Issues in Informing Science + Information Technology

An Official Publication of the Informing Science Institute InformingScience.org

IISIT.org

Volume 22, 2025

AI ASSISTANCE VARIANTS IN SOFTWARE DEVELOPMENT CYCLES

Micheal Callahan	Grand Canyon University, Phoenix, AZ, USA	micheal.callahan@icloud.com
Joseph Claus	Grand Canyon University, Phoenix, AZ, USA	joeyclauss27@gmail.com
Emmy Voita	Grand Canyon University, Phoenix, AZ, USA	JVoita@my.gcu.edu
Christine Bakke*	Grand Canyon University, Phoenix, AZ, USA	christine.bakke@gcu.edu

* Corresponding author

ABSTRACT

Aim/Purpose	With the technology of artificial intelligence (AI) improving every day it is important to find ways to harness AI in the software development life cycle (SDLC). This research demonstrates how AI tools were incorporated into an upper division Computer Science course to assist with development of various memory games.
Background	Since ChatGPT's release in 2022, other companies have released rival chatbots each competing for a piece of the new market. With the plethora of AI options now available, it is important for a developer to learn to use AI as an assistant within the development of a custom project.
Methodology	The research presented is a multi-case, cross-analysis of four student researchers in a required, senior level Computer Science course. All students were tasked with collecting mixed-methods data on two AI assistants, throughout design and development a unique memory app; then these four students pooled data and conducted a cross-comparative analysis. To prepare for cross analysis, standardized Likert rankings and thematic categories were developed and consistently used during data collections. AI assistants evaluated: Claude, Copilot, ChatGPT Free, and ChatGPT Paid. Throughout the development process, each student provided both of their AI assistants with the same initial queries, the results of which were given a Likert ranking and notes were kept regarding AI accuracy. Individual datasets were examined, then pooled and the combined da-

Editor: Eli Cohen | Received: January 9, 2025 | Revised: March 16, April 16, 2025 | Accepted: May 19, 2025
Cite as: Callahan, M, Claus, J., & Bakke, C. (2025). AI assistance variants in software development cycles. *Issues in Informing Science and Information Technology, 22,* Article 16. https://doi.org/10.28945/5539

	taset was used to finalize hypothesis findings. The four student-researchers presented their multi-case, mixed-methods analysis as a snapshot in time regarding the value of AI as assistants in the development of their projects.
Contribution	This paper builds on prior research focusing both on student experience and instructional methods in capstone-like courses. This study examines using AIs as assistants as a current trend in Computer Science education.
Findings	During multi-case analysis, two hypotheses were analyzed against the data of the four student-researchers. The cross examination of data found no statistical significance between the helpfulness of paid vs. free AI as course project assistants; while non-IDE AI assistants performed significantly better than IDE assistants across 7 out of 8 usage type categories.
Recommendations for Practitioners	Technology instructors can use this research to incorporate AI assistants into advanced courses that focus on building custom software, with cautions that foundational coding skills and knowledge should be in place prior to attempting complex projects. Companies that are researching how AI can be integrated into the software development process can use this research to see preferred strengths of various AI's, with cautions for use with proprietary data.
Recommendations for Researchers	Researchers can observe how different AI's can assist with application development. Further research is encouraged as AI capabilities will continue to evolve.
Impact on Society	The researchers' findings show AI in light of its current abilities and limitations in the software development life cycle. While AI assistants excelled in simple to medium complexity debugging tasks, there were many complex tasks where a human coder was preferred over the AI assistants; however, this is expected to change over time.
Future Research	As future technology strengthens AI some aspects of the study may become historical; however, the core of the research, that of using AI as assistants in development of software projects is expected to remain pertinent to education for some time.
Keywords	AI, artificial intelligence, ChatGPT, Claude, Github CoPilot, software development life cycle, SDLC

INTRODUCTION

In recent years universities have increasingly incorporated agile methods, such as Scrum, into Computer Science courses to better prepare students for industry work. Incorporating Scrum-based methodologies has been shown to help students adapt to changing requirements and improve collaboration skills (Hsu et al., 2019). In addition, studies on distributed software development highlight that working in teams, including virtual teams, strengthens development of soft skills in order to better prepare students for modern work environments (Christensen & Paasivaara, 2022). Other approaches, such as innovative educational methods and full-stack project experiences, aim to align academic activities more closely with real-world industry needs (Laval et al., 2021; Metrôlho et al., 2022) and reduce gaps between academic learning and employer expectations (Sahin & Celikkan, 2020).

It serves a dual purpose to evaluate the effectiveness of AI in the role of a software development assistant and to offer practical experience with agile software development methodologies. Current Computer Science research investigates human-AI pair programming as a strategy to enhance productivity and inspire more creative problem-solving in classroom environments (Zhang et al., 2022). There is interest in examining whether AI tools can produce concise, functional code suitable

for both educational and professional contexts (Millam & Bakke, 2024). At the same time, large language models (LLMs) and AI coding assistants are gaining attention for their potential to support programming tasks; however, their effectiveness varies and the level of integration into existing workflows remains unclear (Liang et al., 2024).

In this study, four student-researchers conducted single-semester AI research in an undergraduate "Current Trends in Computer Science" course. Working in two-week Scrum sprints, they developed a simple, custom Unity application with the help of AI. This research tracks four projects, wherein each student selected two different AI tools to assist them with their project development. Throughout the project, student-researchers systematically collected data on each AI interaction. The data sets were categorized and analyzed based on value, with comparisons made between paid and free versions. Additionally, AI tools that integrated with Integrated Development Environments (IDEs) were contrasted with those that did not. The students provided a comprehensive analysis of their fifteen-week experiences in AI-human collaborative project development, focusing on the integration of AI as project assistants in creating a Unity app designed to enhance memory skills.

While the research presented is the combined work of four student-researchers; only two student-researchers followed through with the instructor to finalize the work for publication. The multi-case study thus presents the work of four student-researchers as they independently developed memory loss apps in conjunction with testing the helpfulness of two AI assistants in development of a software project.

Each of the student-researchers developed a hypothesis; however, during cross case analysis it was noted that there was significant overlap. To avoid redundancy and highlight the commonalities, the following two hypotheses are presented in this multi-case analysis:

> **H1 Hypothesis:** Paid AI assistants will significantly outperform free AI assistants in terms of performance.
>
> **H2 Hypothesis:** IDE-based AI assistants will significantly outperform non-IDE-based assistants.

Following this introduction, the paper defines key terms, states the research question, and outlines the hypotheses and objectives. Next, the literature on agile education, industry alignment, and the integration of AI coding assistants is reviewed. The methodology, including participant selection, data collection, and analysis procedures, is then described. After presenting and analyzing the data, the paper discusses the findings, addresses ethical considerations, and concludes with insights, limitations, and suggestions for future research. Finally, the Appendix contains a selection of project snapshots.

LITERATURE REVIEW

The researchers participated in an undergraduate course where they utilized AI throughout the development of custom software. This experience was designed based on current trends in the literature, which highlight the growing use of AI in the industry (Durrani et al., 2024). As AI continues to expand, Panetta (2023) predicts that by 2027, 70% of professional software developers will use AI-powered coding tools. These trends, coupled with ongoing industry research, indicate a clear need to integrate industry-emulating AI into Computer Science courses. Furthermore, Bankins et al. (2024) explore AI's evolving role in the workplace, focusing on employee collaboration, perceptions, and algorithmic management, highlighting the increasing significance of AI in professional settings.

Some researchers express concerns that AI could replace software developers, while others, like Campbell (2020), foresee a growing role for AI throughout the Software Development Life Cycle (SDLC). Ebert and Louridas (2023) investigate generative AI's potential to assist in code generation and improve productivity but do not examine how developers actually use or perceive these tools in

practice. This gap in understanding AI's real-world applicability emphasizes the need for students to become familiar with current trends and challenges in the tech sector. Hands-on collaboration with AI, as seen in the course, helps students engage directly with these emerging tools and the issues surrounding their use.

Shi et al. (2023) examine how generative AI, specifically GitHub Copilot, can improve code security by helping developers avoid common coding errors. While they are optimistic about AI's potential to enhance security, this perspective contrasts with concerns raised by Perry et al. (2023) and Vaidya and Asif (2023), who warn that AI-generated code could introduce security vulnerabilities. These concerns are rooted in the possibility that AI-generated code may contain flaws originating from the training datasets. This dual-edged nature of AI in coding underscores the importance of developers understanding AI-generated code fully to ensure its quality and security. It highlights the challenges of integrating AI into the development process while mitigating potential security risks.

The challenge addressed in this research was to create an AI-integrated software development experience complex enough to require student guidance but bounded enough to be completed within a 15-week timeframe. The students not only had to complete individual projects of their own design but also collect and analyze data comparing two AI models used throughout their development process. After reviewing the literature, it became clear that this approach was novel, offering a meaningful software development experience with AI assistants. It also provided an opportunity for students to learn essential skills in data collection, analysis, and possibly even publication, thus fulfilling the need for both technical and research-based competencies in the curriculum.

METHODOLOGY

PARTICIPANTS

This study was conducted in a senior-level, undergraduate course entitled, "Current Trends in Computer Science". The purpose of the course is to provide students with a comprehensive understanding of current trends and practices in Computer Science through practical experiences which vary each semester. This version of the course incorporated Scrum practices to guide a term project assisted by AI; data was collected throughout the course on the effectiveness of AI as assistants in software development.

The methodology of this study closely aligns with that used by Millam and Bakke (2024), as both studies were conducted within the same course framework. However, the previous project was designed for a client, whereas this project had no client; rather each student determined the design of a custom memory game specifically targeting those suffering from memory loss. Another difference can be seen as students developed an individual game until the last weeks of the course, when they combined the individual games into a single app.

STRENGTHENING RESEARCH VALIDITY

Validity of conclusions is a concern in small sample size studies, but several factors add to the validity of this study. First, carefully chosen participants are crucial. According to O'Reilly and Parker (2013) In qualitative research, the adequacy of the sample size is determined by the richness of the data rather than the number of occurrences. As a result, participants should be selected based on how well they represent the research topic. (Morse et al., 2002). This supports the idea that small, carefully chosen samples are valid if they provide rich, in-depth data.

Validity was strengthened across the study through homogeneity of participants, use of multiple AI's, consistent data collection, standard analysis methods, and mixed methods analysis. The study took place in a required course, resulting in homogeneity of participants; this is seen to reduce variance unrelated to the study's focus, which can be seen in demonstrated consistency in patterns across participants.

The focus of the research further strengthens data validity. In theory, individuals with a clear and focused idea typically have a corresponding research agenda, which helps shape the direction of data collection. This process establishes specific parameters and areas of interest, within which data saturation can be reached. (O'Reilly & Parker, 2013). This supports the study's design, which is narrowly focused on AI performance in specific contexts, namely simple memory game development. This narrow scope increases the potential of achieving saturation within the study's focus.

The utilization of mixed methods and cross-case analysis also increases validation of findings by correlating data from four different student-researcher perspectives. As Cresswell and Cresswell (2018) outline, mixed methods research integrates both qualitative (open-ended) and quantitative (closed-ended) data to address research questions or hypotheses. This approach utilizes rigorous methods (i.e., data collection, analysis, and interpretation) of both quantitative and qualitative frameworks, merging the methods to provide a more comprehensive understanding of the research problem.

Throughout the development process, whenever a student sought assistance from their AI assistants, they recorded both the question and the resulting answer. Additionally, they rated the response using a Likert scale and provided explanatory notes to justify their ranking. To address concerns about potential bias in individual datasets, the four student-researchers combined their datasets for cross-comparative Likert and thematic analysis against each hypothesis. The course multi-case research design is shown in Figure 1.

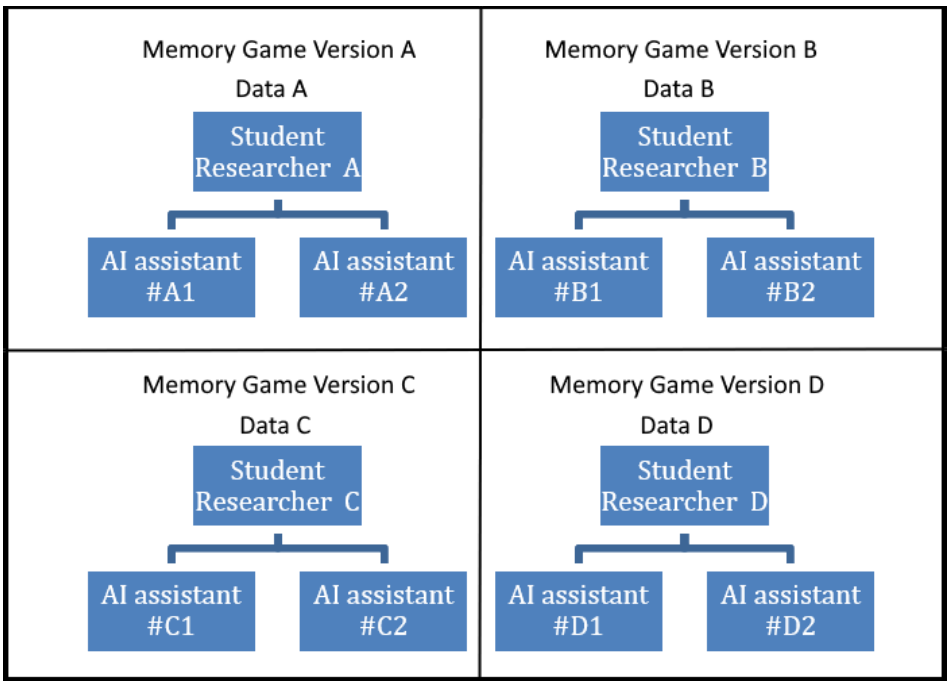


Figure 1: Multi-case project design, individual & group.

Even if saturation cannot be fully achieved, this study focuses more on exploratory insights, rather than drawing definitive conclusions, thus maintaining validity. As O'Reilly and Parker (2013) state, transparency about failing to reach saturation does not automatically invalidate the findings. Instead, it indicates that the phenomenon may not have been fully explored, rather than suggesting the results are unreliable or incorrect (Morse, 1995). This approach supports the exploratory nature of the study, suggesting areas for larger-scale investigations.

Lastly, to support some level of generalizability, we will evaluate the conclusions using regular data collections using both qualitative and quantitative methods; qualitative data collection was employed

with thematic analysis along with quantitative Likert data examination. As highlighted in prior research, alignment with existing work and the clarity of emerging themes can justify the validity and generalizability of results, even in studies with small sample sizes. As Vasileiou et al. (2018) found in their analysis of qualitative health research, the validity of the results was supported by the clarity and distinctness of the emerging themes, as well as their alignment with existing relevant research. Although the sample size was small, the narratives revealed clear themes that were adequate for the purposes of this exploratory study (SHI98)."

Data Collection Methods

At the beginning of the course each student selected two AI assistants based on individually determined criteria; this resulted in a variety of AI assistants being utilized, tested, and analyzed. The inclusion of multiple AI assistants was seen to broaden the scope of the study, strengthen the generalizability of its conclusions, and provide a greater range of experiences for participants.

Each student began the course by designing a custom user interface (UI) for a memory app which could benefit everyone, but was tailored for use by elderly, dementia patients. Unity was used for development as all students had at least one prior Unity course. The course was small, consisting of only eight senior Computer Science students; although students conducted their studies independently, analysis underwent both individual and small group analysis; the eight student-researchers in the course were divided into two groups, randomly selected by the professor, for the small group, analysis. The course consisted of eight, two-week sprints, in which participants used Trello or Jira to keep track of major and minor tasks. This paper describes the results from one of these groups of four. Each sprint contained one stand-up at the end, in which students would each present their project by addressing three-questions of Scrum along with screenshots and an explanation for each question:

1. Which tasks did you complete this sprint?
2. Which tasks will you be working on during the next sprint?
3. What is the greatest challenge you are currently facing?

Before data collection, a common five-level ranking was determined as a class and used consistently among students. Each interaction with AI was ranked using the Likert scale, to track general AI performance and ensure consistency in performance assessment across participants. This Likert scale closely reflects AI tool capabilities at the time, recognizing that the perception of what defines a "high quality" vs. "low quality" AI experience varied somewhat between participants. The Likert ranking used across all data collections is shown in Table 1.

Table 1. Likert rankings, common across all research projects.
Student determined, consistent Likert and note rankings; used across all data collections

Meaning	Ranking
1 star: Tried up to three times, but the AI response is completely unusable	★
2 star: First AI response was on track enough to keep tweaking. 5 tweaks or more and it is now somewhat usable.	★★
3 star: AI response was on the right track; 3 – 4 tweaks to get usable helpful result.	★★★

Meaning	Ranking
4 star: AI response was close or what I needed, but nothing more. 1-2 minimal tweaks if needed.	
5 star: AI did better than I had hoped for with the first response. Excellent.	

Students developed the initial app individually through week twelve, collaborating in the groups of four for data analysis, presentations, and troubleshooting. Each student used two AI assistants, posing the same initial query to both. Midway through and at the end of the course, each small group compiled and compared their AI data, reporting on the value of their assistants both individually and comparatively. Throughout the term project, students were required to record and evaluate a minimum of four AI queries each sprint.

To assess AI performance, students asked the same query to both assistants, treating the query as the independent variable. While both AIs received the same initial question, follow-up queries based on each AI's responses led to slight variations in the total number of queries.

In addition to rating AI performance on a Likert scale, students documented the date, initial prompt, and notable response characteristics for each iteration. While Likert data was directly analyzed, notes were organized by combining all class datapoints and looking for themes. Once the students determined common themes, the notes were grouped by theme and analyzed, separate from their Likert ranking. This data was used for detailed comparisons at the end of data collection. An example of a data entry can be seen in Table 2.

Table 2: Example of a data entry.
Ranking represents the Likert scale rating used to assess AI performance, while Results/Notes are the notable response characteristics as previously mentioned.

Date	Prompt	Result / Notes	Ranking
10/29/2024	if i want to hold a reference to a variable i need to reference in a render feature in a singleton seperated from the render feature how can i manage that without getting an object reference not set to an instance of an object error? Do i need to pass a reference to the Singleton instance variable?	* Two responses * Minor modifications needed to be made with the information it gave, just because I didn't provide it with enough context. (Same as Copilot) Gave it old code rather than the most current code, and asked it to give me a solution using the old code. * Produced very similar code to Copilot * produced working code on the first try. Took Two responses because I had to modify two scripts and I asked about each script separately. * gave what I was looking for. However, Im not 100% sure its the best solution. I have a feeling its not, but either way, im still giving it a 4 because it seems to work fine and fixed what i asked. * gave additional recommendations for implementing a soliton, over Copilot, which was helpful.	4

Importantly, the quality and detail of the notable response characteristics varied significantly between individual students. This inconsistency occurred because a definition for what needed to be included wasn't explicitly established before data collection began, leaving the recorded information to depend on individual understanding of what might be important based on their hypotheses.

Data Analysis Procedures

At the conclusion of the term project development in week twelve, AI interaction responses were analyzed based on several key qualitative factors, which were then quantitatively assessed. These factors included code quality, response effectiveness, explanation clarity, complexity, time efficiency, and usage type. Each student initially reviewed the AI data they had collected individually. To minimize bias, a multi-case cross-examination of the entire small group dataset was conducted. Both Likert and

thematic analyses were performed, and each student's hypotheses were evaluated against their small group's data results.

In the study Liang et al. (2024), the authors extend an existing study by conducting a large-scale investigation, specifically focused on the usability challenges of various AI programming assistants, including GitHub Copilot. Data was collected through a Qualtrics survey sent out to a target group that included the successful use cases of AI for survey participants. The authors found 10 types of situations, which they describe and report the frequencies for. Similarly, this study examines the successful use cases of AI assistants, using some of the situations described by Liang et al. (2024) for qualitative analysis. Not all use cases in this study align perfectly with the situations identified in Liang et al. (2024), so some additional use cases had to be defined to capture the full range of that observed. It is worth noting that, while auto-complete was among the most frequent use cases identified in Liang et al. (2024), auto-complete use cases were not documented in this study. This occurred despite testing including an IDE that supported auto-complete, namely Copilot. This omission could be justified as incorporating auto-complete into our data model would have been too complex, but it does impact the conclusion for H_2.

This study also examines AI usage patterns. Data collection is captured over a three-month period, and project development is composed of eight, two-week sprints. To analyze temporal trends, AI usage data was grouped into two halves: the first half (Sprint 1-4) and later half (Sprint 5-8) of project development. This study examines AI usage patterns, but further insights could be gained using a larger time-stamped data set, allowing examination of characteristics changing over time.

Use case categories

- **Proof-of-concepts.** Providing code or recommendations that explore whether a potential solution would work or determine the best way to implement a system.
- **Learning.** Using AI assistance to understand new programming languages, libraries, or concepts.
- **Efficiency.** Using AI tools to "speed up" workflow, such as by avoiding looking up documentation.
- **Code with simple Logic:** Generating straightforward code, such as utility functions or simple algorithms. These tasks involve minimal complexity.
- **Implementation Guidance.** Providing step-by-step instructions, code, or advice on implementing a solution.
- **Debugging.** Identifying, diagnosing, and fixing code errors with the help of AI tools.
- **Quality Assurance.** Generating test cases, identifying edge cases, or ensuring that code is optimized.
- **Design.** Using AI tools to assist with design-oriented tasks (i.e., UI placement questions, game design, etc.)

Each week, a minimum of two data points for each AI were collected, ranked, and reflected upon. When a student-researcher wished to pose a question to AI, they would ask the same question of both AI assistants, thus reducing preconceived bias on the abilities of each AI model. AI responses were ranked using a five-point Likert scale along with notes explaining the reason for the Likert ranking. In this way, student-researchers were able to evaluate and compare AI responses across projects in a meaningful and consistent manner. Examination of explanatory notes introduced categorical metrics which were used to further assess AI performance, including: task complexity, and time saved by assistance.

Time saved by AI Assistance

- **No Time Saved** - The AI assistance did not save any time and likely increased the time spent.
- **Very Little Time Saved** - The AI assistance did not save any time, or only saved a minimal amount of time.
- **Neutral** - The AI assistance neither saved nor added significant time; it had little impact on time spent.
- **Some Time Saved** - The AI assistance saved a considerable amount of time during the task.
- **A Lot of Time Saved** - The AI assistance saved a significant amount of time and greatly improved efficiency.

Complexity of tasks addressed

- **Very Low Complexity.** The task involved simple operations or beginner-level concepts (e.g., creating a basic function or fixing a syntax error).
- **Low Complexity.** The task required straightforward implementation but with some intermediate-level considerations (e.g., basic algorithm development, or adding a simple UI element).
- **Moderate Complexity.** The task involved integrating multiple components or intermediate-level problem-solving (e.g., designing and implementing a moderately complex feature, or debugging multi-step processes).
- **High Complexity.** The task required advanced problem-solving, concepts, or implementation (e.g., developing custom algorithms, optimizing performance).
- **Very High Complexity.** The task involved highly specialized development (e.g., solving highly intricate debugging issues).

Factors Contributing to High-Quality Responses

Responses that received a 5-star rating from the general AI performance Likert scale were further analyzed to identify contributing factors for high-performing responses, which were condensed into the following categories:

- **Better Explanation of Concepts:** Clearer and more detailed explanations that aid understanding.
- **Preemptively Addressed Follow-Up Questions:** Responses that answered potential follow-up questions that were not specifically addressed in the prompt.
- **Additional Helpful Code:** Offering code beyond what was explicitly requested.
- **Additional Considerations:** Including solution pros and cons, alternative solutions, or other helpful suggestions that added to the response.
- **Multiple Working Scripts:** Provided multiple scripts for multiple solution or approaches to the same problem, or for implementing a solution across multiple scripts.

Differences in explanation quality emerged as a significant factor to AI performance. To investigate further, a similar process was applied to explanation quality, which identified the following:

Factors contributing to high-quality explanation

- **Clarity and Explanation Quality:** Clearer and more detailed explanations of concepts and solutions. Better explanations that help me learn and understand key considerations. Detailed and actionable explanations Extra suggestions for features

- **Multiple Solution Routes:** Suggestions for multiple solutions or approaches to solving the problem.
- **Implementation Guidance:** Detailed implementation instructions or explanations of how to apply solutions.
- **Anticipating Needs and Questions:** Preemptively answering follow-up questions or addressing potential challenges. Including additional considerations or recommendations that I might not have thought of yet.
- **Justification for features:** Justifications for solutions and why they work. Quality assurance.

Code quality assessment

Given that context-aware coding assistance is a key advantage of AI integrated with an IDE, this study also examines code quality differences.

- **Code did not work.** AI-generated code was entirely unusable.
- **Code required Major modifications.** Significant alterations were needed.
- **Code required no or minor modifications.** AI-generated code was functional with minor adjustments.

Table 3 provides an overview of sprint goals for game development that integrates a student researcher as the lead developer with two AI assistants. At the end of each sprint, students completed a stand-up as an informal, required presentation addressing the three questions of Scrum. The structure of the course shown below is for a 15-week, in-person, senior level Computer Science course.

Table 3 Game Project Development Goals

	Software Project	AI Research
SPRINT 1	Project set-up (Unity), exploration of memory-loss needs	Selection of AI assistants, determination of Likert scale, begin collecting AI data
SPRINT 2	UI/UX design & game flow, project backlog, sprite	Hypothesis, problem statement, purpose, research overview
SPRINT 3	Contact, movement, score, user stories	Methodologies, data analysis introduction
SPRINT 4	Music, settings, troubleshooting, Sprint Retrospectives	Analysis of data, determination of themes
SPRINT 5	"User" testing – conducted by classmates	Midterm research report including initial AI data analysis, project demonstration
SPRINT 6	Combine four games and add a menu to select.	Guest speaker, example of published paper and professional presentation
SPRINT 7	Group game testing, individual game troubleshooting "polish"	Review and journal formatting
SPRINT 8	Presentation of memory-assistive games	Presentation, poster, article submission

RESULTS AND DISCUSSION

The analysis revealed distinct patterns in the performance and usage of AI tools, with findings summarized across qualitative and quantitative measures. A comparative graph, listing the quantity of queries posed to each AI is listed in Figure 2; due to the possibility of individual AI's being posed with clarification questions, the total number of queries did not result identical interaction between each of the AI assistants.

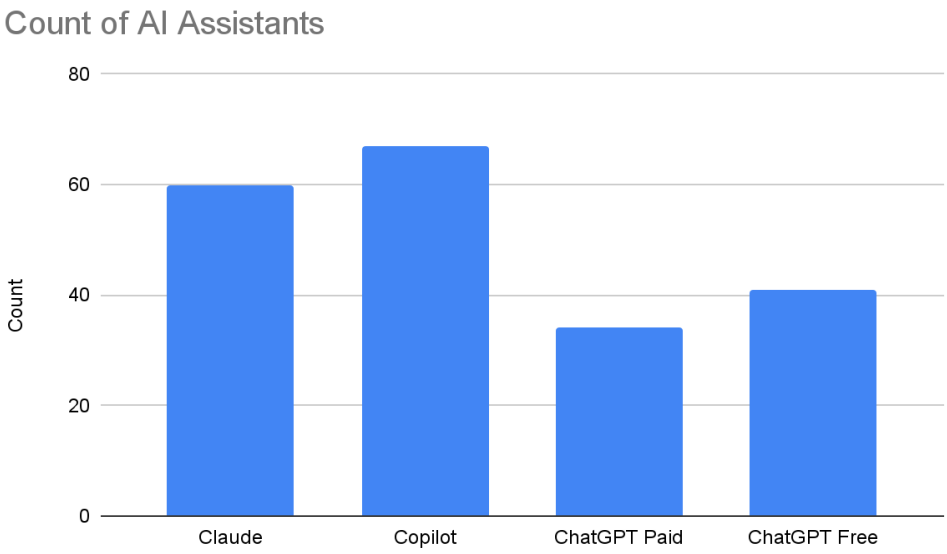


Figure 2: Count of questions asked to each AI assistant.

Code Quality and Reliability

ChatGPT consistently outperformed other AI tools, such as Copilot and Claude, in delivering reliable code. While Copilot struggled with certain prompts, ChatGPT exhibited greater consistency and fewer instances of failure. Claude performed well in specific scenarios, such as outlining processes for UI/UX tasks, but required additional prompting for complex implementations. Calculations for central tendency based on Likert data can be seen in Figure 3.

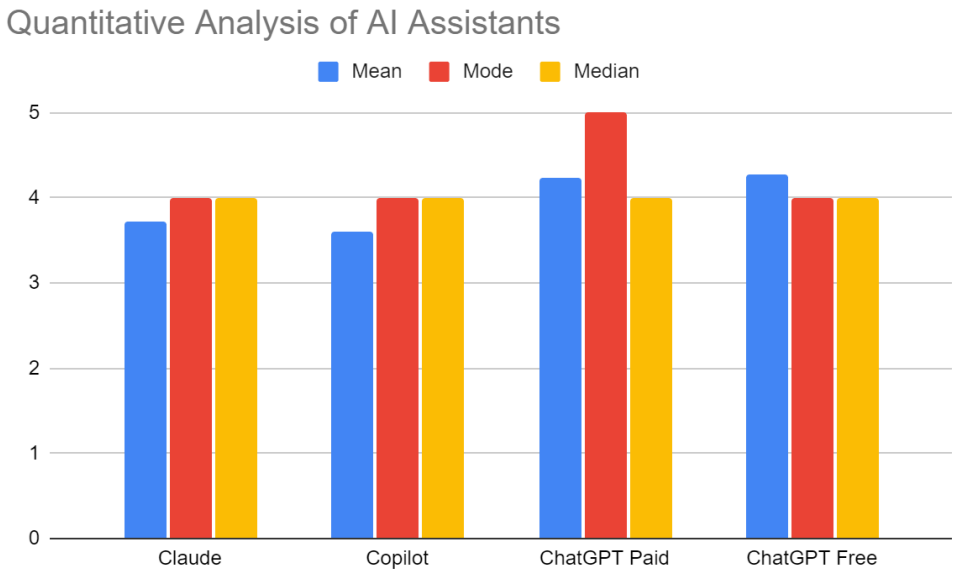


Figure 3: Mean (blue), Mode (red), median (yellow) for each AI assistant.

Usage Patterns and Trends

AI usage varied significantly across the project timeline. During the first half, AI tools were primarily utilized for tasks involving simple logic, learning, and implementation guidance. In the latter half, usage shifted toward learning-focused tasks and code with minimal complexity. ChatGPT demonstrated particular strength in areas like debugging, learning assistance, and conceptual explanations, while Copilot and Claude were more effective for design and proof-of-concept tasks.

Task Complexity and Performance

Performance across all AI tools exhibited a downward trend as task complexity increased. ChatGPT outperformed other tools in high-complexity tasks, showing a 44.3% improvement over competitors. However, 92.6% of tasks fell into the simple to moderate complexity range, with no very high complexity tasks due to project limitations.

Desired Response Characteristics

ChatGPT received significantly higher 5-star ratings compared to Copilot (51.2% vs. 19.5%), driven by its ability to provide additional insights, detailed code explanations, and multiple working scripts. Claude and Copilot were noted for their ability to clearly explain concepts and offer practical solution routes but fell short of matching ChatGPT's overall quality.

Hypotheses Testing

1. **H1 (Paid AI vs. Free AI):** The hypothesis that paid AI tools would significantly outperform free AI versions was rejected. Although paid AI demonstrated slightly better reliability, free AI tools achieved comparable or better results in complexity-related tasks. The mean difference (.24) did not meet the 5% statistical threshold.

2. **H2 (IDE-Integrated AI vs. Non-IDE AI):** The hypothesis that IDE-integrated tools would outperform non-IDE assistants was also rejected. Non-IDE AI tools performed better across multiple categories, with a statistically significant mean difference of .43 in favor of non-IDE assistants.

In order to more thoroughly analyze data collections, each student conducted an individual thematic analysis of their dataset, specifically notes were grouped into common themes, tabulated and compared. The results were then evaluated against the corresponding hypothesis. Cross-analysis graphs showing thematic results across the four data sets are shown in Figures 4, 5, 6, and 7.

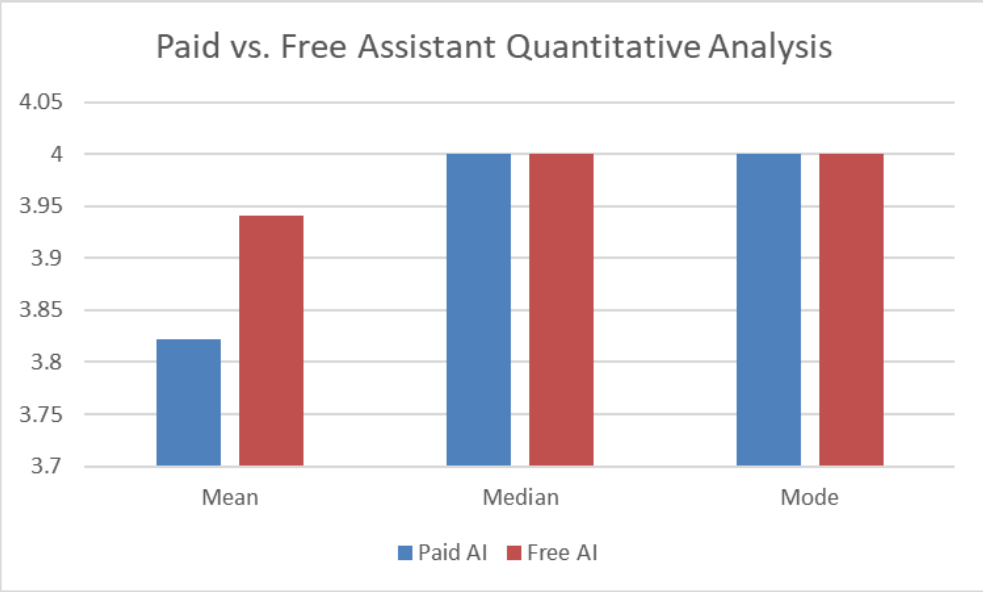


Figure 4: H1 Hypothesis of Paid AI's (CoPilot and ChatGPT Paid) vs Free AI's (Claude and ChatGPT Free) Quantitative Analysis.

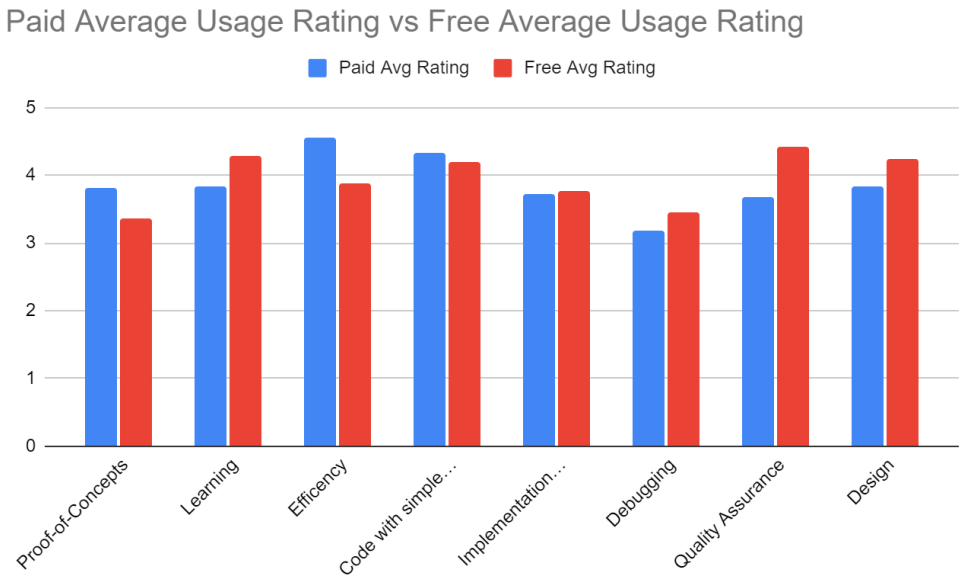


Figure 5: H1 Hypothesis of Paid AI's (CoPilot and ChatGPT Paid) vs Free AI's (Claude and ChatGPT Free) Qualitative Analysis.

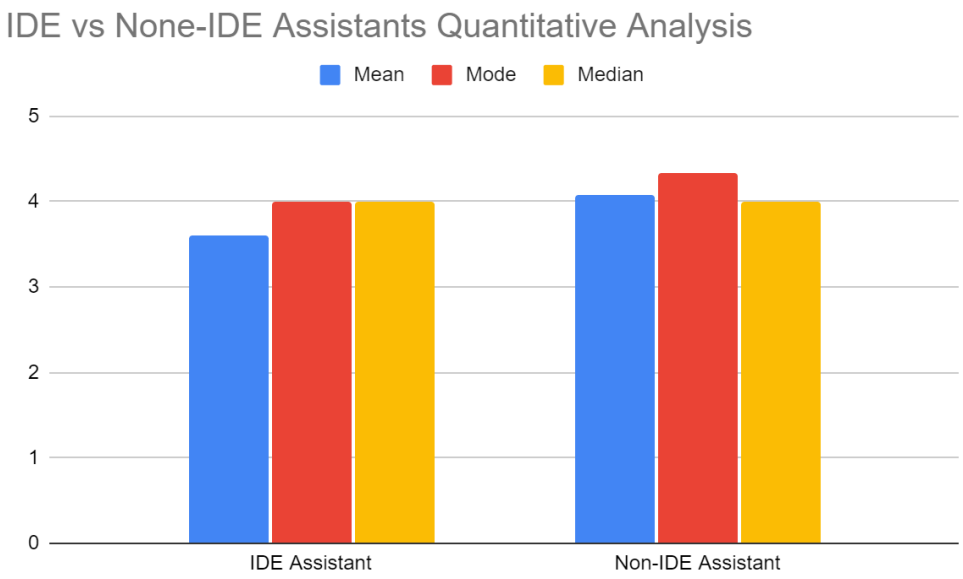


Figure 6: H2 Hypothesis of IDE AI (CoPilot) vs Non-IDE AI's (Claude, ChatGPT Paid, and ChatGPT Free) Quantitative Analysis.

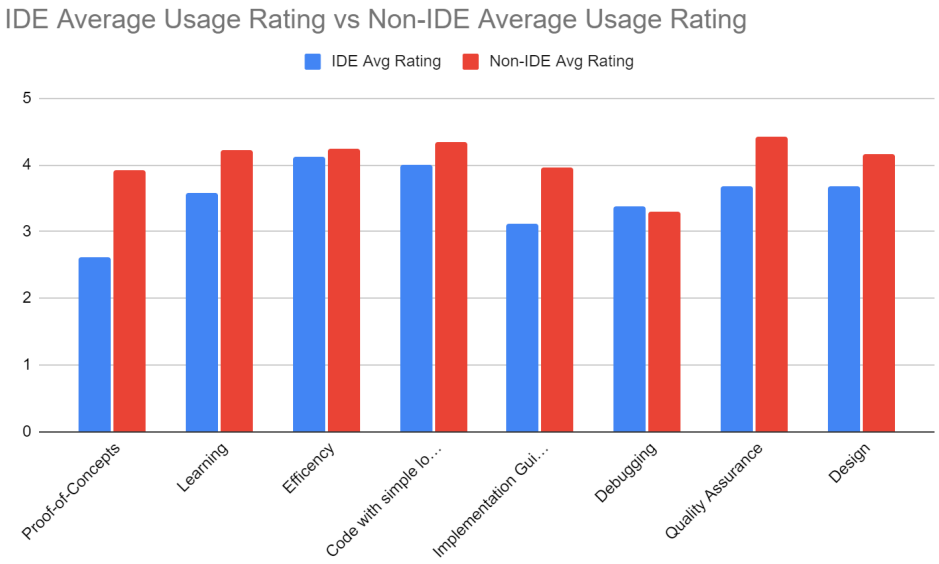


Figure 7: *H2 Hypothesis of IDE AI (CoPilot) vs Non-IDE AI's (Claude, ChatGPT Paid, and ChatGPT Free) Qualitative Analysis.*

Overall Trends

Individual and group analyses were conducted to evaluate the effectiveness of AI throughout development; across all projects, participants kept track of each query they asked their AI assistant. To reduce data bias, when students wished to query AI, they were tasked with querying both AI, rather than selecting one. Students then evaluated both AI responses, keeping brief notes on the helpfulness of each response, and providing a ranking. The results indicate that all AI tools achieved aver-

age performance ratings above 3 on the Likert scale, indicating general usability with minor modifications. A key observation was that performance declined as task complexity increased, but AI tools excelled in low to moderate complexity tasks.

CONCLUSION

In conclusion, the analysis of AI assistants during the development of memory loss apps revealed key insights into their effectiveness and value. By comparing the performance of paid versus free AI assistants and IDE-based versus non-IDE-based assistants, the study aimed to assess their impact on project development. The mixed-methods approach, combining both qualitative and quantitative data, allowed for a comprehensive evaluation of the AI tools' contributions across different student projects. Ultimately, the findings offer valuable implications for integrating AI into development processes and provide a basis for future research into optimizing AI-assisted project development.

The class expectations were that individuals complete their project development by the end of week twelve (sprint six), in order to focus the final two sprints on combining the group projects into a single app and wrapping up research, presentation, and analysis requirements. The class was split in meeting this expectation, with all members of one group completing the combined project by the seventh sprint and each member of the second group completing their individual projects and partially completing the combined group project by the final sprint.

The student researchers were surprised to find that both hypotheses were rejected. The H1 hypothesis of paid AI assistants significantly outperforming the free AI assistants was rejected as there was not a significant difference between the two types. The H2 hypothesis of IDE AI assistants significantly outperforming the Non-IDE assistants was rejected as it was found that Non-IDE assistants performed significantly better than the IDE AI assistants. Beyond the hypotheses, analysis also revealed that AI assistants averages were all over three, showing that AI on average was able to answer questions well, with only minor wording modifications from any given student-researcher. Of note was the examination of patterns which revealed an unexpected trend, common across all AI showing a decrease in AI assistive value when queried for help with complex coding tasks, i.e., as complexity increased AI performance significantly worsened, whereas with simple to moderate tasks each of the AI assistants consistently performed well.

It could be valuable for future researchers to note the class was structured into two-week sprints, each with well-defined project objectives (Bakke & Sakai, 2022). With the exception of the midterm and final project presentations, students shared their progress in informal stand-up presentations to their classmates every two weeks. Although all students have recently completed their undergraduate studies, several showed enthusiasm for integrating AI into their future work, emphasizing a commitment to ethical practices and safeguarding data privacy. As advancements in technology continue to enhance AI, certain aspects of this study may eventually become part of history. However, the essence of the research—leveraging AI as collaborative assistants in software project development—is likely to remain a cornerstone of educational relevance for some time.

LIMITATIONS

The data collected for this study is limited in scope, as it was conducted within an educational setting, in a required class. While each student-researcher collected data independently over a fifteen-week period a homogeneous, quasi-experimental setting existed. Conducting a multi-year study was not possible for the undergraduate seniors, as many had not previously developed a complex software project during a course. The student researchers also acknowledge that the presented work includes a limited sample size of four single-case studies which occurred concurrently. The course-wide Likert ranking scale and themes determined by the class, could be seen as both a strength and a possible weakness.

Moreover, the AI selection process was unbounded, allowing students to choose any AI tool that interested them, essentially any AI they thought would be helpful for their project development. While this flexibility led to unexpected conclusions for the students, the resultant data may not offer sufficient depth for researchers outside of an undergraduate educational context. As such, the findings are likely more valuable for understanding the experiential learning process within an academic setting, than for generalizable insights applicable to broader research or professional environments outside of education.

REFERENCES

Bakke, C., & Sakai, R. (2022). Using design-based research to layer career-like experiences onto software development courses. *Journal of Information Technology Education: Innovations in Practice, 21*, 25–60. https://doi.org/10.28945/4988

Bankins, S., Ocampo, A. C., Marrone, M., Restubog, S. L. D., & Woo, S. E. (2024). A multilevel review of artificial intelligence in organizations: Implications for organizational behavior research and practice. *Journal of Organizational Behavior, 45*(2), 159–182. https://doi.org/10.1002/job.2735

Campbell, M. (2020). Automated coding: The quest to develop programs that write programs. *Computer, 53*(2), 80–82. https://doi.org/10.1109/MC.2019.2957958

Christensen, E., & Paasivaara, M. (2022). *Learning soft skills through distributed software development*. ICSSP.

Cresswell, J., & Cresswell, D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed). SAGE Publications.

Durrani, U. K., Akpinar, M., Adak, M. F., Kabakus, A. T., Öztürk, M. M., & Saleh, M. (2024). A decade of progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023). *IEEE Access, 12*, 171185-171204. https://doi.org/10.1109/ACCESS.2024.3488904

Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. *IEEE Software, 40*(4), 30–38. https://doi.org/10.1109/MS.2023.3265877

Hsu, H., Lin, E., Chang, K., & Hsiao, E. (2019). Practicing scrum in institute course. In *Proceedings of the 52nd Hawaii International Conference on System Sciences* (pp. 1–10). https://doi.org/10.24251/HICSS.2019.935

Laval, J., Fleury, A., Karami, A. B., Lebis, A., Lozenguez, G., Pinot, R., & Vermeulen, M. (2021). Toward an innovative educational method to train students to agile approaches in Higher Education: *The A.L.P.E.S. Education Sciences, 11*(6), 267. https://doi.org/10.3390/educsci11060267

Liang, J. T., Yang, C., & Myers, B. A. (2024). A large-scale survey on the usability of AI programming assistants: Successes and challenges. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)* (Article 52, pp. 1–13). Association for Computing Machinery. https://doi.org/10.1145/3597503.3608128

Metrôlho, J. C., Ribeiro, F. R., Batista, R., & Graça, P. (2022). Prepare students for software industry: A case study on an agile full stack project. *Proceedings of the Seventeenth International Conference on Software Engineering Advances (ICSEA 2022),* 75–80. https://doi.org/10.18260/1-2--19756

Millam, A., & Bakke, C. (2024). Coding with AI as an assistant: Can AI generate concise computer code? *Journal of Information Technology Education: Innovations in Practice, 23*, Article 9. https://doi.org/10.28945/5362

O'Reilly, M., & Parker, N. (2013). 'Unsatisfactory Saturation': A critical exploration of the notion of saturated sample sizes in qualitative research. *Qualitative Research, 13*(2), 190-197. https://doi.org/10.1177/1468794112446106

Panetta, K. (2023). *Set up now for AI to augment software development*. Gartner. https://www.gartner.com/en/articles/set-up-now-for-ai-to-augment-software-development

Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023). Do users write more insecure code with AI assistants? *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2785–2799). Association for Computing Machinery. https://doi.org/10.1145/3576915.3623157

Sahin, Y. G., & Celikkan, U. (2020). Information technology asymmetry and gaps between higher education institutions and industry. *Journal of Information Technology Education: Research, 19*, 339-365. https://doi.org/10.28945/4553

Shi, Y., Sakib, N., Shahriar, H., Lo, D., Chi, H., & Qian, K. (2023, June). AI-assisted security: A step towards reimagining software development for a safer future. *Proceedings of the IEEE 47th Annual Computers, Software, and Applications Conference, Torino, Italy*, 991–992. https://doi.org/10.1109/COMPSAC57700.2023.00142

Vaidya, J., & Asif, H. (2023). A critical look at AI-generate software: Coding with the new AI tools is both irresistible and dangerous. *IEEE Spectrum, 60*(7), 34–39. https://doi.org/10.1109/MSPEC.2023.10177044

Vasileiou, K., Barnett, J., Thorpe, S., & Young, T. (2018). Characterising and justifying sample size sufficiency in interview-based studies: Systematic analysis of qualitative health research over a 15-year period. *BMC Medical Research Methodology. 18*, Article 148. https://doi.org/10.1186/s12874-018-0594-7

Zhang, D., Bin Ahmadon, M. A., & Yamaguchi, S. (2022). Human-ai pair programming by Data Stream and its application example. *2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 1–4. https://doi.org/10.1109/icce-asia57006.2022.9954649

APPENDIX

Project Highlights. The project was guided with independent components; previous iterations of project courses have found that guided projects with some independent components and regular sprints results in the highest quality software and greatest completion rate of version 1.

Figure 8 shows the initial goal for all projects to be combined into a single app; however, one group did not complete all individual versions, while the other group did. As a result the instructor modified the expectation to be a combined group project of four independent projects.

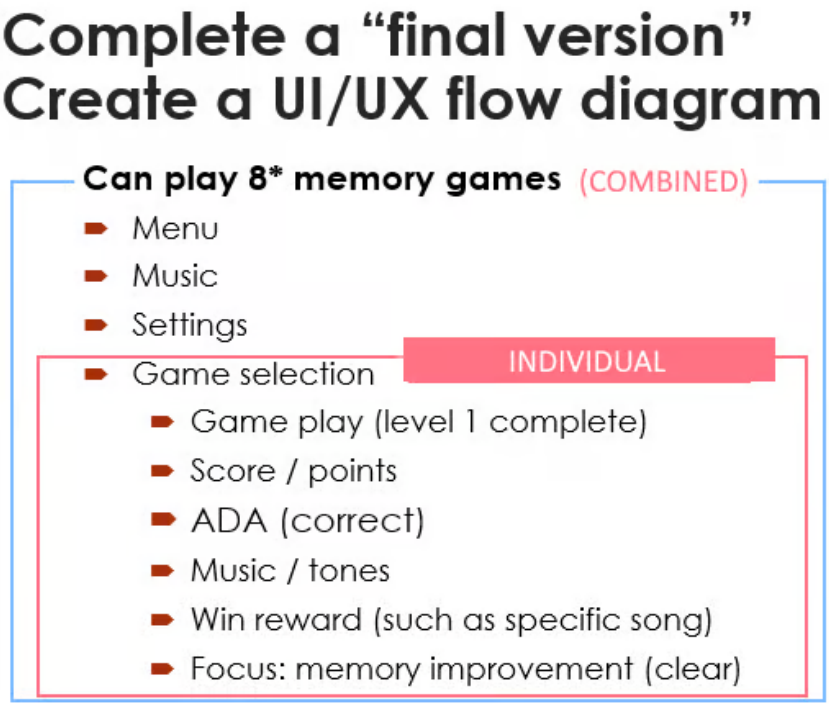


Figure A1: Memory game, individual and small group major tasks. Instructor determined. Original. Modified 8* to 4*; * is that each student is responsible to assist others as time allows, grade is based on individual completion.

During sprint 1 and 2, students designed the UI/UX for their app; after the flow diagram sketch was approved, students could begin coding the individual memory game. Figures A2 – A6 are samples from Sprint 2.

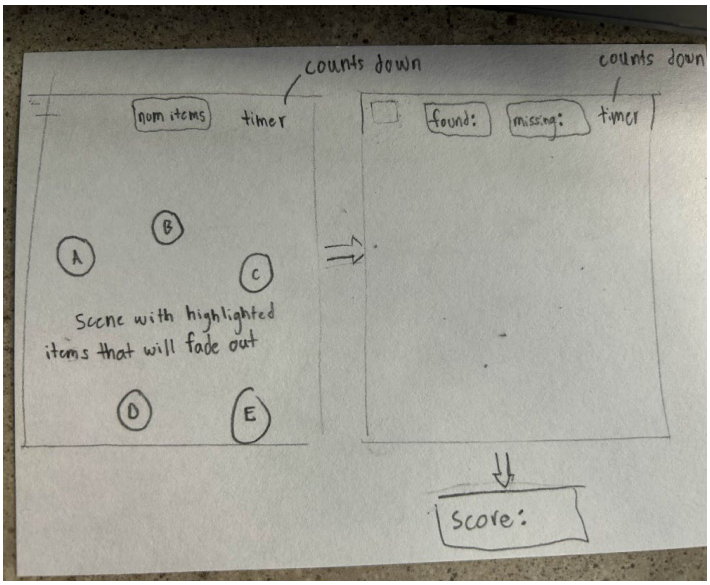


Figure A2: initial, individual UI sketch.

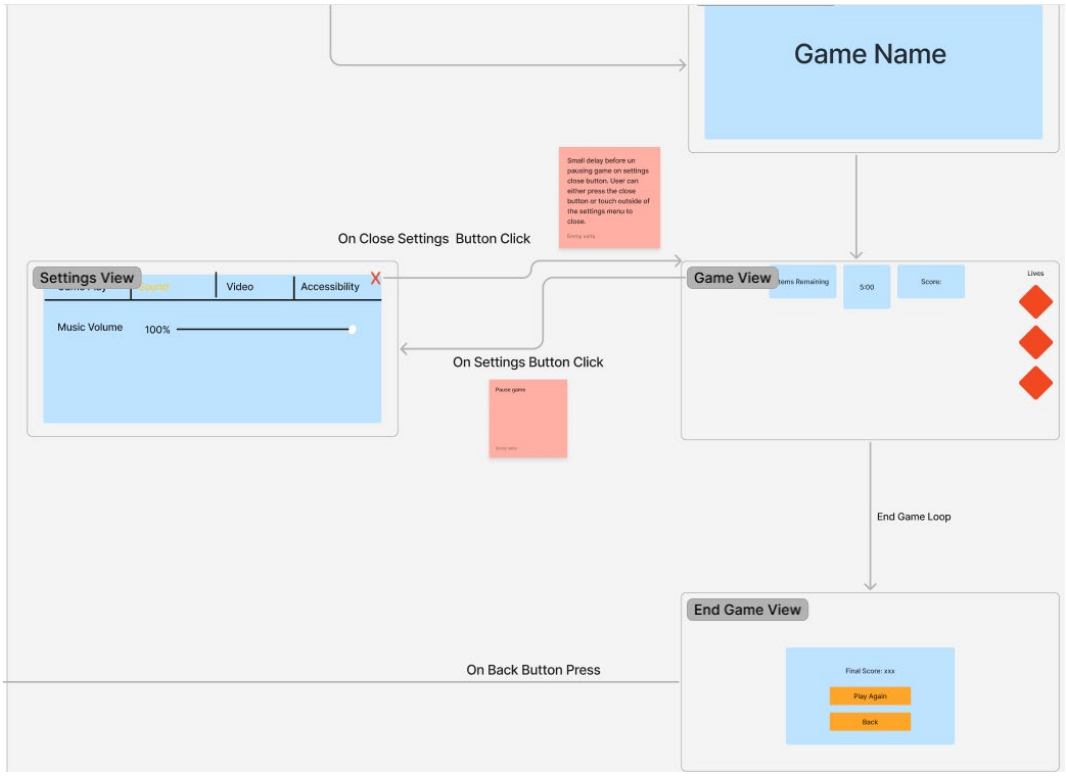


Figure A3: initial, individual, digital UI /UX game flow planning.

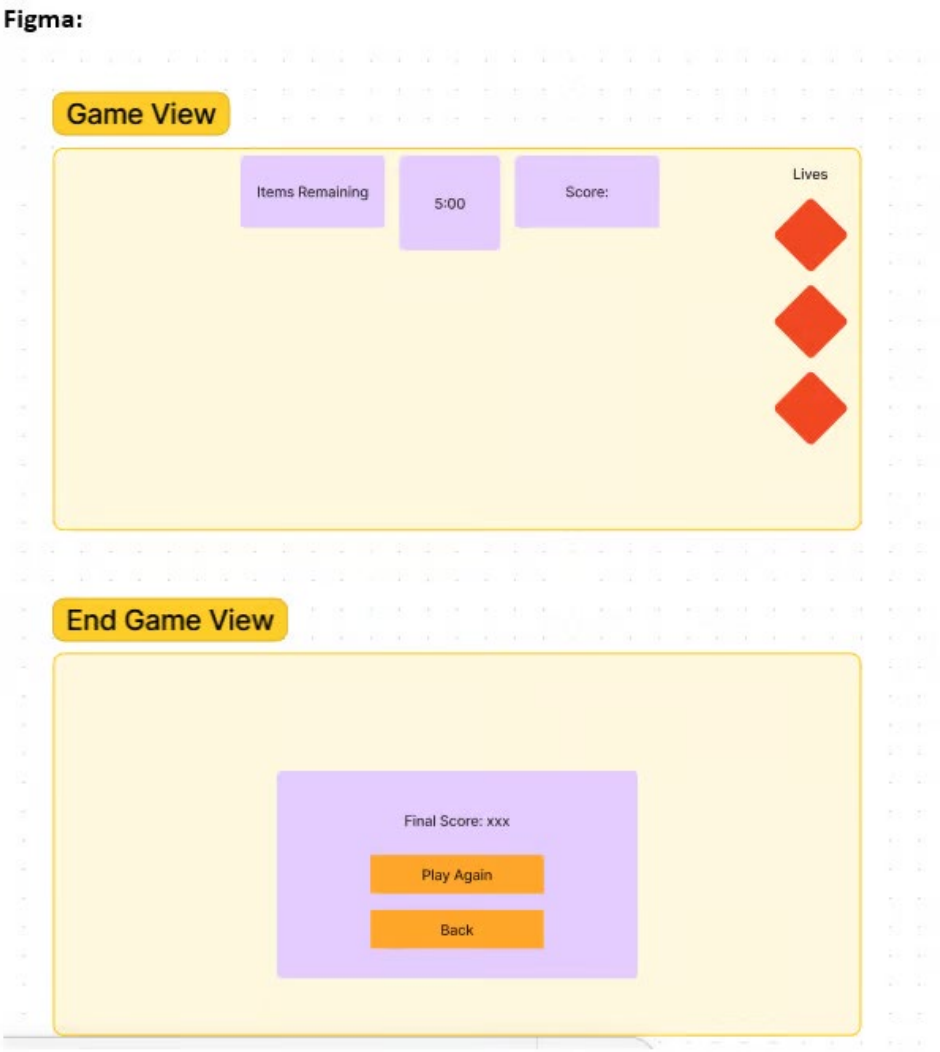


Figure A4: Figma was used by most students to create UI/UX designs.

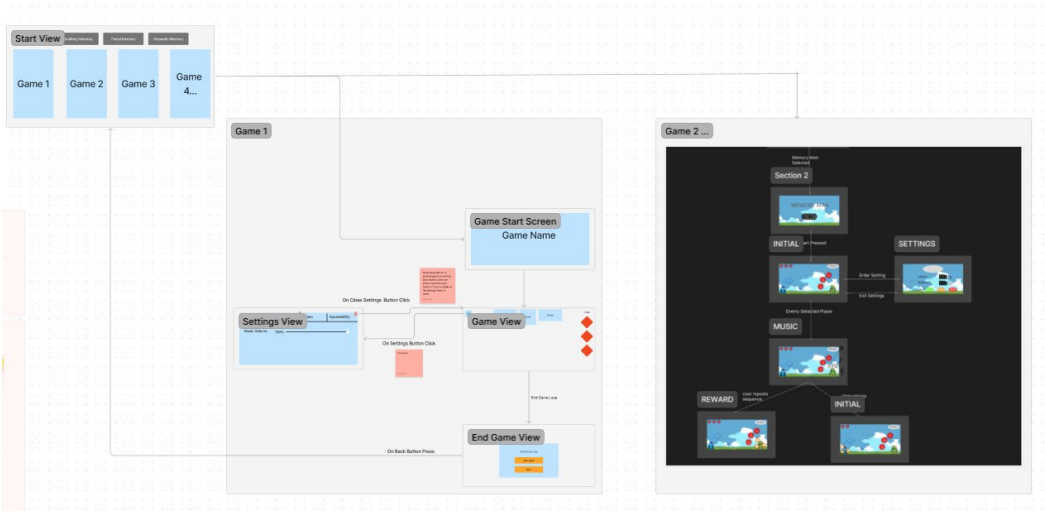


Figure A5: individual UI/UX flow diagram.

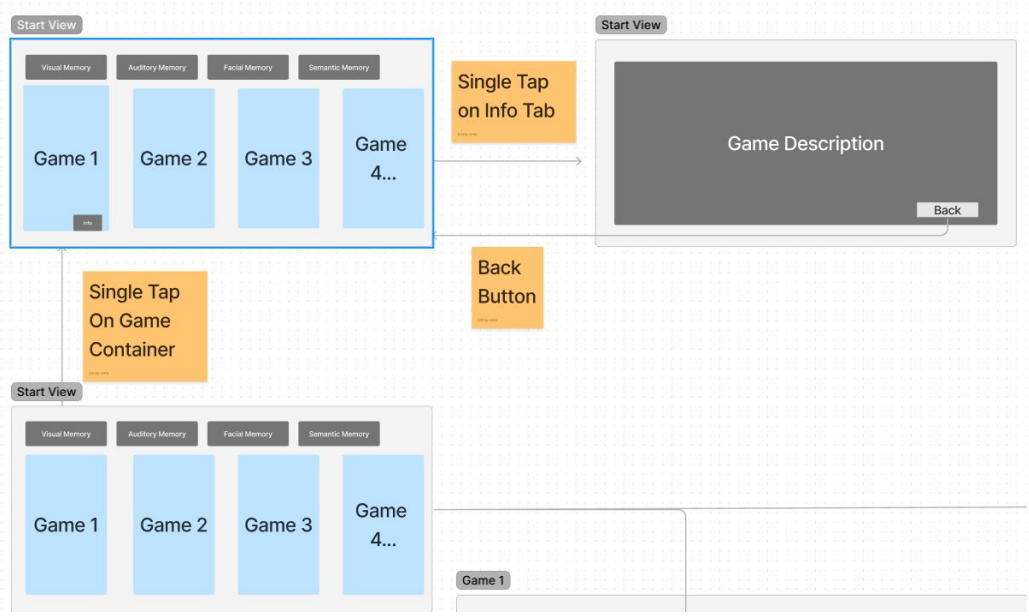


Figure A6: Mock-up of small group menu, combining multiple games.

Examples of stand-up slides for sprint 7 are shown in Figures A7 – A11

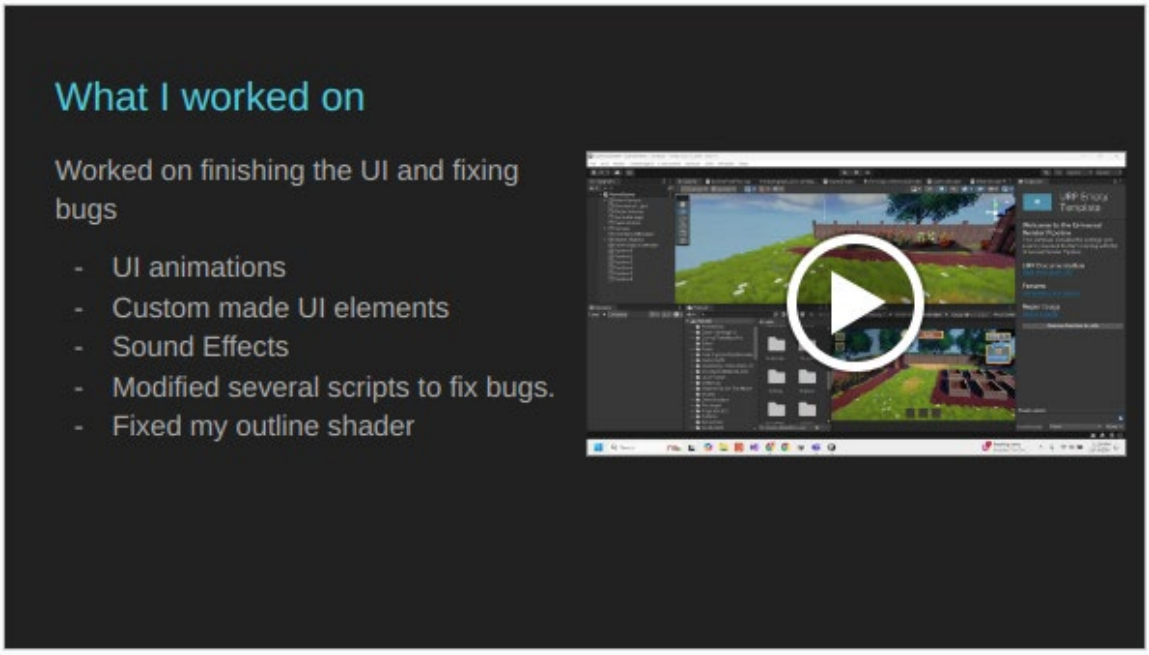


Figure A7: "What I worked on this sprint"

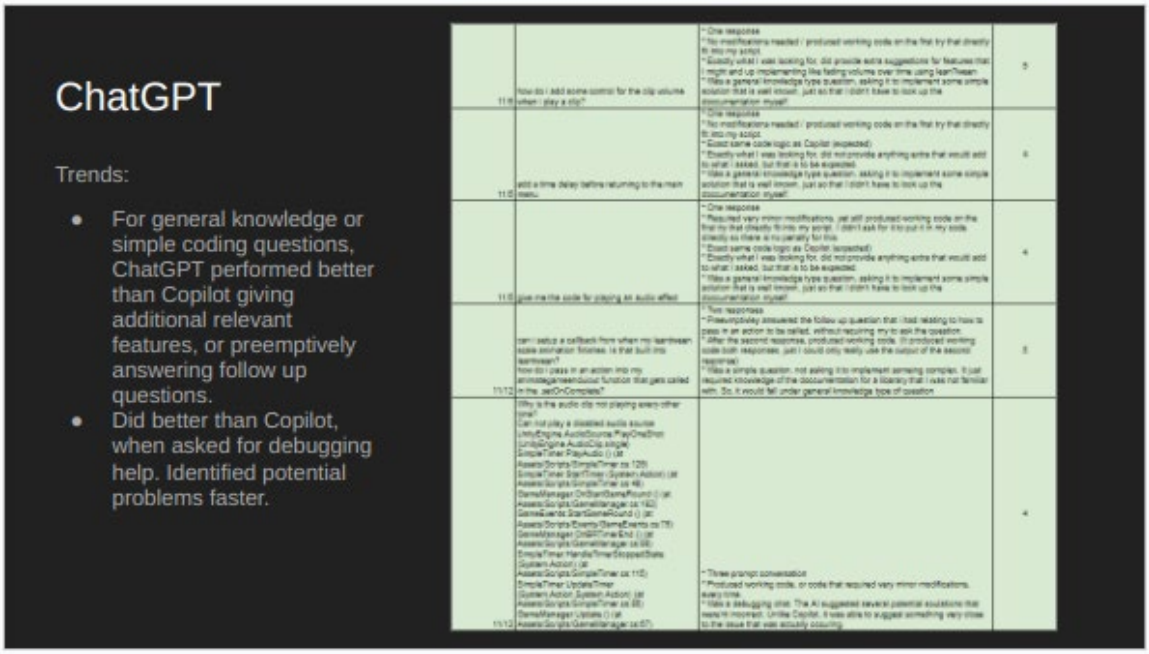


Figure A8: Q&A dataset for this sprint for one of student's AI (ChatGPT)

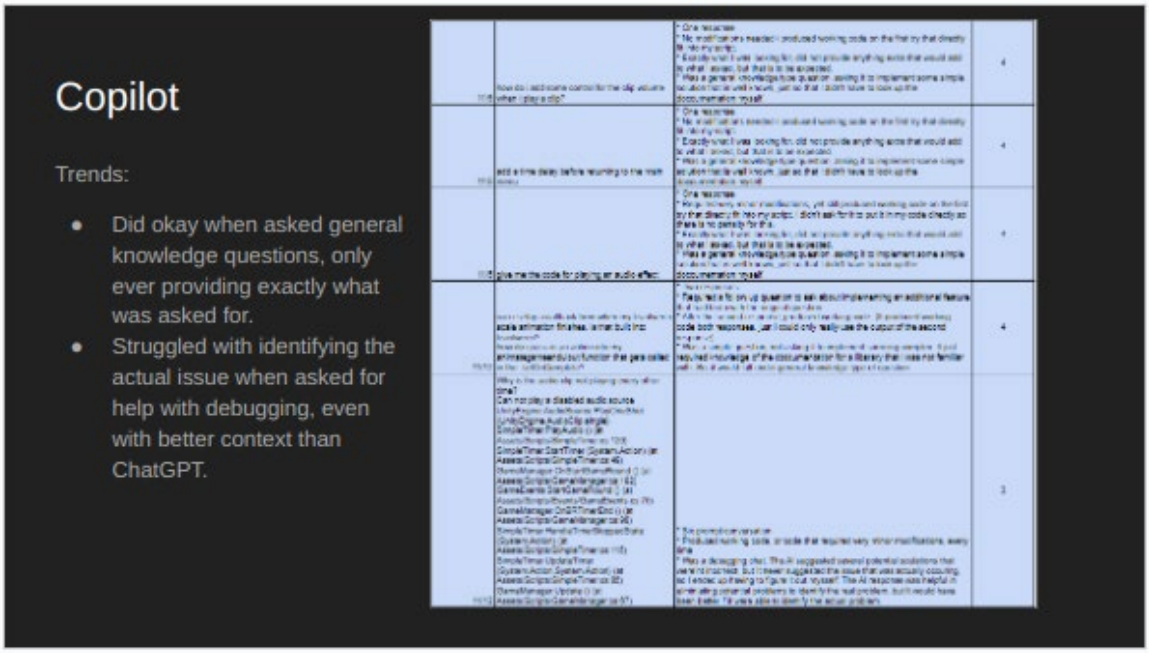


Figure A9: Q&A dataset for this sprint for student's other AI (CoPilot)

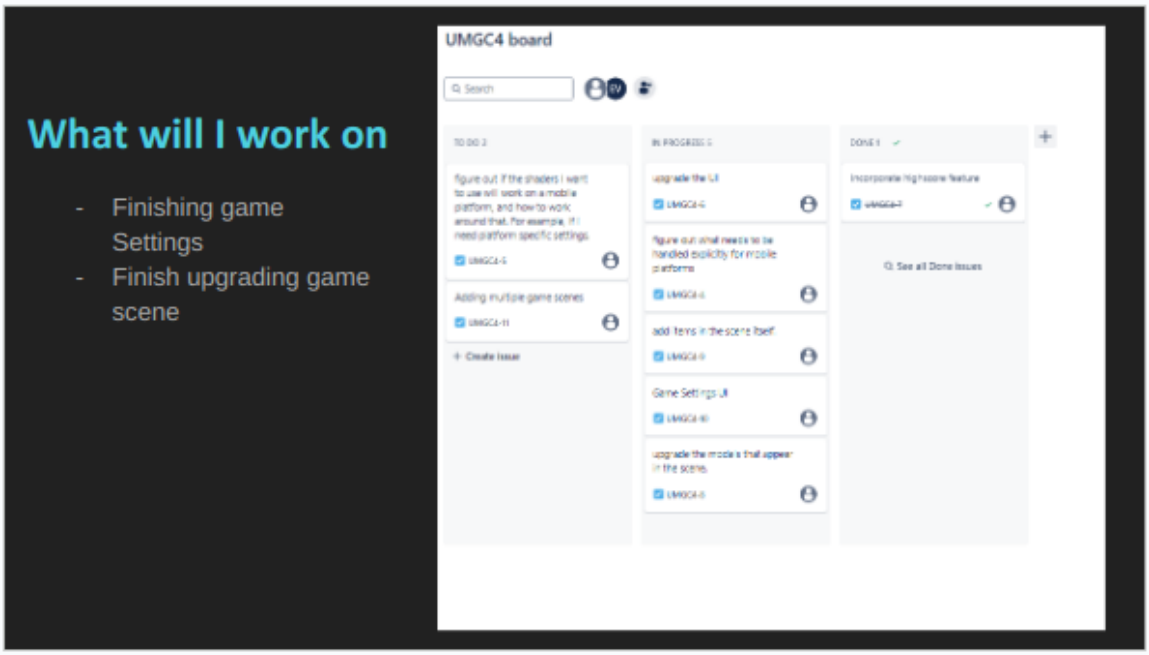


Figure A10: "What will I work on during the next sprint?"

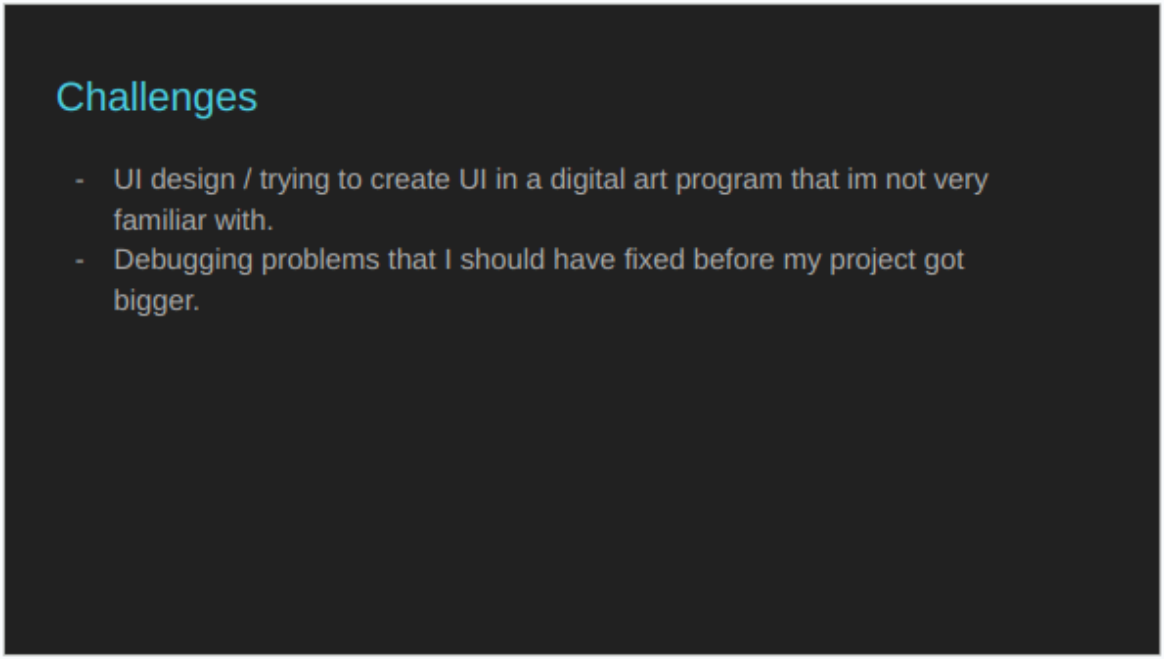
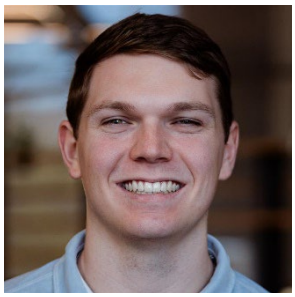


Figure A11: "What challenges am I working on?"

AUTHORS



Micheal Callahan graduated in December 2024 with a Bachelor's degree in Computer Science from Grand Canyon University in Phoenix, Arizona. He is currently working in a data analytics position, pursuing his passion for data science and exploring how new technologies, such as Artificial Intelligence, can be used to improve the field. He has completed several projects, with some highlights including a neural network for deepfake detection, a data analysis project on flight scheduling, and sports analytics.

Joseph Clauss is a Computer Science student at Grand Canyon University in Phoenix, Arizona, with an emphasis in Business Entrepreneurship. With a passion for blending technical innovation and real-world application, Joseph is focused on developing software solutions that support user-centered design and entrepreneurial strategy. His academic interests include software development, game design, AI-assisted learning tools, and full-stack applications for business scalability. Past projects include developing Unity-based memory games, integrating VR into educational simulations, and creating Excel-based financial modeling tools for project analysis. Joseph is driven by the goal of using technology to solve practical problems and is actively exploring how to merge coding expertise with entrepreneurial insight to launch future ventures.

Emmy Voita [Information not available]

Dr. Christine Bakke is an Associate Professor at Grand Canyon University in Phoenix, Arizona. She has been a technology instructor since 2008 and earned her doctorate in 2013, specializing in IT with a focus on robotics and coding within educational settings. Dr. Bakke is passionate about project-based learning, particularly in guided learning environments. Her professional career spans 18 years in the industry, with expertise in networks, cybersecurity, databases, and programming. Her research is centered on integrating academic and professional best practices into agile, active learning methodologies. Past projects include developing Scrum-inspired chatbots, speech-assistant software, memory-assistive learning games, and custom IoT devices equipped with tailored software solutions.