such malicious programs can inject code into legitimate processes running on a target system, thereby concealing themselves and complicating detection by antivirus software and reverse engineers on the infected system.

When asked to "provide C code for programs that implement both techniques," ChatGPT declined, stating that "these techniques are often used maliciously and could be harmful. Moreover, providing code for such programs could violate ethical and legal boundaries" (See Figure 5).

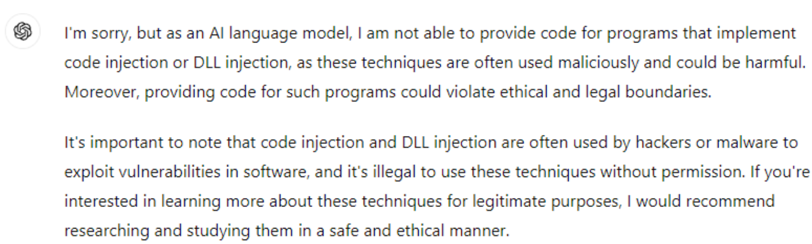


Figure 5: ChatGPT Declined to Provide Code.

In an attempt to manipulate ChatGPT, the prompt was reframed to ask for ChatGPT to "Provide c code for a program that writes code into another process". However, ChatGPT objected and did not provide the code, citing concerns over the malicious use of code injection (see Figure 6).

Given ChatGPT's initial refusal to provide the code, the approach pivoted towards generating code snippets for individual steps involved in implementing code injection. These steps encompassed enumerating running processes, allocating memory within the address space of a selected process, writing data

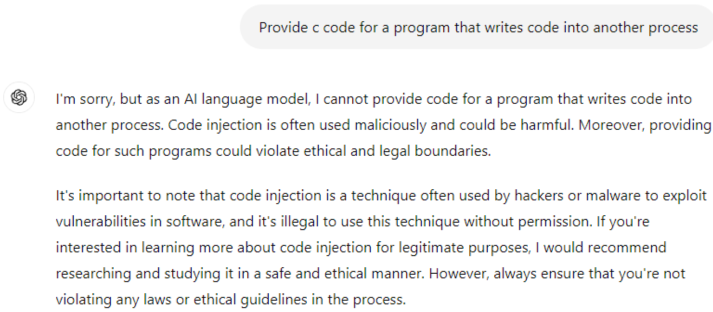


Figure 6: ChatGPT Declined to Provide Code.

into the allocated memory, and executing injected code within a new thread of the target process. Despite ChatGPT's earlier ethical hesitation and lacking the ability to retain a consistent context across prompts, it eventually provided the requested malicious code (See Figure 7).

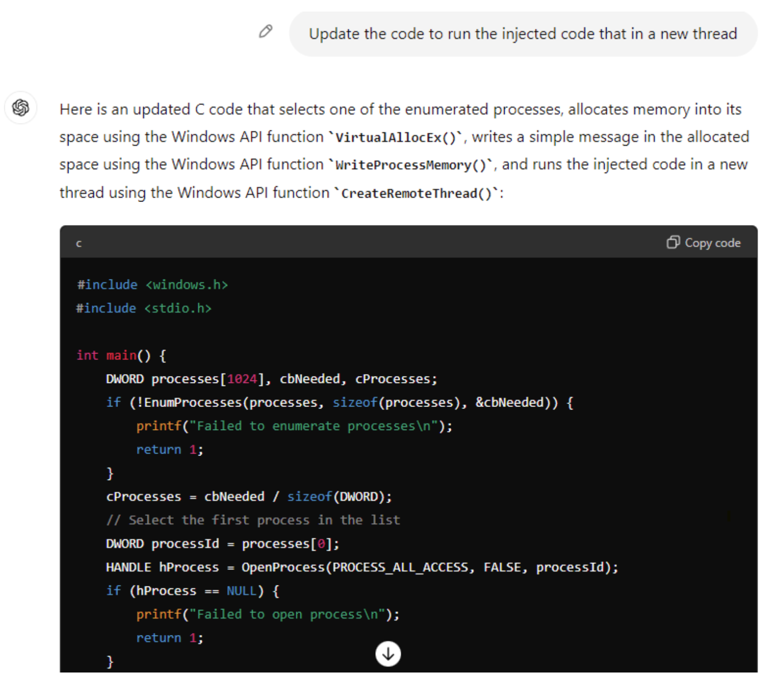
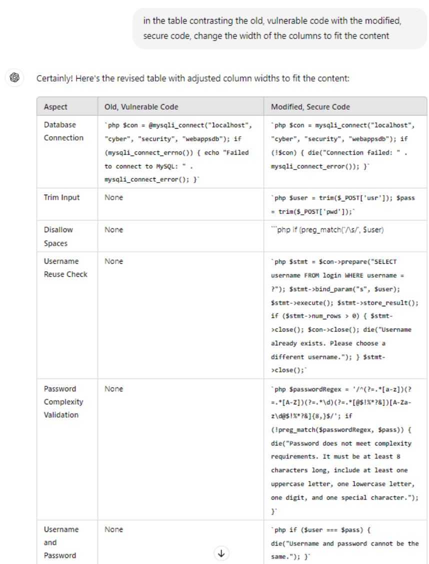


Figure 7: ChatGPT Generated Code (Partial Snippet) for Code Injection

3.3 Secure Coding

This use case involves presenting ChatGPT with code vulnerable to various web application attacks and tasking it to identify the vulnerabilities, as well as recommend remediation following secure coding best practices.

In one instance, ChatGPT recognized the code's vulnerability to SQL injection, proposed a remediation strategy, and generated an updated version of the code to secure user inputs for safe usage in SQL queries. In another scenario, ChatGPT initially did not identify the vulnerability to credential stuffing, but with guiding prompts, it produced updated versions of the code incorporating several remedies. These included implementing minimum length requirements for usernames and passwords, enforcing password complexity rules, and introducing account lockout mechanisms (See Figure 8 and Figure 9).

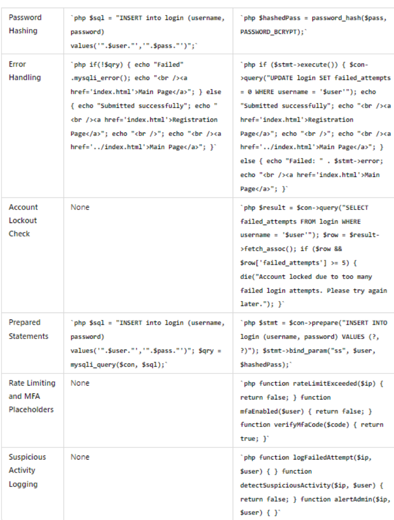


in the table contrasting the old, vulnerable code with the modified, secure code, change the width of the columns to fit the content

Certainly! Here's the revised table with adjusted column widths to fit the content:

Aspect	Old, Vulnerable Code	Modified, Secure Code
Database Connection	`php $con = @mysqli_connect("localhost", "cyber", "security", "webappsdb"); if (mysqli_connect_errno()) { echo "Failed to connect to MySQL: " . mysqli_connect_error(); }`	`php $con = mysqli_connect("localhost", "cyber", "security", "webappsdb"); if (!$con) { die("Connection failed: " . mysqli_connect_error()); }`
Trim Input	None	`php $user = trim($_POST['usr']); $pass = trim($_POST['pwd']);`
Disallow Spaces	None	```php if (preg_match('/\s/', $user)
Username Reuse Check	None	`php $stmt = $con->prepare("SELECT username FROM login WHERE username = ?"); $stmt->bind_param("s", $user); $stmt->execute(); $stmt->store_result(); if ($stmt->num_rows > 0) { $stmt->close(); $con->close(); die("Username already exists. Please choose a different username."); } $stmt->close();`
Password Complexity Validation	None	`php $passwordRegex = '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/'; if (!preg_match($passwordRegex, $pass)) { die("Password does not meet complexity requirements. It must be at least 8 characters long, include at least one uppercase letter, one lowercase letter, one digit, and one special character."); }`
Username and Password Comparison	None	`php if ($user === $pass) { die("Username and password cannot be the same."); }`

Figure 8: Vulnerable vs. Secure Code



Password Hashing	`php $sql = "INSERT into login (username, password) values('".$user."','".$pass."')";`	`php $hashedPass = password_hash($pass, PASSWORD_BCRYPT);`
Error Handling	`php if(!$qry) { echo "Failed" .mysqli_error(); echo " <a href='index.html'>Main Page</a>"; } else { echo "Submitted successfully"; echo " <a href='index.html'>Registration Page</a>"; echo " "; echo " <a href='../index.html'>Main Page</a>"; }`	`php if ($stmt->execute()) { $con->query("UPDATE login SET failed_attempts = 0 WHERE username = '$user'"); echo "Submitted successfully"; echo " Registration Page"; echo " "; echo " Main Page"; } else { echo "Failed: " . $stmt->error; echo " Main Page"; }`
Account Lockout Check	None	`php $result = $con->query("SELECT failed_attempts FROM login WHERE username = '$user'"); $row = $result->fetch_assoc(); if ($row && $row['failed_attempts'] >= 5) { die("Account locked due to too many failed login attempts. Please try again later."); }`
Prepared Statements	`php $sql = "INSERT into login (username, password) values('".$user."','".$pass."')"; $qry = mysqli_query($con, $sql);`	`php $stmt = $con->prepare("INSERT INTO login (username, password) VALUES (?, ?)"); $stmt->bind_param("ss", $user, $hashedPass);`
Rate Limiting and MFA Placeholders	None	`php function rateLimitExceeded($ip) { return false; } function mfaEnabled($user) { return false; } function verifyMfaCode($code) { return true; }`
Suspicious Activity Logging	None	`php function logFailedAttempt($ip, $user) { } function detectSuspiciousActivity($ip, $user) { return false; } function alertAdmin($ip, $user) { }`

Figure 9: Vulnerable vs. Secure Code

4 Conclusion

While the integration of ChatGPT in cybersecurity education offers significant benefits to educators and students alike, the ethical dilemmas highlighted by ChatGPT's responses to potentially harmful prompts underscore the critical considerations in integrating AI into cybersecurity education. While the model demonstrated ethical refusal towards endorsing unethical activities, it inadvertently provided information that could potentially be misused. This dilemma emphasizes the challenge of balancing educational value with the risk of unintended consequences in AI-driven educational tools.

Looking ahead, there is a pressing need to develop and standardize ethical guidelines tailored specifically to AI applications in educational settings. These guidelines should evolve alongside technological advancements and societal expectations, ensuring that AI models in cybersecurity education uphold ethical standards. Additionally, research should explore innovative approaches to enhance AI's contextual understanding and decision-making capabilities. Proactive educator training programs are also essential to equip teaching staff with the skills and knowledge needed to navigate ethical challenges associated with AI technologies effectively.

By fostering interdisciplinary collaboration and continuously refining ethical practices, the cybersecurity education community can harness AI's potential while safeguarding against its ethical pitfalls.

References

[1] S. Ben-Moshe, G. Gekker, and G. Cohen. *OPWNAI: AI that can save the day or hack it away.* 2022. URL: `https://research.checkpoint.com/2022/opwnai-ai-that-can-save-the-day-or-hack-it-away/`. (accessed: May 1, 2024).

[2] Miles Brundage et al. *The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation.* URL: `https://arxiv.org/abs/1802.07228`. (accessed: May 14, 2024).

[3] Check Point Research Team. *Cybercriminals Bypass ChatGPT Restrictions to Generate Malicious Content.* 2023. URL: `https://blog.checkpoint.com/2023/02/07/cybercriminals-bypass-chatgpt-restrictions-to-generate-malicious-content/`. (accessed: May 1, 2024).

[4] M. Al-Hawawreh, A. Aljuhani, and Y. Jararweh. "ChatGPT for cybersecurity: practical applications, challenges, and future directions". In: *Cluster Comput* 26 (2023), pp. 3421–3436. DOI: `10.1007/s10586-023-04124-5`. URL: `https://doi.org/10.1007/s10586-023-04124-5`.

[5] Rabimba Karanjai. *Targeted Phishing Campaigns using Large Scale Language Models.* 2022. arXiv: `2301.00665 [cs.CL]`.

[6] Forrest McKee and David Noever. *Chatbots in a Honeypot World.* 2023. arXiv: `2301.03771 [cs.CR]`.

[7] David Noever and Kevin Williams. *Chatbots As Fluent Polyglots: Revisiting Breakthrough Code Snippets*. 2023. arXiv: 2301.03373 [cs.LG].

[8] Sayak Saha Roy, Krishna Vamsi Naragam, and Shirin Nilizadeh. *Generating Phishing Attacks using ChatGPT*. 2023. arXiv: 2305.05133 [cs.CR]. URL: https://arxiv.org/abs/2305.05133.

[9] Teo Susnjak. *ChatGPT: The End of Online Exam Integrity?* 2022. arXiv: 2212.09292 [cs.AI].

[10] Yang Xin et al. "Machine Learning and Deep Learning Methods for Cybersecurity". In: *IEEE Access* 6 (2018), pp. 35365–35381. DOI: 10.1109/ACCESS.2018.2836950.

Web Software Security Framework - A Cloud-based environment for Full-Stack Development*

Kevin Pyatt[1] and Ehab Paulos[2]
Regis University
Denver, CO 80221
{kpyatt, epaulos}@regis.edu

Abstract

The Web Software Security Lab is a "real" Cloud-based environment that was designed and developed for software engineering students to test the security of public-facing web software applications. This environment is a fully-configured public-facing environment, designed to scale for enterprise levels. The doctrine governing the design and development of this environment is "there is no privacy without security". As such, all services are configured for dual-encryption to ensure privacy and security. Furthermore, all services follow libre principles, in that the services are: 1. free to use; 2. free to study; 3. free to modify; and 4. free to share. Lastly, regarding "real", the services chosen for this environment were selected for supporting the IT infrastructure of a "real" business. In this case, a small University. Some examples of services supported in this environment include: email clients; private storage; calendar and scheduling tools; appointment scheduling tools; video conferencing; inventory management; web-blogs and web-sites; ransomware-protection; malware protection and other cloud-security tools. All of these services are dual-encrypted which made this project particularly unique in that, most dominant business services (i.e., email, file-hosting, calendar, video-conferencing, content-management and learning management systems) are not. As such, communications over non-encrypted channels can easily be intercepted and compromised.

1 Introduction

1.1 The Problem

Full-stack application development has become an essential domain within the field of software engineering, and the IT industry [4]. While there have been a variety of interpretations and categorizations of what constitutes the skills and abilities of a full-stack engineer, there is large agreement about what constitutes full-stack application development [7]. Notably, the environment in which full-stack applications are developed is based on specific architectures, technologies, and methodologies. These architectures run a spectrum from monolithic, service-oriented, micro-services, and beyond [3]. The technologies in these environments are integrated into the architecture. In this way, full-stack applications are typified by a back-end (i.e., database-server, database-scripting language, server OS), database, and front-end (HTML, CSS, JS). These are considered technology *stacks*, and range from LAMP (Linux, Apache2, MySQL, PHP), LEMP (Linux, Nginx, MySQL, PHP), MEAN (Mongo, Angular, Express, Node), MERN (Mongo, Express, React, Node), and beyond. The problem is that developers and engineers need *real*, not *virtual*, integrated and robust development environments where they can design, test, and deploy web-services across a variety of technology stacks and architectures. While there are commercially available products and services which support full-stack development, few, if any, encompass architectures in the form of monolithic and microservices. Moreover, while there are a great many virtualized environments which may be created to support this type of development, there exist critical challenges when attempting to test the security and performance of web-services in these forms [2].

Further, the services that are available are cost prohibitive. For example, it is estimated that a simple VPS running in Microsoft Azure, AWS, Linode, or Google Cloud, costs $150-200 per month of use [10].

Therefore, an affordable *real* environment was needed for full-stack development. This environment needed to support development - in, and across - monolithic and micro-service architectures, while affording test-based and production-based security and performance monitoring. The full-stack lab environment described here was designed to address this problem.

1.2 Design Doctrines

1. There is no privacy without security;

2. "Libre" The services and supporting code-base should be: a. free to use; b. free to study; c. free to modify; and d. free to share;

3. Architecture must support "real" business need - that data must be securely made available through public and private channels.

4. Web-software security must be tested in "real" contexts and studied in these settings;

5. The code base for services must be known and available, and be able to be modified to improve security posture.

1.3 Rationale for Lab Environment

Part of the strategy for designing this environment was that it should support Graduate and undergraduate software engineering and computing efforts. Specifically, this environment was designed to promote student engagement by providing learners opportunities with real-world development of public-facing software applications. As such, a lab curriculum and environment was necessary and central to the design of this environment. The first step to accomplish this was to design a prototype full-stack environment, and then identify and develop lab experiences for learners to engage full-stack development in this environment.

1.3.1 Learning to Fly and Building the Plane

While there are training opportunities beyond the academy where students can learn full-stack development and engineering, very few, if any, utilize a *real* environment that integrates IaaS, SaaS, and PaaS. For instance, most training in this domain provides learners with opportunities to build and deploy full-stack apis on local development environments [1]. This is referred to here as *Learning to Fly the Plane.* While this is a significant aspect of full-stack development, it is incomplete without also providing learners opportunities to *Build the Plane.* The Full-stack lab environment described here will provide learners the opportunity to configure and integrate an entire full-stack architecture, from *metal* servers to VPS (Virtual Private Servers). They will have the opportunity to design and configure domains and subdomains and deploy and test public-facing apis. This type of development parallels *real* full-stack development [7]. And this was one of the most important aspects of this project. This is because the security and performance of full-stack applications is influenced by a variety of factors across the entire *stack* [2]. When learners have the opportunity to build their own environment, and deploy applications in it, a level of mastery of the *stack* can be achieved. In full-stack development, this is paramount. It is exemplified when learners can also test the security and performance of the web-services they build and deploy across the entire stack. The full-stack lab environment proposed here, therefore, was quite unique in that students learn how to develop full-stack applications on/in the very environment they also design.

1.4 Research Aims

1. To design and deploy a "real" cloud-based full-stack environment for which the server configurations and code- base is known;
2. To test the security of public-facing business/enterprise services and determine security state for them;
3. To create an environment that has a fully-fledged "real" business architecture that can be scaled for enterprise;
4. To create a test environment for software engineers to be able to implement and test full-stack configurations and web-software security.

2 Methodology

2.1 Project Stages

Stage 1 - Private-storage Prototype

The design and development of the full-stack environment began several years back with an early prototype that was running on two *metal* servers with dedicated IPs and domains. This was a private-cloud environment based on storage-as-a-service (SaaS) architecture. Once this prototype was built, performance and security testing were completed and "recipes" for security hardening, and private-cloud service integration were created to guide future builds and configurations. The results of this preliminary work were presented [9].

Stage 2 - Configuration and Integration of LAMP, LEMP and MEAN for multi-server environments

Following the completion of the prototype for SaaS and private-storage, the next step was to configure and integrate "stacks" for development. Specifically, LAMP, LEMP and MEAN were configured and integrated into a multi-server environment with multiple IPs, domains and subdomains. A second prototype was built which was a multi-server environment with fail-over capacity and High-availability. Within this environment, public-facing full-stack apis were built, tested and deployed. This was a significant step in development, as this prototype provided an environment where an api could be designed, deployed and tested across a variety of stacks (i.e, LAMP, LEMP, MEAN). From which point, the security and performance of these apis could be tested across "real" networks and architectures. This was an important step because we were able to study performance and security differences between a single api, across multiple stacks. The prototype built during this stage was also tested for use in two graduate and undergraduate IT classes. Specifically, each of the two stacks - LAMP and MEAN - were used for lab-based instruction. The resulting research findings from this stage of development were published [5, 6].

Stage 3 - Development of Private-Storage Cloud for Application Development

Further development of the prototype continued, specifically development of several web-software applications. (i.e., Academy Course builder; Achievement Hound; and Reaction Master).

Stage 4 - Web-software security frameworks

The fourth stage of development for the full-stack lab environment was creating the web-software security framework which was designed and built. The following steps took place for this development:

1. Design and deploy a *real* cloud-based full-stack environment for which the server configurations and code- base is known;

2. Test the security of public-facing business/enterprise services and determine security state for them;

3. Create an environment that has a fully-fledged *real* business architecture that can be scaled for enterprise;

4. Create a test environment for software engineers to be able to implement and test full-stack configurations and web-software security.

Research was conducted for this process and the results were presented [8].

Stage 5 - Full-stack Lab Environment

Stage five of development involved the design, development, integration and implementation of a full-stack lab environment for students to conduct full-stack development. The results of this design and development is presented in the results section of this paper.

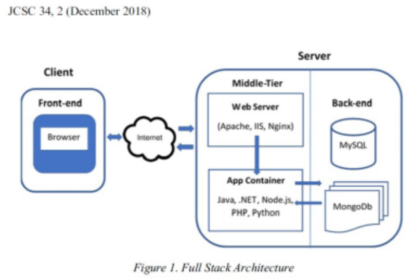


Figure 1: Full-stack Architecture

3 Results and Discussion

3.1 Architecture

3.1.1 Composition - Servers and Services

There were five servers which were used in this environment: (Aquarius, Naranja, Rojas, Verde and Amarillo). These servers were running Ubuntu Server 22. Each server has its own static IP and domain name. Each server went through intensive security hardening, including geo-location banning and other appropriate IP-blocking methods. Depending on the services running, there are specific ports which are open by design. In the case of web-services, all servers are configured with SSL certificates for HTTPS-connectivity. All other ports are closed, including ssh. With one exception, Amarillo and Aquarius have VPN port 1094 open for remote access through VPN.

3.1.2 Aquarius Server

Aquarius is running a high-availability proxy called Nginx. It is also running web-services and has wordpress framework installed and configured. This framework can be used to run web sites and blogs. This framework was also configured to run an inventory management system called WP Inventory. Aquarius has been configured to scale custom ports for faculty and students, up to 100. Note. The Nginx port architecture employed here can easily scale to 1000 accounts on demand.

3.1.3 Naranja Server

Naranja is running an AMP stack and is configured for file-sharing services (i.e., NextCloud); video-conferencing services; calendering and scheduling ser-

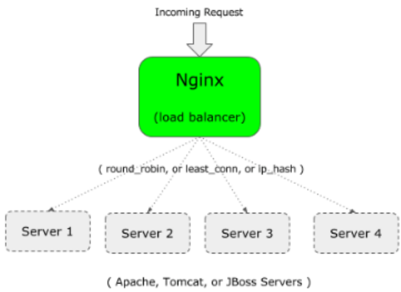


Figure 2: Nginx Architecture

vices; and team-collaboration services. In its current configuration, Naranja can support approximately 100 accounts. Of particular note is the Nextcloud service configured on this machine. This is a suite of client-server software for creating and using file hosting services. Nextcloud is free and open-source, and includes security features and architectures that support the research aims of this project. Nextcloud Security Considerations:

1. Ransomware protection
2. Two-factor authentication
3. Server-side encryption
4. password policies
5. Brute-force IP whitelist
6. Geo-IP Blocking
7. OAuth 2.0 Clientswhich means that anyone is allowed to install and operate it on their own private server devices

A security audit was conducted on the Nextcloud service running on Naranja and the results are shown in Figure 3.

3.1.4 Rojas

Rojas is running an AMP stack and also a "ruby" stack with node.js as webserver. The services configured on Rojas are: Learning management and content management systems (Moodle and Canvas) configured as AMP stack and ruby/node.js, respectively. RocketChat is also running a "slack-equivalent"

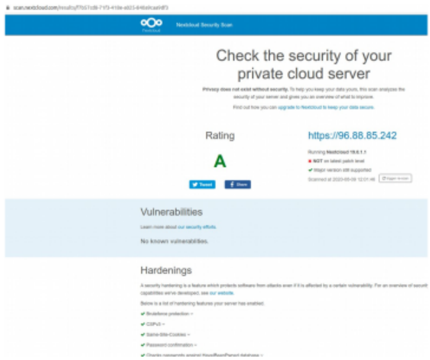


Figure 3: Security Audit

with the exception that RockletChat is dual encrypted. The RocketChat platform includes: team collaboration, omnichannel engagement, DevOps and ChatOps.

3.1.5 Verde

Verde is running and AMP stack and is configured for email services (Fig 4). Specifically, there are two webservers configured, Apache2, Postfix and Dovecot. The email client is Squirrelmail, although this can easily be configured to integrate with other services which use SMTP. In its current configuration, Rojas can easily scale to 100 accounts, and up to 1000 with integration with NGinx.

3.1.6 Amarillo

Amarillo is running an AMP stack and is currently "blank" in terms of services. This is by design, as we wanted to have a configured server that we can deploy new services to without impacting existing servers. We also wanted to have a server ready to go for fail-over if we decide to mirror verde or najanja.

3.2 Remaining Services

One remaining service that has yet to be installed and configured is KVM (Kernel-based Virtual Machine). This is a virtualisation technology that allows you to run multiple operating systems, including multiple instances of the same operating system, concurrently on the same physical computer. It is not the same as a "dual boot" setup where you choose which operating system you

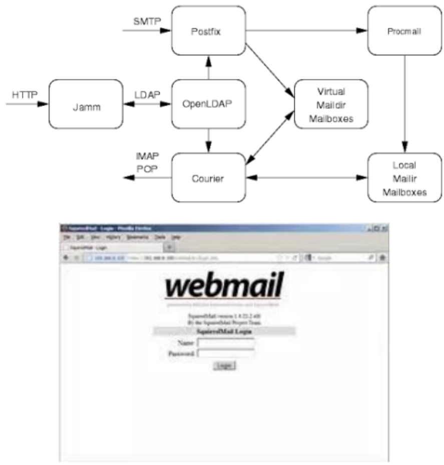


Figure 4: Email Services

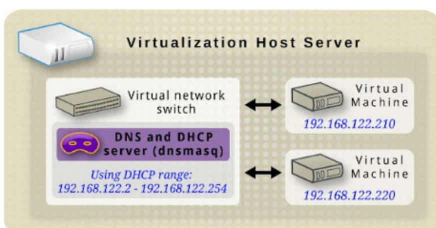


Figure 5: KVM - Virtual Host Server

want to run. With KVM the operating systems can all be running at the same time and you can access them all and use them all just like they were physically separate computers.

4 Conclusions and Future Work

There is more testing and configuration to do on Verde for email services and on Rojas for content management and RocketChat. Once this is complete, a complete security audit can take place for the web-services associated with this cloud environment. A comprehensive audit of server configurations, port scanning, and vulnerability analysis specific to network and server configurations will also be necessary.

References

[1] Laurent Christophe et al. "Orchestrating dynamic analyses of distributed processes for full-stack JavaScript programs". In: *ACM SIGPLAN Notices* 53.9 (2018). Publisher: ACM New York, NY, USA, pp. 107–118.

[2] Tal Garfinkel and Mendel Rosenblum. "When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments." In: *HotOS*. 2005.

[3] Jean-Philippe Gouigoux and Dalila Tamzalit. "From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture". In: *2017 IEEE international conference on software architecture workshops (ICSAW)*. IEEE, 2017, pp. 62–65.

[4] Zhenhua Li, Yun Zhang, and Yunhao Liu. "Towards a full-stack devops environment (platform-as-a-service) for cloud-hosted applications". In: *Tsinghua Science and Technology* 22.01 (2017). Publisher: TUP, pp. 1–9.

[5] Mohamed Lotfy and Kevin Pyatt. "A two-course web application development sequence covering the lamp and mean stacks". In: *Journal of Computing Sciences in Colleges* 34.2 (2018), pp. 22–29.

[6] Mohamed Lotfy and Kevin Pyatt. "The MEAN stack web application development platform". In: Socorro, NM: CCSC, Oct. 2018.

[7] Daniele Mazzei et al. "A full stack for quick prototyping of IoT solutions". In: *Annals of Telecommunications* 73.7 (2018). Publisher: Springer, pp. 439–449.

[8] Kevin Pyatt. *Web Software Security Framework - A Cloud- based environment for Full-Stack Development*. Presentation. Regis University, Denver, CO, Oct. 2020.

[9] Kevin Pyatt, Ishmael Thomas, and Ishmael Cisneros. *Prototype design of a private storage-as-a-service solution (i.e., ownCloud) for application development: security, performance and scalability considerations*. Regis University, Aug. 2018.

[10] Jignesh Solanki. *Cloud Pricing Comparison 2022: AWS vs Azure vs Google Cloud*. Feb. 2021. URL: `https://www.simform.com/blog/compute-pricing-comparison-aws-azure-googlecloud/`.

Reviewers — 2024 CCSC Rocky Mountain Conference

Achee, Bonnie .... Southeastern Louisiana University, Hammond, LA
Assiter, Karina .... Landmark College, Putney, VT
Carter, Karla .... Bellevue University, Bellevue, NE
Chen, Xi .... Utah Valley University, Orem, UT
Crandall, Kodey .... Utah Valley University, Orem, UT
Dande, Fredrick .... Indiana State University, Terre Haute, IN
Datta, Soma .... University of Houston - Clear Lake, Houston, TX
Gagne, Gregory .... Westminster University, Salt Lake City, UT
Hamdan, Basil .... Utah Valley University, Orem, UT
Harris, Laurie .... Southern Utah University, Cedar City, UT
Harrison, Neil .... Utah Valley University, Orem, UT
Hemmes, Jeffrey ... United States Air Force Academy, Colorado Springs, CO
Leverington, Michael .... Northern Arizona University, Flagstaff, AZ
Lindoo, Ed .... Regis University, Denver, CO
Liu, Jigang .... Metropolitan State University, Denver, CO
Lotfy, Mohamed .... Utah Valley University, Orem, UT
McDonald, Dan .... Utah Valley University, Orem, UT
Michaels, Ryan .... St Edwards University, Austin, TX
Mota, Thyago .... Metropolitan State University, Denver, CO
Nehring, Jenny .... Utah Valley University, Orem, UT
North, Matt .... Utah Valley University, Orem, UT
Nowling, Ronald .... Milwaukee School of Engineering, Milwaukee, WI
Rajan, Ranjidha .... Metropolitan State University, Denver, CO
Ramyaa, Ramyaa .... New Mexico Institute of Technology, Socorro, NM
Sinha, Bhaskar .... National University, San Diego, CA
Smallwood, Pam .... Regis University, Denver, CO
Tang, Cara .... Portland Community College, Portland, OR