

Teaching Responsible AI-Assisted Coding through Project-Based Learning and Collaborative Reflection

In submission. Do not distribute.

Michael D. Taylor

mdtaylor@andrew.cmu.edu

Carnegie Mellon University, Computer Science Department
Pittsburgh, PA, United States

Abstract

We describe a replicable project-based course focused on effective, responsible coding with generative AI, along with our observations of student growth, and a summary of students' written reflections. The course provides a low-stakes sandbox for students to explore genAI, its ethical complexities, and best practices for AI-assisted coding. Grading is based on effort and effective communication rather than artifact quality. Each weekly assignment builds the student's online portfolio of creative and technical projects, and its value as a career asset replaces grades as the primary motivator for students to produce exceptional work.

Due to the rapidly advancing capabilities of genAI, comprehensive best practices have yet to be broadly established. In lieu of these, students developed class-wide best practices through regular discussions and written reflections after each assignment. This semester's student reflections emphasized spec-driven development, separation of concerns, verification, and attention to security among other mature priorities.

In addition, lectures provide opportunities for exploring ethics, current events, and speculative fiction. Guest lecturers from industry and academia expose students to broader perspectives. It is our goal to produce students who responsibly evaluate and engage with emerging technology while thoughtfully considering its societal impacts.

We expect that the exploratory nature of the course and its focus on durable skills will maintain its relevance even as technology continues to advance rapidly. In presenting the following observations and behavioral data, we hope that the course can provide inspiration and can be easily adapted by other institutions.

CCS Concepts

• **Social and professional topics** → **Computer science education; Model curricula; Computing literacy**; • **Computing methodologies** → *Natural language generation*.

Keywords

Active Learning, Artificial Intelligence, Computational Thinking, Experience Report, Programming, Tools and Tool Use, Undergraduate Instruction

ACM Reference Format:

Michael D. Taylor. 2026. Teaching Responsible AI-Assisted Coding through Project-Based Learning and Collaborative Reflection: In submission. Do not distribute.. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 7 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction

At the time of writing, most undergraduate students at ANONYMIZED are at least aware of several major AI providers (such as OpenAI, Anthropic, and Google) and most regularly use a browser-based chatbot like chatGPT. Students self-report seeking AI assistance for homework, studying, composing emails, and personal projects, as described in Table 1. Meanwhile, our recent graduates almost unanimously report that their employers expect (and sometimes require) dramatic increases in productivity from AI use.

Widespread student use and high employer expectations create tension in fundamental courses where unsupervised student reliance on AI can lead to severely reduced learning outcomes. Some courses such as ANONYMIZED have shifted grade weights to almost exclusively consist of proctored assessments where students cannot use AI, while actively warning students that relying on AI for assignments can lead to reduced learning outcomes and low exam performance. In the last year, students seem to be taking these warnings more seriously, but many find it difficult to develop effective study habits when easy answers are always within reach.

Research suggests that lower AI literacy can predict higher receptivity to generative AI due to its seemingly "magical" ability to solve complex tasks. In particular, students with low AI literacy are more receptive to using AI tools to complete assignments [5]. This suggests that improving our students' AI literacy may help students to better self-regulate their AI use. Additionally, if we directly provide the opportunity to practice effective and responsible AI use, we can empower them with greater creative self-efficacy [4], in-demand skills, and exposure to important ethical considerations. Unlike many other subjects, however, there is not yet a set of broadly-accepted best practices for AI use. The way in which informed users interact with AI (and even the vocabulary they use to do so) is in constant flux. This suggests that learning environments for practical AI skills must be able to rapidly adapt to new models, platforms, and ideas. Additionally, learning activities must focus on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, Woodstock, NY

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/2026/06

<https://doi.org/XXXXXXX.XXXXXXX>

a diverse range of tools and strategies so that students can develop durable skills and the ability to quickly evaluate new technology.

The course we will describe was built as an experimental attempt to teach introductory effective coding with genAI in a way that we believe is both replicable and easier to rapidly adapt than many traditional course formats. The course is designed primarily for first-year students who have taken one rigorous programming course at or above the level of AP Computer Science A. The course structure draws inspiration from ANONYMIZED and the project-based learning approach of Harvey Mudd College and Franklin W. Olin College of Engineering.

2 Main Design Decisions and Rationale

We begin by describing our goals and key pedagogical and structural choices for the course.

- (1) **Create a designated space for AI exploration:** We believe that creating a new elective course is the best way to have students thoughtfully explore the subject without risking their core fundamentals. Adding genAI learning goals to existing courses may require a corresponding reduction of other material, and students may have difficulty self-regulating genAI use while developing their technical foundations. We messaged strongly that this course was a safe place to use genAI with very few restrictions, and they **must** practice restraint in courses where genAI reliance may pose a risk to their learning.
- (2) **Embrace community-based knowledge discovery:** It is likely that students will collectively discover and explore new strategies and tools faster than any one instructor. As such, the course is structured to encourage and facilitate exploration and knowledge sharing. We begin the semester by establishing a course-wide goal to build a class-wide set of best practices. We will then pass those best practices to next semesters' students as a starting point for their own iteration. In doing so, we hope to keep the course highly relevant and up-to-date with future advances while encouraging a prosocial learning environment.
- (3) **Keep grade risks low while valuing "failure":** We believe students will more readily try creative and unproven strategies if grading is based on thoughtful effort and participation, and if students are not primarily motivated by protecting their GPA. Without internal grade shame, students may also be more willing to discuss strategies and tools that they tried even if those strategies did not produce optimal results.
- (4) **Build career assets:** Each assignment in the course is designed to help students build and augment a web portfolio of their creative work. Students are highly concerned with finding internships and permanent employment. The portfolio approach provides a very strong incentive for students to produce high-quality work. Most assignments are left open-ended so that students can align their work with their interests. Also, our verbal assessments require students to present their work in the form of a mock job interview, which provides valuable practice and sends strong feedback for whether they can communicate effectively and transparently about their AI use.

- (5) **Provide ample space for exploration of ethics and societal impacts:** Class discussions and guest lectures encourage students to engage with diverse perspectives regarding the potential benefits and harms of generative AI. Students are encouraged to develop their own personal decision-making framework so that they can establish boundaries that align with their values.

3 The Learning Cycle

In its initial semester, the course enrolled roughly 50 students. This size is conducive to both interactive lectures, full class discussions, and smaller breakout groups. A typical week focuses on one or two application categories or genAI strategies, along with space for discussion of current events. Each week intentionally pushed students to explore farther beyond the material taught in their prerequisite programming fundamentals course, which encourages students to engage with genAI in increasingly organized and informed ways.

Assignment cycles typically lasted one week, and began with hands-on lectures, technology and strategy demonstrations, and discussions before students independently began creating their artifact. Following each due date, roughly half of the next lecture was allocated toward breakout discussions, after which each student was asked to fill out a reflection form. These reflections (discussed in more detail later) gathered data on how students used genAI, what they found effective or ineffective, and advice for future students who attempt similar projects. The reflection data was then used to seed a larger group discussion on what we collectively learned and how it could shape our process in future weeks.

The student-sourced reflections and advice subsequently shaped student behavior on later assignments. For example, after we introduced spec-driven development, several students attempted to heavily rely on genAI to write their SPEC.md file and subsequently observed that their agentic code output was prematurely and undesirably constrained. All spec-driven assignments had warned against asking genAI to write the spec, but students' judgment appears to have been more greatly impacted by the first-hand experience of their peers. We hypothesize that the rapid advancement of technology and the volume of articles promising to explain the "best ways to use AI" have made students less receptive to isolated recommendations and comparatively more receptive to strategies that have been validated and shared by multiple peers.

While we encouraged all students to actively participate in the larger group discussion, we observed that students who were more reluctant to verbally engage with their peers were often among the most thoughtful in terms of written reflections. We conclude that it is important to provide multiple avenues for students to contribute to the collective class knowledge, and will be considering additional ways to solicit and share peer advice in future semesters.

4 Student Model Access

Providing students with equitable access to suitable genAI tools will likely remain a major challenge. While many AI providers offer a free access tier, they are typically quite limited in terms of model selection and usage limits. Many schools and universities provide access or subsidies for AI tools, but these options may be limited to browser-based platforms or generally less flexible than typical

Pro-level subscription plans. This challenge becomes greater when one of the learning objectives of the course is to teach students how to evaluate the utility of multiple models and platforms, especially given the potential delay between a technology's release and its adoption by specific institutions.

Our solution for our course is unfortunately temporary. Roughly around April 2025, OpenAI, Google Gemini, and Claude began offering time-limited free or very-low-cost trials to their Pro plans for verified students. The details of these plans have changed over time, and occasionally these offers have been limited to specific institutions, but students were able to effectively use these trials to access frontier models from the major providers. Unfortunately this is a one-semester solution that relied on the high likelihood that students had not yet exhausted these offers. Before assuming access, We polled the students to verify that each of them had access to at least two of them, and we generally found that students had access to all three. In future semesters this is increasingly unlikely, and we anticipate that these trials may become more limited as the demand for AI continues to rise.

5 Initial student characteristics

The semester's first assignment included a survey to gather information about how students initially used genAI, which tools they relied on, and their general perceptions on where genAI has helped or hindered their peers. We observed that 95% of students listed Python as the language they felt most comfortable with.

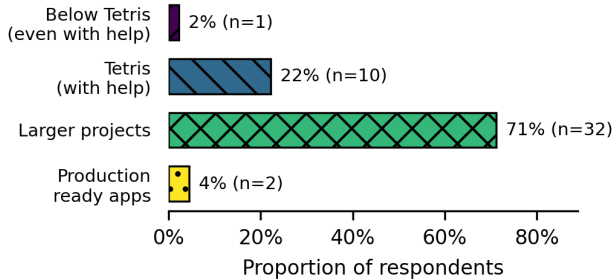


Figure 1: Initial self-assessed ability to create without AI (N=45)

Consistent with the course prerequisite, most students report that they would be capable of creating term projects typical of a rigorous introductory programming course without relying on genAI. However, Figure 1 also shows that roughly 22% of students feel that at most they could complete Tetris with assistance, despite having passed the prerequisite (with one exception who had special permission to enroll).

When asked "How often do you worry that you are not as proficient or experienced as others in your programming class(es)," students averaged 3.35 On a scale of 1 (never) to 5 (always). By contrast, when asked how often they believe their peers have the same worry, students replied with an average of 2.83. This supports our general observation that our students overestimate their peers' abilities in comparison with their own.

Additionally, we asked students how often they used genAI for various tasks, summarized in Table 1. Student responses were quite variable, and while it is evident that some students use genAI for each of these tasks, a larger-than-expected number of students report only occasional use.

6 Course Assignments

Our initial semester challenged students with 3 open-ended code production projects and 7 homework assignments. Of the homework assignments, 7 included code production. Here we provide brief descriptions of each assignment, and further details can be viewed on the course website. For assignments involving code production, the deliverables were submitted as a Github repository that typically included (at a minimum) all code, key prompts, and a README. Students were also asked to publish a description of their work and a repository link to their web portfolio. Some assignments also required an explanatory video and a reflection. Starting with HW2, each homework assignment was designed to provide experience working with a new technology or development concept. Projects provided students with a more open-ended opportunity to synthesize what they have learned. Student submissions and videos for all three projects can be viewed here: ANONYMIZED.

- **HW1: Initial reflections.** The initial survey on students' genAI use and their objectives for the course.
- **Project 1: Build a web portfolio.** Create and deploy a personal Github Pages website that includes biographical and contact information along with a project gallery. We chose this project because students would be heavily invested in creating a high quality website without errors, and because most students had no prior HTML experience. These factors meant that students would have to rely heavily on genAI while also being highly motivated to maintain control of the design and avoid hallucinations.
- **HW2: Crossy Road in an hour.** Within a strict 1-hour time limit, make as much progress toward a working Crossy Road implementation as possible. The game is similar to a 2.5D version of Frogger, and students chose to build incrementally or to write longer and more detailed prompts. The task was intentionally too large to fully complete within the allotted time, and students discussed their game state and strategies with each other afterward. Many who used higher-tier models found them to be too slow, while others found that faster models frequently introduced bugs that they could not rapidly fix, with the most satisfied students typically establishing the core mechanics first.
- **HW3: Explore an API.** Create something interesting and creative with any web API. This was the first very open-ended assignment, and students predictably felt challenged by the lack of familiar constraints. Nevertheless, students were largely very creative and successful when they chose a well-scoped idea, and the process seemed to raise their confidence for incorporating powerful tools into their projects. Several students chose to integrate API-driven features (maps, chatbots, etc.) into their portfolios. Students were cautioned that any exposed private API keys would result in a zero for

Table 1: Initial self-reported frequency of generative AI/LLM use (N=45).

Task	Almost never	Occasionally	More often than not	Almost always	Prefer not to say
Summarize an unfamiliar topic	9 (20%)	8 (18%)	23 (51%)	5 (11%)	0 (0%)
Check understanding of a topic	11 (24%)	9 (20%)	19 (42%)	5 (11%)	1 (2%)
Write code/text for course assignments	20 (44%)	19 (42%)	4 (9%)	1 (2%)	1 (2%)
Edit/debug for course assignments	10 (22%)	18 (40%)	9 (20%)	7 (16%)	1 (2%)
Write code/text for personal projects	7 (16%)	14 (31%)	14 (31%)	8 (18%)	2 (4%)
Edit/debug for personal projects	9 (20%)	9 (20%)	15 (33%)	10 (22%)	2 (4%)
Compose/improve emails or messages	11 (24%)	14 (31%)	10 (22%)	10 (22%)	0 (0%)
Explore how AI behaves/breaks	17 (38%)	14 (31%)	9 (20%)	4 (9%)	1 (2%)

the assignment, which incentivized learning about appropriate security precautions.

- **HW4: Frontend + Backend.** Create an app that relies on a deployed and secure backend. We believe this assignment was one of the two most challenging, largely because non-agentic chatbots cannot fully automate the deployment process. To constrain the solution space, we recommended that they use Render as their backend host, and we provided a video example. Students were frustrated with the time required to manually research and debug their issues, but most were able to meet their goals. The most common pattern for unsuccessful students was getting stuck in a prompting loop for hours and being too reluctant to seek help from faculty and teaching assistants.
- **HW5: Reading assignments.** Read "Cognitive Helmets for the AI Bicycle" [3] and one or more of the following: "Rossum's Universal Robots" [6], "Runaround" [2] or "The Evitable Conflict" [1]. This assignment coincided with the start of Project 2 and our series of lectures on ethics and the societal impacts of genAI. Students discussed the latter three readings in small groups before presenting their reflections to the class and participating in a lively group conversation about the predictive potential of science fiction. This assignment also coincided with a highly-engaging guest lecture from ANONYMIZED who studies the intersection of ethics and technology. Several students noted that this was the most personally transformative lecture of the semester.
- **Project 2: Web Apps.** This assignment challenged students to create a valuable and creative portfolio asset by synthesizing their learning and genAI skills from the previous weeks. Students were allowed to refine a prior assignment or pursue a new idea, and we saw a very wide variety of organizational tools and interactive visualizations.
- **HW6: SQLite App.** Learn database fundamentals with SQLite by creating an app that creates, retrieves, updates, and deletes data (CRUD). Given the focus of this assignment, students were encouraged to create a non-graphical CLI (command-line interface) app. This assignment coincided with a guest lecture from an alumni whose career involves addressing LLM shortcomings in formulating SQL queries.
- **HW7: Code Handoff.** Spend just 30 minutes creating the start of a game, and create documentation as you go. After half an hour, each student received the code from another randomly-assigned student and was instructed to continue

completing the game without substantially changing the existing code or structure. This assignment was meant to highlight the challenges of shared code creation while emphasizing the value of well-written documentation. Most students relied heavily on genAI to create their documentation and comments in the first phase, but very fortunately most students found that the code they received was documented in sufficient detail.

- **HW8: Spec-Driven Multi-Agent Development.** First write a detailed SPEC.MD describing an app for quiz prep, then ask an agent to create that spec, and finally use a new agent to review the output against the spec's acceptance criteria. This assignment exposed students to a simplified agentic workflow with careful up-front planning. While there was some disagreement, most students reported that spec-driven development was an effective approach that they would use on future projects. Others reflected that manually writing the spec was very challenging, and some resorted to asking genAI to write it for them, despite our warnings that an AI-generated spec would likely contain constraints that they do not understand or prioritize.
- **HW9: Phone Apps.** Use Expo Go and React Native to create and install an app on iOS or Android. As we predicted, this was widely reported to be one of the two most difficult assignments, due to comparatively complex file organization, new syntax, and deployment challenges. Successful students often wrote a SPEC.md first, and they seemed more able to navigate their code once they were able to categorize code as boilerplate or content.
- **Project 3: Capstone.** We left this project almost entirely open-ended, and challenged students to create a capstone project that would demonstrate their creativity, technical depth, and AI skills. As with Project 2, we saw a wide range of impressive projects, now with more databases, phone apps, and sophisticated user interaction. We noticed that some students struggled to come up with an idea that was more exciting than what they submitted for Project 2, and in the future we might encourage students to wait on their best idea until the end of the semester. Regardless, students seemed knowledgeable and proud when explaining their work. We gave each student the ability to opt out of having their work featured on the course website's gallery page and received no objections. We sincerely hope that these projects serve as valuable assets for them in the future.

7 Quantitative survey responses

The post-assignment surveys were designed solely with student reflection in mind, and to seed class discussions, but we began to notice some interesting relationships between these responses that warrant further investigation. We posed the following Likert-scale questions for each assignment that included code production:

- (1) "How satisfied are you with [the assignment deliverables], given the time you invested?" ("1: Not at all satisfied" to "7: Absolutely in love with it")
- (2) "How much of the code did you read?" ("1: Literally none of it" to "7: Literally all of it")
- (3) "How much of the code do you feel like you understand?" ("1: Literally none of it" to "7: Literally all of it")
- (4) "Do you feel like this level of understanding was more or less sufficient to comfortably create your desired outcome?" ("1: I really needed to understand more" to "5: I understood far more than I needed to")
- (5) "How confident are you that there are no bugs?" ("1: Not confident at all" to "5: Very confident, no bugs")

We did not notice any uniformly increasing or decreasing trends, which we hypothesize is because each assignment differed substantially in its context and complexity. For example, students reported reading the least amount of code and the least understanding of that code on HW9. This assignment challenged them to create phone apps in Expo and React Native, which required substantially more code than the command-line SQLite apps created in HW6.

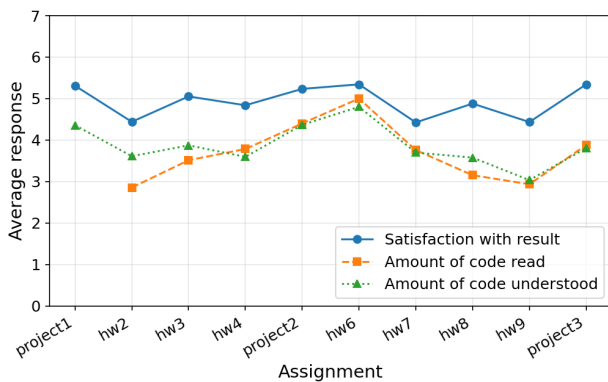


Figure 2: Average self-assessed survey responses across each assignment in chronological order. Note: "Amount of code read" was introduced in HW2.

We observed, however, that class-average satisfaction and self-reported understanding seem closely related across all assignments, as shown in Figure 2. While we only added the "How much of the code did you read" question in HW2, we also observe that the average responses to that question are very similar to the average level of self-reported understanding. These surveys were not designed as a formal research study, so we do not feel that we should make fully generalizable conclusions here, but we plan to more rigorously study the relationship between code reading, understanding, and satisfaction next semester. Given that our current

data only captures self-assessed comprehension, we are searching for ways to more directly assess comprehension while maintaining the open-ended nature of the assignments.

8 Emergent best practices

Each post-assignment reflection form asked students to respond to the following prompt: "Briefly, share one thing you found challenging, or one useful AI-related suggestion or observation with your classmates." From roughly 320 answers, we identified common themes and advice that surfaced throughout the semester. We leveraged genAI (specifically Claude Opus 4.7) to help classify and rapidly synthesize a student-facing collection of best practices sourced entirely from these comments, with manual validation to check for hallucinations or suggestions that were not well-supported by the reflections. The full document can be found on the course website (ANONYMIZED). We present a heavily condensed version here, with one representative quote provided for each point.

- (1) **Manually write a thoughtful SPEC.md before any code:** "With projects that spans multiple files or have bigger complexity, I think using agentic development is a really good starting point and I think writing the spec out actually really helps with organizing your thoughts for what exact functionalities you want versus just going with the flow in increments in conversational AI."
- (2) **Use multiple models and take advantage of their unique strengths:** "I found that chatgpt took longer and tried to create something more sophisticated while gemini was alot quicker however took alot of prompts to get a product but it made less mistakes."
- (3) **Use IDE tools with Plan and Agent mode:** "I made a plan after spec using cursor ai's plan mode and created a detailed plan first and then asked one agent to build and other to review. It got done in 3-4 agent prompts."
- (4) **Practice modularity and separation of concerns:** "Tell AI to separate the code in multiple files — I think this makes understanding the logic of the code easier."
- (5) **Build incrementally:** "AI is better suited for iterating and optimizing after an MVP is designed, rather than creating everything from scratch. The key is that you must have your own ideas first, it shouldn't be completely handed over to AI."
- (6) **Keep reading the code:** "I found [the fully agentic approach] caused me to not read my code at all. [...] I thought the finding bugs like this to be more efficient, but maybe in retrospect now, I would've preferred finding the bugs myself through physical testing rather than just trusting the bug existed and then trusting the AI solved it."
- (7) **Practice good documentation:** "I found it very challenging to read and understand the code that the AI wrote. I had to create a new agent so that it could fix all the [poorly] named variables across the project."
- (8) **References can outperform descriptions:** "If you want to combine multiple pieces for inspiration, rather than describing it in the prompt, actually provide the sources you want to combine."

- (9) **Start new chats to avoid context rot:** "Starting a new chat even with the same model fixes problems."
- (10) **Attempt to understand all errors:** "Always try to solve an error first before copying into the prompt. Even if you just spend 20 seconds looking at it, you will learn a lot more."
- (11) **Spend time researching modules, libraries, and platforms:** "Making a phone app was so different than making a web app — i was very surprised."
- (12) **Protect API keys and rate limits:** "AI is not very specific in what to do to not push an api key unless you tell it explicitly."

9 Other notable student quotes:

We also found that the following quotes represented some of the challenges faced by our students:

- "For more developed projects, if you don't clearly specify what you mean AI won't be able to do it. It can't read my mind like I thought it would."
- "I found I had to strike a balance between specificity and freedom in my spec. If given too much specificity, the AI could not do its job and the benefit of time saving was removed. If I wasn't specific enough, fixes to the code later on would no longer be simple syntax/functionality changes, but rather fundamental structural changes that the AI is not as good with."
- "I noticed that basic technical understanding is helpful and even necessary for a good spec. I failed to specify file structure. I hope I can obtain enough knowledge about architecture to be able to do so in future projects though."
- "I found it quite challenging to write a spec when I already wasn't entirely sure how things were working. Like I knew I needed to use a camera function, but had no idea how to write that into my spec other than literally and hope that is made enough sense."
- "Having a clear idea / vision was very important. I went with the flow too hard and my project was just very random."

This next group of quotes illustrates some key discoveries that students were eager to share with their peers:

- "When prompting, emphasize that you want to understand why we're changing certain code and what certain code does for us implementing our desired feature."
- "I tried this process where I asked Gemini for prompts and planning while giving prompts to Claude. The process itself was less buggy, I think I encountered less bugs than I would have if I just prompted myself, but it took much longer than my peers to get to the same point."
- "I compared the output of both chatgpt and gemini and I found that chatgpt took longer and tried to create something more sophisticated while gemini was alot quicker however took alot of prompts to get a product but it made less mistakes."
- "One really helpful thing I learned from this homework was how effective a framework I learned from David Hanson (DHH) is. Essentially, I had AI help generate my plan for a manually coded feature, then gave it my best shot without the AI's help, then gave it all the code I wrote and had it comment on it. This way, I really understood the code from

the start, and any edits/improvements the AI suggested, I actually could directly understand why it was making those changes, because I knew how it impacted the lines around it."

- "With projects that spans multiple files or have bigger complexity, I think using agentic development is a really good starting point and I think writing the spec out actually really helps with organizing your thoughts for what exact functionalities you want versus just going with the flow in increments in conversational AI."

10 Conclusions and Future Work

We present this course here so that it can serve as an experiential reference to other educators and institutions that wish to provide students with durable AI skills. In its first semester, we made continuous adjustments in response to student feedback and current events. We believe that the key to keeping hands-on genAI courses current is to engage in collaborative exploration with our students, thereby increasing the flow of new ideas and the speed with which we can test emerging AI developments. Furthermore, a socially supportive learning environment creates valuable opportunities for students to improve their communication skills with their peers and instructors. The overall quality of student projects, their fluency when presenting their work, and their thoughtful reflections on best practices suggest to us that our students have substantially improved their genAI literacy and proficiency while maintaining a strong desire to build their technical fundamentals.

There is still much work to be done. The first iteration of our course was highly experimental, with many aspects developed in real-time. Moving forward, the course must remain adaptive, but we now have a demonstrated course structure that we can rely on, even if specific topics change. We also now realize that the course carries the potential for important pedagogical research, and the reflections we gather in future semesters will be designed to fit the dual purposes of student growth and rigorous investigation.

We hope that other educators who are experimenting in this space will contribute their own findings. Similar to how we benefit from our students' exploration, educators and institutions can benefit from sharing our experiences through mutually-supportive communities of practice.

Acknowledgments

ANONYMIZED

Ethics and Privacy Statement

We do not foresee new and significant risks from releasing this report, but we acknowledge that modern generative AI is built on uncredited labor and personal data, its use carries a risk of dependence, and it places significant burdens on the environment and disadvantaged populations. As educators we bear the responsibility of ensuring that our students are empowered by our teaching, and this requires us to effectively communicate these dangers and hidden costs.

Generative AI was used in the process of creating student-facing resources for the course and for figure formatting in this report. The text of this report is entirely human-authored.

We have taken care to omit all potentially identifying student information from this report. The premise of this work has been submitted to the (redacted) Institutional Review Board and categorized as Exempt.

References

- [1] Isaac Asimov. 1950. The Evitable Conflict. In *I, Robot*. Gnome Press, New York, 193–253. Originally published in *Astounding Science Fiction*, June 1950.
- [2] Isaac Asimov. 1950. Runaround. In *I, Robot*. Gnome Press, New York, 40–58.
- [3] Cat Hicks. 2025. Cognitive Helmets for the AI Bicycle: Part 1. <https://www.fightforthehuman.com/cognitive-helmets-for-the-ai-bicycle-part-1/>. Accessed: 2026-07-03.
- [4] Xin Liu, Longxin Zhang, and Xiaochong Wei. 2025. Generative Artificial Intelligence Literacy: Scale Development and Its Effect on Job Performance. *Behavioral Sciences* 15 (2025), 811. doi:10.3390/bs15060811
- [5] Stephanie M. Tully, Chiara Longoni, and Gil Appel. 2025. Lower Artificial Intelligence Literacy Predicts Greater AI Receptivity. *Journal of Marketing* 89 (2025), 1–20. doi:10.1177/00222429251314491
- [6] Karel Čapek. 1920. *R. U. R. (Rossum's Universal Robots)*. Aventinum, Prague. First performed in 1921. Introduces the word "robot".

Received 3 July 2026

DRAFT