

AI CODE GENERATION TOOLS - EXPERIENCE AND PERSPECTIVES OF
COMPUTER SCIENCE STUDENTS

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAII AT MĀNOA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

AUGUST 2024

By

Tyler G. Anderson

Thesis Committee:

Dr. Anthony Peruma, Chairperson

Dr. Edoardo S. Biagioni

Dr. Daniel Port

Keywords: Computer Science, education, AI, LLM, survey, students

Copyright © 2024 by
Tyler G. Anderson

ABSTRACT

Recent advancements in large language models (LLMs) and generative artificial intelligence (GenAI) have made significant impacts globally. These AI tools have simplified traditionally strenuous and time-consuming tasks, fostering both optimism and concern regarding their use by students in educational settings. This thesis investigates the usage and perceptions of AI code generation tools, particularly ChatGPT, among computer science (CS) students. We conducted a study involving seventy students from varied academic levels who participated in a 45-minute programming activity with the optional use of ChatGPT assistance. This was followed by an extensive online questionnaire designed to explore the AI code generation tools students use, the frequency of their use, the specific tasks for which they employ these tools, and their underlying motivations.

Through comprehensive qualitative and quantitative analysis, we discovered that an overwhelming majority of students are not only familiar with ChatGPT, but also use it for over half their academic work. Students expressed a strong interest in leveraging AI tools to enhance their learning experience, citing benefits such as increased efficiency and deeper understanding of complex concepts. However, they also shared concerns similar to those of their instructors, particularly regarding over-reliance on these tools, difficulty comprehending generated code, and lacking the skills necessary for their future careers. Additionally, we observed a wide range of behaviors in how students use ChatGPT, with most not employing advanced techniques like model priming and prompt engineering strategies.

Our findings highlight the need for greater awareness and training in the design and behavior of these tools to maximize the benefits of AI assistance. We discuss the implications of our findings for students, educators, and the industry, suggesting strategies to ensure that the integration of AI code generation tools in educational contexts results in a net positive impact for all stakeholders.

TABLE OF CONTENTS

Abstract	3
List of Tables	6
List of Figures	7
1 Introduction	9
1.1 Goals & Research Questions	10
1.2 Contributions	11
2 Related Work	12
2.1 Classroom Innovations	12
2.2 Technical Evaluations	13
2.3 Experiences and Perceptions	14
3 Methodology	16
3.1 Overview	16
3.2 Study Participants	17
3.3 Programming Activity	17
3.4 Questionnaire	19
3.5 Pilot Run	23
4 Results	24
4.1 Participants Demographics	24
4.1.1 Academic Year	24
4.1.2 Gender	25
4.1.3 Ethnicity	25
4.1.4 Primary Language	26
4.1.5 Type of Degree	26
4.1.6 Field of Study	27
4.1.7 First Generation College Student	27
4.1.8 Family in Software Industry	28
4.1.9 Years of Java Experience	28
4.1.10 Years of Software Industry Experience	29
4.2 RQ1: To what extent are students familiar with existing AI code generation tools?	29
4.3 RQ2: To what extent are students reliant on AI code generation tools?	32
4.3.1 Students who used ChatGPT	32
4.3.2 Students who elected not to use ChatGPT	35
4.4 RQ3: To what extent are students using AI code generation tools in an educational setting?	37
4.5 RQ4: What perceptions do students have towards AI code generation tools?	39
4.6 RQ5: How do students interact with AI code generation tools?	44
4.6.1 Questionnaire Results	44
4.6.2 ChatGPT Interactions	45
5 Discussion	50

6 Threats to Validity	53
7 Conclusion and Future Work	54
A Activity Instructions	55
B Source code with errors	59
Bibliography	1

LIST OF TABLES

3.1	Questionnaire provided to participants following the programming activity.	20
3.1	Questionnaire provided to participants following the programming activity.	21
3.1	Questionnaire provided to participants following the programming activity.	22
3.1	Questionnaire provided to participants following the programming activity.	23
4.1	Familiarity with AI code generation tools	30
4.2	Perceived difficulty of programming activity for those who rated their confidence in ChatGPT responses as <i>Neutral</i>	34
4.3	Perceived difficulty of programming activity for those who rated their confidence in ChatGPT responses as <i>Somewhat confident</i>	35
4.4	Perceived difficulty of programming activity for those who rated their confidence without ChatGPT as <i>Only slightly confident</i>	36
4.5	Perceived difficulty of programming activity for those who rated their confidence without ChatGPT as <i>Moderately confident</i>	36
4.6	Tool names used by students for programming purposes.	38
4.7	Average rankings of how students use AI code generation tools. A lower numbered ranking signifies a high number of students listing it as a top reason for using the tool.	45

LIST OF FIGURES

3.1	Overview of our empirical study.	16
3.2	Sample question and answers used for the console-based Math quiz application.	18
3.3	Sample display of quiz results.	18
3.4	A matrix-style question used to gather student perception towards AI code generation tools.	19
4.1	Academic years of survey participants.	24
4.2	Genders of survey participants.	25
4.3	Ethnicity of survey participants.	25
4.4	Primary language of survey participants.	26
4.5	Degree pursued by survey participants.	26
4.6	Fields of study of survey participants.	27
4.7	Visual representation of students who are first generation college students.	27
4.8	Visual representation of students with family in the software industry.	28
4.9	Self-declared years of Java experience of survey participants.	28
4.10	Self-declared years of software industry experience of survey participants.	29
4.11	Resources used by students when encountering difficulties while programming.	31
4.12	Perceived difficulty of programming activity for those using ChatGPT assistance.	33
4.13	Level of confidence in correctness of ChatGPT response.	33
4.14	Perceived difficulty of programming activity for those <i>NOT</i> using ChatGPT assistance.	35
4.15	Level of confidence in correctness of non-ChatGPT solution.	36
4.16	Student responses regarding how often they use AI code generation tools to assist with school-related activities.	38
4.17	Student levels of concern across a variety of topics	41
4.18	Student levels of optimism across a variety of topics	43
A.1	Page #1 of the instructions provided to students at the start of the programming activity.	55
A.2	Page #2 of the instructions provided to students at the start of the programming activity.	56
A.3	Page #3 of the instructions provided to students at the start of the programming activity.	57
A.4	Page #4 of the instructions provided to students at the start of the programming activity.	58
B.1	Source code of App.java. This was the driving class for the application, and did not contain any seeded errors. File available at https://github.com/tylera211/thesis/blob/main/App.java	59
B.2	Source code of Question.java. Class variable assignments within the constructor use incorrect references. File available at https://github.com/tylera211/thesis/blob/main/Question.java	60

B.3	Source code of AbstractQuiz.java. Seeded+ errors include an incorrectly placed closing bracket '}', an extra semicolon ';', poor naming conventions for class variables, and missing code calculating and displaying the user score. File available at https://github.com/tylera211/thesis/blob/main/AbstractQuiz.java	61
B.4	Source code of MathQuiz.java, part 1. Seeded errors include a mismatch of class variable names from its superclass, AbstractQuiz.java, an uninitialized array prior to index assignments, an incorrect points assignment of -5, and an incorrect correctAnswer assignment of 'c' instead of 'C'. File available at https://github.com/tylera211/thesis/blob/main/MathQuiz.java	62
B.5	Source code of MathQuiz.java, part 2. Seeded errors include a mismatch of class variable names from its superclass, AbstractQuiz.java, and an uninitialized array prior to index assignments. File available at https://github.com/tylera211/thesis/blob/main/MathQuiz.java	63

CHAPTER 1

INTRODUCTION

Artificial Intelligence (AI) was eloquently defined by the computer scientist John McCarthy as "the science and engineering of making intelligent machines, especially intelligent computer programs" [20]. This term encompasses a broad spectrum of computer programs and systems designed to mimic human intelligence. Recent successes of AI rely on machine learning algorithms and techniques that have matured over decades, evolving in tandem with advancements in computing hardware. Over the years, AI has become a powerful toolkit, used for an array of applications, including but not limited to recommendation engines for online commerce, the safeguarding of financial transactions through fraud detection and prevention, facial recognition systems, and the conversion of voice into text on mobile devices [5][12].

In recent years, there has been a notable emergence of advanced generative AI tools, with increased rates of adoption worldwide [30]. These tools are enticing to programmers largely due to the increased productivity they can provide. For instance, a study on GitHub Copilot, a generative AI toolkit built to assist in software development, highlights various reported benefits, including 74% of developers finding it easier to focus on more satisfying tasks, 88% reporting improved productivity, and 96% experiencing increased efficiency when dealing with repetitive tasks [15].

Commonly listed as the the most popular generative AI tool across the globe [7, 11, 31], ChatGPT is an AI model developed by OpenAI, offering users an immersive conversational experience. In this interaction, users provide natural language prompts, to which the model responds, allowing for the continuation of the conversation through follow-up prompts [24]. ChatGPT's versatility extends to a wide range of applications, including in the realm of computer programming. With the ability to generate and explain code, along with the ability to debug, provide documentation, and a variety of other tasks, ChatGPT proves itself to be a valuable tool in software development [6].

There are an untold number of similar AI tools, either released or in development, that can be used to generate code snippets for programmers [19, 23, 25]. These tools can be extremely useful to programmers by improving their productivity, helping them brainstorm, and assisting in debugging or troubleshooting [3, 8]. However, generative AI is not perfect. It can generate content that fails to meet the needs of the users, provide inaccurate information, or it can hallucinate the existence of content and studies that aren't real [4, 10].

In the realm of education, instructors are rightfully cautious about the adoption of AI code generation tools and the effects they can impose on their students. Despite the aforementioned benefits of generative AI tools, educators are largely concerned that widespread use of these tools will propagate cheating and prevent students from developing the necessary skills needed for software development [14, 16, 29, 32]. Conversely, many educators are exploring ways to leverage AI advancements in the classroom [9, 17, 21, 22].

1.1 Goals & Research Questions

Given the multitude of benefits and concerns surrounding AI code generation tools in Computer Science education, our study delves into the experiences and perceptions of programming students towards these tools. To this end, we ask the following Research Questions (RQs) we aim to answer in this paper:

- **RQ1: To what extent are students familiar with existing AI code generation tools?**

We examine what AI code generation tools students are aware of, use, and are familiar with. This helps us understand the technology students use the most often, so further studies can explore the *why* and *how* regarding their usage.

- **RQ2: To what extent are students reliant on AI code generation tools?**

We examine the likelihood of a student using AI code generation tools for a non-graded task, their reasons for choosing to use these tools, and what impact it has on their perceived difficulty of a task and their confidence in solving it.

- **RQ3: To what extent are students using AI code generation tools in an educational setting?**

We examine how frequently students use AI code generation tools for school-related activities, such as their homework, projects, and exams. This helps us understand if students are primarily using these tools in academic settings, as well as any differences in the tools they choose to use.

- **RQ4: What perceptions do students have towards AI code generation tools?**

We examine whether students perceive AI code generation tools as a resource to use when learning or struggling, or if they believe these tools are a necessary skill set to develop for their future careers. We also examine what concerns students have regarding usage of these tools, and what benefits they believe these tools can provide to them.

- **RQ5: How do students interact with AI code generation tools?**

Lastly, we examine how students engage with ChatGPT to understand the techniques used, their ability to engineer the AI according to their needs, and the type of prompts they provide for these needs.

In this study, **RQ1-RQ4** are primarily addressed using the results of an online questionnaire on AI code generation tools. To answer **RQ5**, we developed a programming activity for students of varying academic levels to accomplish, in which we provide the option to use ChatGPT. Submissions

for this activity that use ChatGPT include the prompt logs of the students, which are qualitatively analyzed for common trends and behaviors.

1.2 Contributions

1. **Educators:** By understanding the frequency with which students use AI code generation tools, how students use them, and what their perceptions are around their use, educators can build lesson plans more appropriately. By understanding how likely students are to use AI code generation tools, educators can develop assignments and examinations that further develop student skills when using these tools, or they can develop them in a way where students cannot rely on this technology.
2. **Students:** Students may find value in comparing their experience and perceptions with AI code generation tools to that of their peers. By studying the different techniques and behaviors when interacting with ChatGPT, students can better understand their own strengths and weaknesses when using this technology. Additionally, students can see what alternative AI code generation tools are being used by their peers in order to discover additional resources they can use themselves, perhaps better fitted to their personal needs and preferences.
3. **Industry:** Academia helps prepare individuals for future careers in their industry. Therefore, it's important for employers to understand how these individuals are being taught, and what skills they are developing prior to their entry in the workforce. This knowledge can help employers develop beneficial on-boarding regimens, or restructure their interview process to focus on the strengths and weaknesses they care most about.

CHAPTER 2

RELATED WORK

While research into ways to improve Computer Science education is nothing new, the recent developments in large language models (LLMs) has created new areas of interest for study. These new studies have explored ways to include new AI technology into course content, evaluations of AI tools performance on traditional problems and assignments, and both experiences and perspectives of students and educators alike in regards to AI code generation tools.

2.1 Classroom Innovations

The power of new AI tools has encouraged creative minds to explore new opportunities in the realm of Computer science education. Some studies in this area focus on widely available tools, and how they can be integrated into coursework or lesson plans. Others instead focus on development of course-specific tools with specialized goals in mind.

With the growing number of publicly available AI tools, many educators have looked for ways to use these tools to improve the educational environment for their students. One such study explores a use-case that many educators may have considered for themselves: using AI as a Teaching Assistant. Anishka et al. tested this possibility by using ChatGPT to both grade student code submissions and provide feedback for an introductory level course [21]. Their findings indicated that ChatGPT may be beneficial in providing student feedback, but unreliable in grading assignments. A similar study by Nguyen and Allan used GPT-4 to generate feedback for student code submissions, and found that the model provided correct feedback the majority of the time, their results being comparable to human-level evaluations. [22].

In a similar fashion to the previous studies, other educators have worked towards developing their own AI-powered tools to use in the classroom. One such example is the development of Prompt Problems, and the Promptly tool, by Denny et al. [9]. This tool provided students with programming problems, and tested the student's ability to provide a correct and detailed prompt to generate a working code solution through ChatGPT. Their findings suggest most students can refine their prompts to get a working solution in just a few attempts, though some may take significantly more. By analyzing student feedback, they also found that this practice introduced students to new coding constructs and techniques, while also enhancing their computational thinking. However, some students displayed a high degree of reluctance towards using AI for programming, believing it would negatively impact their creativity or cause over-reliance.

Another example in this area comes from Harvard University's CS50 course [17]. In the paper by Liu et al., they describe the development of CS50.ai, a multi-tool web application to be used by students in the CS50 course as an alternative to other popular resources, such as ChatGPT, Copilot, and Bing. This tool allows for students to easily explain highlighted code, get improvements for code

styling, and converse with a chatbot for rubber-duck debugging. Their end of semester feedback showed 53% of students loving the tool, 33% of students liking it, 13% feeling neutral, and only 1% disliking it. Their findings for this study demonstrates potential for AI-driven tools to enhance the student’s learning experience in a controlled and meaningful way.

These studies demonstrate a variety of ways in which AI advancements can be used to benefit computer science education. As AI technology continues to advance, new possibilities will present themselves, with creative minds leading the way in classroom innovations. However, it’s important to understand the technical proficiency of these technologies, along with student behaviors and perceptions, to design future innovations in a manner best fitting student needs and preferences.

2.2 Technical Evaluations

Given the concerns of individuals developing over-reliance on AI tools, there has been significant research into how well these tools perform on traditional coursework. Primarily these studies evaluate the efficacy of ChatGPT, given its popularity and ease-of-use compared to other available tools. In a study by Joshi et al., questions from 6 different subjects of Computer Science, and 6 different styles of questions, were fed to ChatGPT to measure the tool’s accuracy in producing a correct answer [13]. Their results ranged from a 70.1% accuracy for questions relating to Data Structures and Algorithms, to a 33.4% accuracy for questions on Database Management Systems, with an average accuracy of 56.9% across all categories. In a similar study by Malinka et al., ChatGPT was compared to student performance in computer security courses [18]. They tested ChatGPT against a variety of examination methods (full-text exams, essays, programming assignments, etc.) with three levels of human involvement when using the AI tool. Their results indicate that ChatGPT is proficient enough to pass computer security courses by simply copying and pasting the material, though it still falls short compared to regular student performance. In a study by Savelka et al., the generative pre-trained transformer (GPT) model text-davinci-003, developed by OpenAI and belonging to the same GPT-3.5 series behind ChatGPT, was tested against a large number of Python programming questions [28]. These questions were split among an introductory-level Python course, an intermediate-level course, and a practical applications course. In their results, text-davinci-003 was unable to pass any of the three courses, scoring 56.1% in the introductory-level course, 67.9% in the intermediate course, and 65.4% in the applications course. They found that questions requiring specific output formatting, utilizing real-world artifacts, or involving chains of reasoning steps, posed the greatest difficulty.

Though these studies demonstrate the weaknesses of new AI technologies, it goes just as far in demonstrating their strengths. For instance, ChatGPT may prove itself unable to pass a CS course on its own, but it’s still very likely that an instructor would not wish for a student to achieve near-passing scores without putting in their own effort. It is also possible that students use ChatGPT and other code generation tools in a manner that would produce better results than shown in these

studies, signaling the importance of studying student interactions with this technology.

2.3 Experiences and Perceptions

The use of AI code generation tools by students is a controversial subject, partially thanks to the mixed benefits and detriments these tools can provide in the classroom environment. Some educators look to leverage the potential of AI technology and adapt it into their courses [21, 22, 17]. Other educators shape their courses to focus on material where AI yields lower performance, such as is outlined in [28]. By understanding the actions of others, we can better shape our own opinions and plans on how we interact with this technology, as evidenced by the numerous studies delving into the experiences and perceptions towards AI code generation tools, particularly in students and educators of programming courses. Lau and Guo were one of the first to conduct this type of study in the wake of ChatGPT’s release in late 2022 [16]. In their study, they interviewed 20 instructors of introductory programming courses, spread across 9 different countries. They asked their participants to envision a future where AI tools could perfectly complete any programming problem, is undetectable by plagiarism detectors, and could explain the code for any free-response type questions. They then asked their participants what they would do in the short-term and long-term to help their students learn. Their findings showed a consistent short-term concern of student cheating, with split views on long-term actions, with some wishing to resist the use of AI tools, and others looking to embrace it. Their findings discuss various proposed methods for both fields of views, such as using paper exams to resist reliance on AI tools, or using AI tools within a course to accommodate open-ended design assignments. A study by Sheard et al. was conducted in the same fashion, interviewing 12 instructors from six institutions across three countries, with similar findings. [29].

A number of other studies have examined the student experience and perspective. In a study by Rogers et al., 70 students completed an online questionnaire, detailing their awareness and use of ChatGPT, along with their attitudes towards ChatGPT [27]. Their results indicate students use ChatGPT primarily to facilitate their own learning, using it as a patient, non-judgemental, personal tutor. They also found that students held mixed opinions on whether ChatGPT could hinder their ability to learn to programming, while the majority were confident in doing assignments without ChatGPT’s assistance, and felt that ChatGPT will revolutionize software development and be used in their future careers. Another study by Amoozadeh et al. analyzed student usage and trust in generative AI for programming tasks [1]. Their findings suggest students use generative AI primarily to understand code and seek assistance rather than using it to write or modify code. Additionally, students are very split on their levels of trust for generative AI when asked about their trust in the system’s output, its predictability, and similar questions. Overall students demonstrated a healthy amount of skepticism towards the accuracy of generative AI models.

These studies do well to paint the current landscape of how students and instructors view AI

code generation tools, and help us understand what students use these tools for. However, studies in this area largely rely on questionnaires or interviews. While students may intend to use code generation tools for a purpose more accepted by the academic world, i.e. explaining concepts and providing self-facilitated learning, these studies don't provide any hard evidence to confirm these claims. This paper aims to fulfill that need by studying exactly how students interact with generative AI tools, while simultaneously confirming the findings of similar studies with our own questionnaire.

CHAPTER 3 METHODOLOGY

3.1 Overview

The objective for this study was to have students engage in a common programming task, with the option of using ChatGPT assistance, followed by a questionnaire. The programming activity was intended to measure how many students chose to use ChatGPT when given the option, and to gather qualitative data on how students form their prompts when using ChatGPT. The online questionnaire was designed to collect quantitative results to be compared with similar studies in the field, and to compare with the qualitative results of the programming activity.

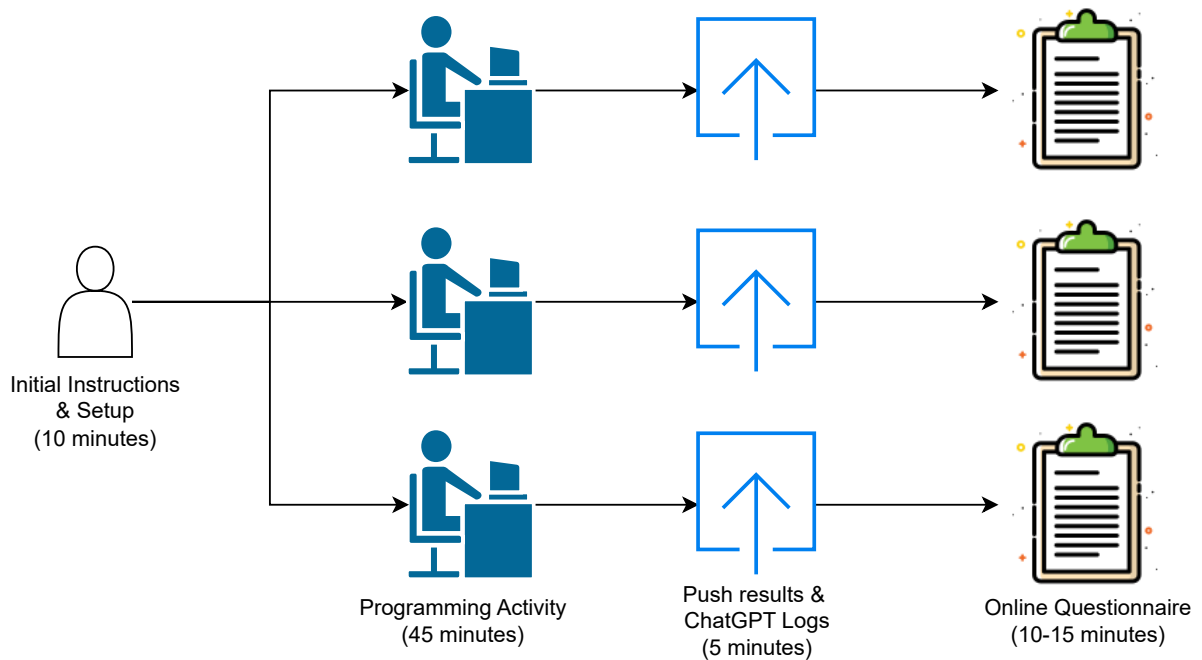


Figure 3.1: Overview of our empirical study.

Participants were provided instructions and guidance between the programming activity and questionnaire, to ensure all participants understood their tasks and to ensure proper data formatting. To avoid distraction and potential behavioral changes due to stress, participants were not monitored during the 45 minute programming activity or the online questionnaire, though the investigator was available for assistance if needed.

Studies were conducted with a mix of batch studies, with small classrooms of students all

participating at the same time, and with individual meetings. All students were presented with the same programming task and time allotment.

3.2 Study Participants

All participants in this study were volunteer students from the University of Hawai'i at Mānoa, recruited through convenience sampling [2]. While there were no requirements in regards to the academic level of the students, participants were encouraged to have developed basic programming skills and be familiar with the Java programming language. To encourage participation, students were given the option of participating in-person at a place of their choosing, or via Zoom. For both formats, we made every effort to accommodate the day and time of the participant's choosing.

Students were recruited through various means. Some students were recruited through referral from university faculty, these students being sent personal invitations via email or Discord. Some students were recruited from an email listing composed of current and previous students in a CS2 course. Some students participated through their instructors volunteering class time to conduct the study with all interested students. In this latter case, students were not required to participate, and could opt-out if desired.

3.3 Programming Activity

The intent of this activity was to give students a relatable programming task, with moderate difficulty for all academic levels, and reasonable to complete in a 45 minute time frame. Thus, we decided to focus on a programming task that involved troubleshooting and debugging existing code. The application provided to participants contained 4 Java files used to create a simple, console-based Math quiz with 10 questions. These questions included "What is $2 + 5$?", "What is $5 * -3$?", and "What is $2*2*2$?", with 7 other questions of similar difficulty. All questions were multiple choice. 2 questions were true/false questions, while the other had 4 options. The application, when run, would display the question to the user, along with the multiple choice options, and wait for user input on their selected answer. After input, the quiz would proceed to the next question. Once all 10 questions were answered, their score would be displayed. Figures 3.2 and 3.3 demonstrate the functionality of the quiz application, and were presented to all students at the start of the activity to help ensure their understanding of the application's functionality.

Within the 4 Java files, a total of 10 seeded errors were created. These include a variety of common programming errors, to include syntax errors, runtime errors, logical errors, typos, and missing code pieces. All students were informed that the Java files had a variety of seeded errors placed within them, and were instructed to make any necessary changes they felt they needed to make in order to get the application in working condition, using the 2 presented images as references of a working quiz.

```
Question #2: What is 8 - 3?  
A) 11  
B) 5  
C) 24  
D) -1  
Please enter your answer: █
```

Figure 3.2: Sample question and answers used for the console-based Math quiz application.

```
Question #10: What is -9 / -3?  
A) -27  
B) -3  
C) 3  
D) 27  
Please enter your answer: C  
  
-----  
Your score: A | 100.00 % | 50/50 points
```

Figure 3.3: Sample display of quiz results.

The programming activity was conducted using GitHub Classroom and GitHub Codespaces. Students were provided with a GitHub Classroom assignment link which, when accepted, would create a copy repository with the seeded errors under their own GitHub account. They would then use GitHub Codespaces as their IDE during the allotted time frame. Students were provided instruction and individual guidance as needed to ensure proper setup of their repository and GitHub Codespaces IDE prior to beginning the activity. Lastly, students were informed that they may utilize ChatGPT to complete the assignment, using whatever means they prefer, with the request that we obtain a copy of their ChatGPT prompt log upon completion of the coding activity. Students were discouraged from using any other outside resources, however, this was not enforced since the students were not monitored during the activity.

Once all the students had their GitHub Codespaces environment properly setup, and they confirmed their understanding of what the goal of the activity was, they were given 45 minutes to work on the assignment. Students were kept informed of the time either through occasional announcements or a display of the 45 minute timer counting down. An investigator was available during the activity to address student questions, without giving them answers on how to complete the task. These questions were mostly confirming for the student whether they completed the task or not.

Once the 45 minutes were up, or the student managed to complete the task early, they were giving instruction to commit all changes they made during the activity, and push their changes to their GitHub repository. If the student chose to use ChatGPT during the assignment, they

were also given instruction on including their ChatGPT prompt log within their repository. This process involved the student finding the shared link for their discussion, opening the link in a Google Chrome tab, and saving the page as an “HTML Only” file format. This was done to ensure the prompt log would be available for analysis at a later date. The URL generated by ChatGPT for sharing their individual discussion is reliant on the student not deleting the discussion at a later date, which we did not want to risk happening.

3.4 Questionnaire

Upon completion of the programming activity, participants were provided a link to an online questionnaire. This questionnaire was created and hosted on the Qualtrics platform, and contained a total of 31 questions. Of these questions, 15 were used to collect demographic information, while the remaining questions were aimed at answering our research questions. 7 questions were used to gather additional information relating to the student’s performance during the programming activity, 6 questions were used to gather information on student experiences using AI code generation tools, and 3 questions were used to measure student perception around AI code generation tools. Of the 3 questions focusing on student perceptions, 2 were multiple part questions with a total of 13 parts. Figure 3.4 illustrates one of these multiple part questions.

How concerned are you regarding students (including yourself) using AI code generation tools that result in the following?

	I haven't thought about it	Not concerned at all	Somewhat concerned	Concerned	Very concerned
Plagiarism	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Over-reliance on AI tools	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Poor quality of AI-generated code	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Poor student understanding of generated code	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Students not developing creative problem-solving and critical-thinking skills	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Legal or ethical issues	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="text"/>					

Figure 3.4: A matrix-style question used to gather student perception towards AI code generation tools.

Questions were a mix of multiple choice, text entry, Likert scales, and ranking questions. These questions were designed to obtain a mix of qualitative and quantitative data. While there was no time limit for the questionnaire, we estimate that it takes 10-15 minutes to complete on average, largely depending on the level of detail provided in the text entry questions.

Table 3.1: Questionnaire provided to participants following the programming activity.

No.	Question	Type	Required	Notes
1	Please include your GitHub username.	Text entry	Yes	Used for thematic analysis between questionnaire results and programming submission
2	What is your current academic year?	Single-choice	Yes	Includes "Other" text entry option
3	What best describes your gender?	Single-choice	Yes	Includes "Other" text entry option
4	What best describes your ethnicity?	Multiple choice	Yes	Includes "Other" text entry option
5	What is your primary language for both spoken and written communication?	Dropdown	Yes	
6	What type of degree are you pursuing?	Single-choice	Yes	Includes "Other" text entry option
7	What is your primary field of study?	Single-choice	Yes	Includes "Other" text entry option
8	Are you a first generation college student?	Single-choice	Yes	
9	Do you have an immediate family member who has worked or is working in the software industry? (this includes internships)	Single-choice	Yes	
10	How many years of Java programming experience do you have?	Single-choice	Yes	

Table 3.1: Questionnaire provided to participants following the programming activity.

No.	Question	Type	Required	Notes
11	How many years of experience in the software industry do you have? (this includes internships)	Single-choice	Yes	
12	What is your preferred programming language?	Single-choice	Yes	Includes "Other" text entry option
13	What is your primary Integrated Development Environment (IDE) ?	Single-choice	Yes	Includes "Other" text entry option
14	Which AI code generation tools are you familiar with?	Multiple choice	Yes	Includes "Other" text entry option
15	What resources do you usually use for help with resolving difficulties you encounter while programming?	Text entry	Yes	
16	How would you rate the difficulty of the programming assignment you just participated in?	Single-choice	Yes	
17	In the programming activity you just participated in, did you choose to utilize ChatGPT?	Single-choice	Yes	
18	Why did you choose to use ChatGPT for this coding activity?	Text entry	Yes	Question only displays if "Yes" is selected for Question 17.
19	How confident are you with the correctness of the response you received from ChatGPT?	Single-choice	Yes	Question only displays if "Yes" is selected for Question 17.
20	Please elaborate on your answer choice to the above question.	Text entry	Yes	Question only displays if "Yes" is selected for Question 17.

Table 3.1: Questionnaire provided to participants following the programming activity.

No.	Question	Type	Required	Notes
21	How confident are you in the correctness of your solution for the programming activity?	Single-choice	Yes	Question only displays if "No" is selected for Question 17.
22	Please elaborate on your answer choice for the above question.	Text entry	Yes	Question only displays if "No" is selected for Question 17.
23	How often do you use AI code generation tools to assist you with school-related activities? This includes activities like homework, projects, exams, etc.	Single-choice	Yes	
24	Which AI code generation tools do you have experience using for programming purposes?	Multiple choice	Yes	Question only displays if "Never" is NOT selected for Question 23. Includes "Other" text entry option.
25	Please let us know the courses in which you have used AI code generation tools	Text entry	Yes	Question only displays if "Never" is NOT selected for Question 23.
26	Were AI code generation tools permitted by your instructors for the previously listed courses?	Single-choice	Yes	Question only displays if "Never" is NOT selected for Question 23.
27	What is the single most frequent reason for you to use AI code generation tools?	Text entry	Yes	Question only displays if "Never" is NOT selected for Question 23.
28	Please rank the statements below to most accurately represent why you utilize AI code generation tools for programming assistance	Ranking	Yes	Question only displays if "Never" is NOT selected for Question 23. The entry from Question 27 defaults to the top ranked item for this question (can be changed)

Table 3.1: Questionnaire provided to participants following the programming activity.

No.	Question	Type	Required	Notes
29	Which of these statements do you identify with most?	Single-choice	Yes	
30	How concerned are you regarding students (including yourself) using AI code generation tools that result in the following?	Matrix	Yes	Includes "Other" text entry option.
31	How optimistic are you regarding students (including yourself) using AI code generation tools that result in the following?	Matrix	Yes	Includes "Other" text entry option.

3.5 Pilot Run

We conducted 3 pilot runs of the combined coding activity and follow-up questionnaire. The participants for these pilot runs were 3rd year students and lab assistants at the University of Hawa'i at Mānoa. These students were well-versed in the Java programming language due to their roles as lab assistants, and were used to gauge the difficulty and clarity of the study. The primary feedback received from participants in the pilot study was confusion towards the code structure in the programming activity. Based on this feedback, some elements that were not ultimately necessary for the completion of the activity, such as multiple constructors for the provided Java classes, were removed. Additionally, small comments were included for various code pieces and methods to reduce the amount of time students needed to comprehend the existing code that was provided to them. Lastly, the difficulty of the quiz questions was reduced so students could more easily test their solution and calculate the correct answers to the quiz questions (even though the correct answers were provided in the code). While no feedback was provided on the questionnaire, some questions were added or modified based on the gathered results. The results of these pilot runs are not included in the findings of this study.

CHAPTER 4

RESULTS

4.1 Participants Demographics

For our study, we obtained a total of 67 results for the programming activity, and 70 results for the online questionnaire. This mismatch in numbers derived from the batch studies we conducted in a classroom environment, where instructions were provided to a larger number of students simultaneously. It became evident that some students did not commit and push their changes from the activity to their GitHub repository, resulting in their programming activity submission remaining the same as the original code provided. This went unnoticed, and those students continued on to complete the questionnaire, without having a matching repository submission. The demographics provided here are according to the 70 students who completed the online questionnaire.

4.1.1 Academic Year

Of our 70 participants, the majority ($N = 37$, 52.9%) were in their Freshman (1st) year. 6 students (8.6%) were in their Sophomore (2nd) Year, 18 students (25.7%) were in their Junior (3rd) year, 3 students (4.3%) were in their Senior (4th) year, and 6 students (8.6%) were Graduate students. Figure 4.1 illustrates this breakdown.

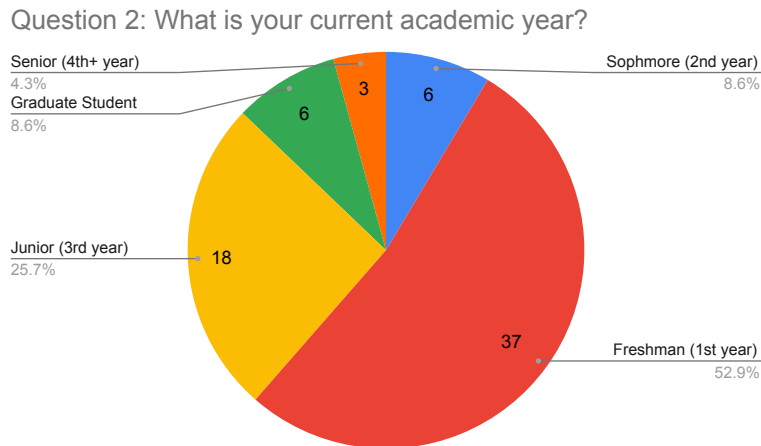


Figure 4.1: Academic years of survey participants.

4.1.2 Gender

48 participants (68.6%) of our survey identified as Male, 21 participants (30.0%) identified as Female, and 1 participant (1.4%) identified as Non-binary/third gender. Figure 4.2 illustrates this breakdown.

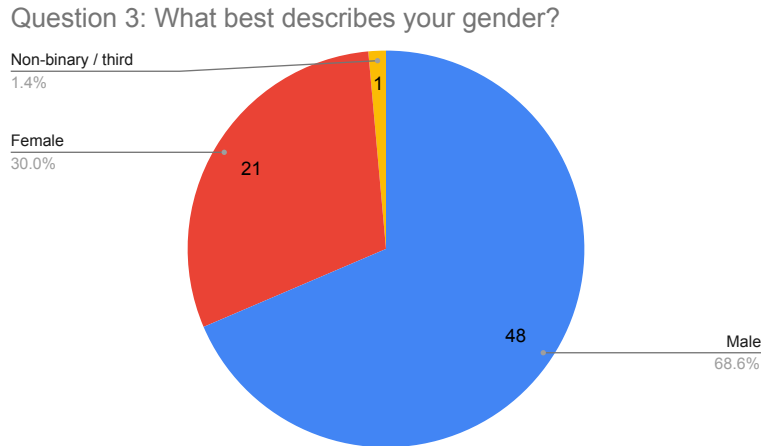


Figure 4.2: Genders of survey participants.

4.1.3 Ethnicity

The majority of our participants (N = 43, 61.4%) identified themselves as Asian ethnicity, with the next largest category identifying as Asian, White or Caucasian (N = 9, 12.9%). The remaining 19 participants were composed of another 10 categories, which can be seen in Figure 4.3.

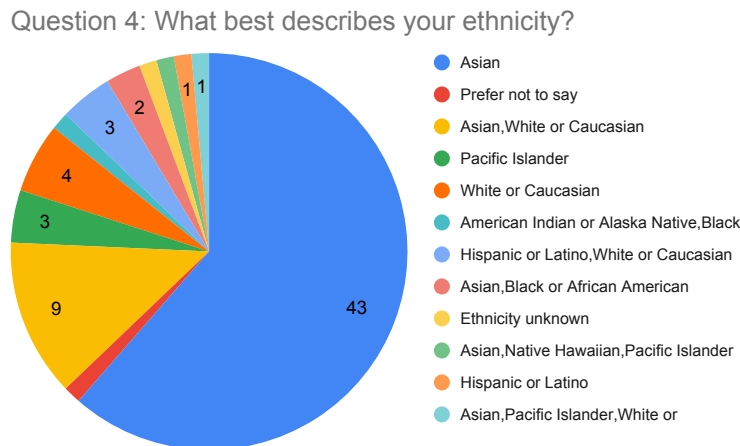


Figure 4.3: Ethnicity of survey participants.

4.1.4 Primary Language

The majority of our participants ($N = 64$, 91.4%) selected English as their primary language, with a small number of other students selecting other languages. 2 students (2.9%) selected Chinese, 2 students (2.9%) selected Korean, 1 student (1.4%) selected Japanese, and 1 student (1.4%) selected Tagalog.

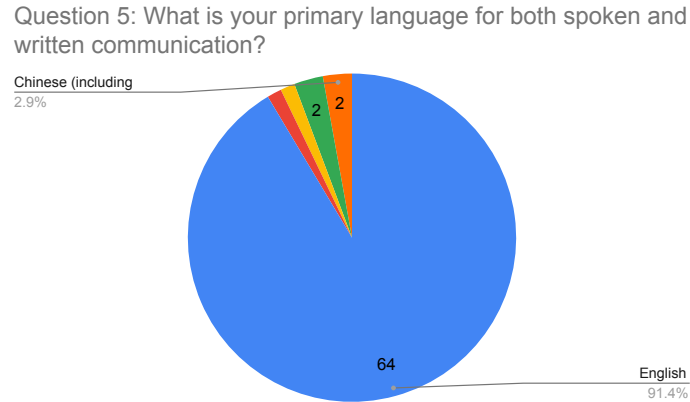


Figure 4.4: Primary language of survey participants.

4.1.5 Type of Degree

The majority of our participants ($N = 62$, 88.6%) were undergraduate students pursuing a Bachelor's Degree. For graduate students, 7 participants (10.0%) were pursuing a Master's Degree, and 1 student (1.4%) was pursuing a Doctoral Degree.

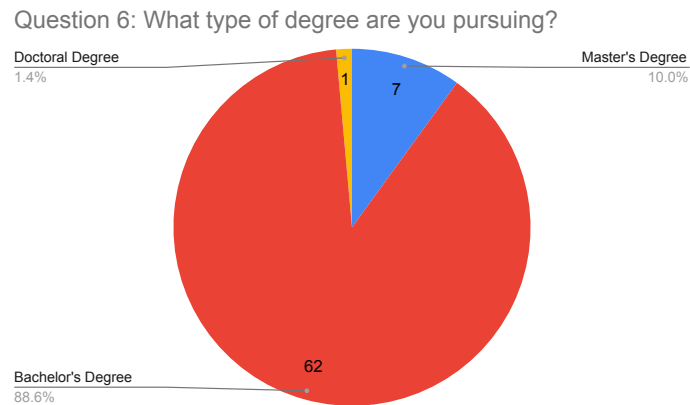


Figure 4.5: Degree pursued by survey participants.

4.1.6 Field of Study

The majority of our participants chose Computer Science (N = 50, 71.4%) or Information and Computer Sciences (N=11, 15.7%) as their field of study. All other selections include 3 students (4.3%) under Management Information Systems, and 1 student each (1.4%) for Mechanical Engineering, Computer Engineering, Math Data Science, Mathematics, Accounting, and Linguistics.

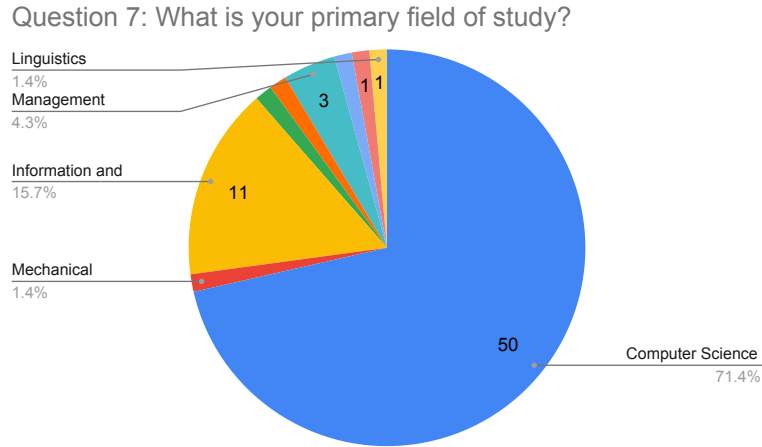


Figure 4.6: Fields of study of survey participants.

4.1.7 First Generation College Student

The majority of our participants (N = 50, 71.4%) responded that they were NOT first generation college students, while 19 participants (27.1%) responded that they were, and 1 participant (1.4%) elected not to say.

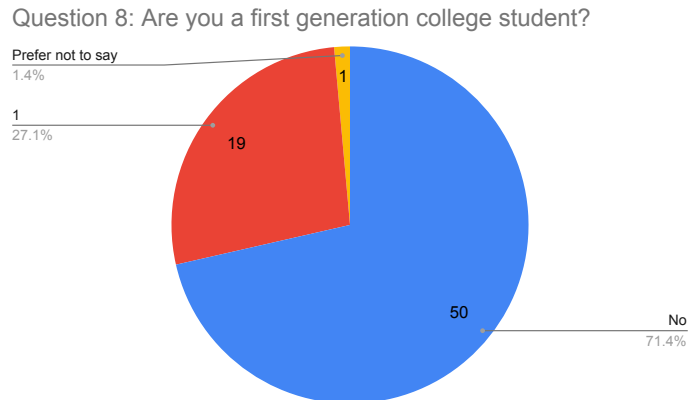


Figure 4.7: Visual representation of students who are first generation college students.

4.1.8 Family in Software Industry

The majority of our participants (N = 55, 78.6%) responded that they did not have any immediate family members in the software industry. The remaining participants (N=15, 21.4%) selected that they did have immediate family members in the industry.

Question 9: Do you have an immediate family member who has worked or is working in the software industry?

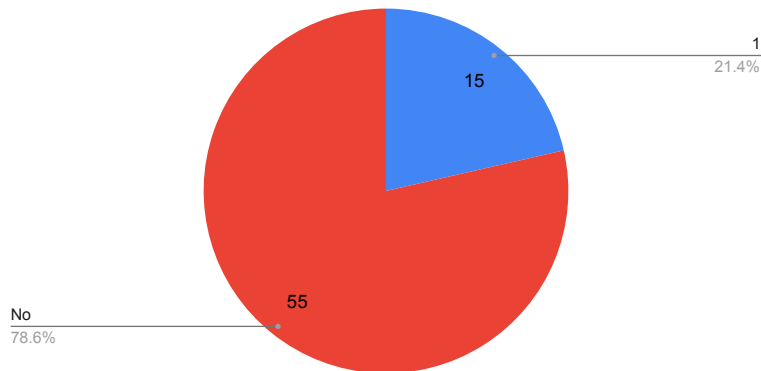


Figure 4.8: Visual representation of students with family in the software industry.

4.1.9 Years of Java Experience

The majority of our participants (N = 47, 67.1%) responded that they have less than 1 year of experience coding in the Java language. 16 participants (22.9%) claimed to have 1-2 years experience, and 7 participants (10.0%) responded with 3-5 years experience. There were no participants in this study with more than 5 years experience in Java.

Question 10: How many years of Java programming experience do you have?

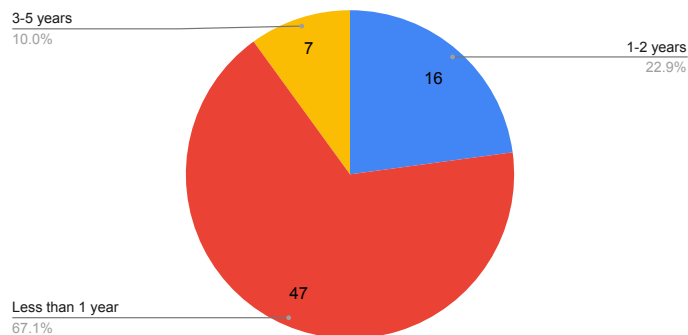


Figure 4.9: Self-declared years of Java experience of survey participants.

4.1.10 Years of Software Industry Experience

The majority of our participants ($N = 40$, 57.1%) claimed to have at least some level of experience working in the software industry, either as an intern or a regular hire. Of these participants, 34 (48.6%) claimed less than 1 year of experience, 5 (7.1%) claimed 1-2 years experience, and 1 individual (1.4%) claimed 3-5 years experience. The remaining participants ($N=30$, 42.9%) did not have any experience in the software industry at the time of this survey.

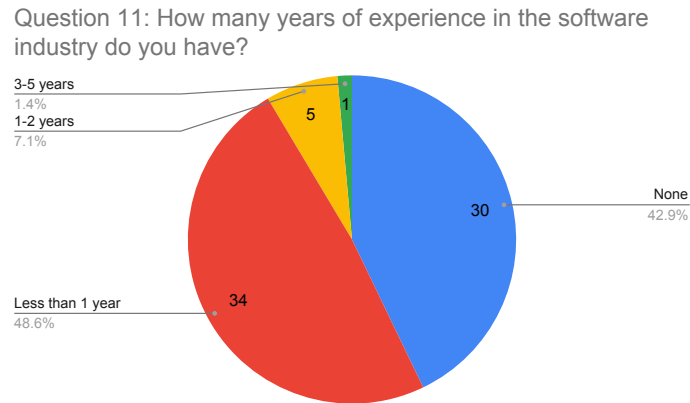


Figure 4.10: Self-declared years of software industry experience of survey participants.

4.2 RQ1: To what extent are students familiar with existing AI code generation tools?

To address this RQ, we analyzed the responses to questions #14 and #15 from our online questionnaire, regarding AI code generation tool familiarity and resources used to resolve difficulties while programming.

Question #14 is a multi-choice question listing 8 AI code generation tools, with additional options for None and Other, the latter containing text entry to manually include any additional AI tools the participant may be familiar with.

As demonstrated in Table 4.1, the overwhelming majority ($N = 67$, 95.71%) of students claimed to be familiar with ChatGPT. Following ChatGPT, the order of most frequent to least frequent selections are as follows: GitHub Copilot ($N = 13$, 18.57%), Gemini ($N = 7$, 10.00%), Bing Chat ($N = 4$, 5.71%), Replit GhostWriter ($N = 2$, 2.86%), Claude ($N = 2$, 2.86%), OpenAI Codex ($N = 1$, 1.43%), Phind ($N = 1$, 1.43%), Julius AI ($N = 1$, 1.43%), and lastly both Tabnine and Amazon CodeWhisperer both with 0 selections. The selections for Claude, Phind, and Julius AI were all entered through the Other selection of the multiple choice options. A total of 3 participants (4.29%) selected None, indicating they were not familiar with any AI code generation tools.

Table 4.1: Familiarity with AI code generation tools

AI Code Generation Tool	Count	Percentage
ChatGPT	67	95.71%
GitHub Copilot	13	18.57%
Gemini (formerly Bard)	7	10.00%
Bing Chat	4	5.71%
Replit GhostWriter	2	2.86%
Claude	2	2.86%
OpenAI Codex	1	1.43%
Phind	1	1.43%
Julius AI	1	1.43%
Tabnine	0	0.00%
Amazon CodeWhisperer	0	0.00%

With the exception of the 3 None responses, every participant who was familiar with an AI code generation tool included ChatGPT in their selection. Furthermore, 47 participants (67.14%) indicated they were *only* familiar with ChatGPT, selecting it as their lone option. The average number of AI code generation tools a participant was familiar with was 1.40, with only nine participants selecting 3 different options, one participant selecting 4 options, and no participants selecting more than 4 options.

Question #15 asks students to list resources they typically use when encountering difficulties while programming, providing participants a text entry box for open-ended responses. Among the results, we annotated 7 categories of resources used by students, described as follows:

- **Artificial Intelligence:** Responses in this category made mention of one or several AI code generation tools. The majority of these responses only listed ChatGPT. Five students listed additional AI code generation tools in their responses, including among them GitHub Copilot, Blackbox AI, and Claude.
- **Stack Overflow:** Responses in this category specifically mentioned Stack Overflow as a resource, with no further details provided.
- **Subject Matter Experts:** Responses in this category made reference to other individuals who were used for assistance. Responses varied in nature, with answers including family members, friends, and educators.
- **YouTube:** Responses in this category generally listed YouTube specifically as a resource. One student listed *videos* as a resource, with no context provided on which website or platform was used.
- **Google Search:** Responses in this category generally mentioned Google specifically as a resource, while some were more vague in nature, describing *searching* the internet for resources.

- **Online Documentation:** Responses in this category made mention of *documentation* or *documents*, without providing details. Some responses in this category listed official websites for programming languages (e.g., “*The Java Website*”) as their resources.
- **Course Material:** Responses in this category gave mention to course-specific material, such as textbooks, lecture slides, and class examples provided by their instructors.

Among these categories, the most common resource mentioned was Artificial Intelligence (N = 51, 72.86%), where all but one of these responses indicated ChatGPT specifically as a tool used. Following this, the most common resources included Stack Overflow (N = 20, 28.57%) and Google Search (N = 17, 24.29%). Further results regarding the frequency of each category can be seen in Figure 4.11.

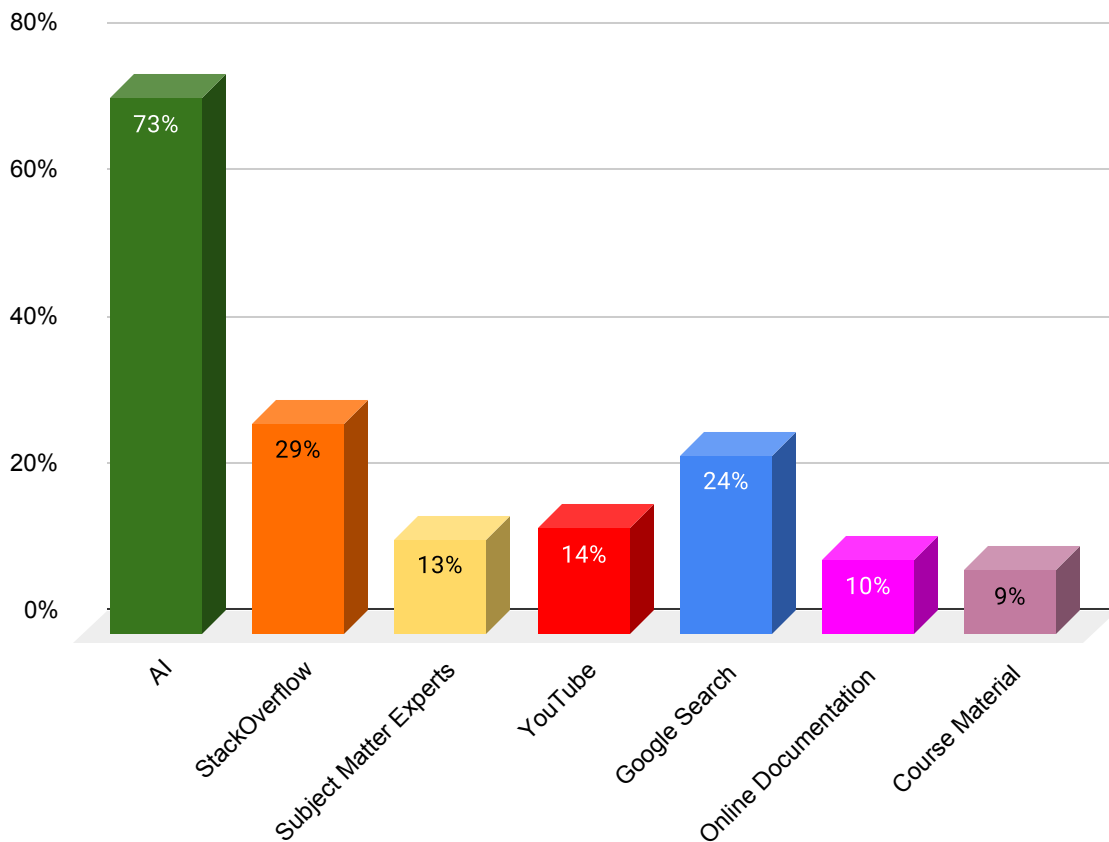


Figure 4.11: Resources used by students when encountering difficulties while programming.

Summary for RQ1. Participants were extremely likely to be familiar with ChatGPT, and unlikely to be familiar with any other AI code generation tool. Among alternative AI code generation tools, students were most familiar with GitHub Copilot and Gemini (formerly Bard). A small number of students included AI tools not originally provided in our listing, the most popular being Claude. When asked more broadly to describe the resources used while programming, the majority of students made mention of AI tools, particularly ChatGPT, with all other types of resources being used much less frequently.

4.3 RQ2: To what extent are students reliant on AI code generation tools?

For this RQ, we analyzed the answers to Questions #16-22 from the online questionnaire. These questions focused on the participant's perceived difficulty of the programming activity, whether they elected to use ChatGPT in the activity, and their confidence in their solutions with or without ChatGPT assistance.

For Question #17, participants were asked *In the programming activity you just participated in, did you choose to utilize Chat-GPT?*. For this question, *Yes* was selected by 50 participants (71.43%), indicating that they elected to use ChatGPT in the activity. The remaining 20 participants (28.57%) selected *No*, indicating ChatGPT was not used for their solutions. In the online questionnaire, Questions #18-20 were presented only to those who selected *Yes*, while Questions #21 and #22 were only presented to those who selected *No*. To analyze the extent of reliance students have towards AI tools, we compare the results between these two groups of participants.

4.3.1 Students who used ChatGPT

All participants were asked to select their perceived level of difficulty for the programming activity on a 4-point Likert scale, ranging from *Not at all difficult*, to *Very difficult*. For the 50 participants who elected to use ChatGPT in the activity, *Not at all difficult* was selected by 2 participants (4.00%), *Only slightly difficult* was selected by 17 participants (34.00%), *Moderately difficult* was selected by 22 participants (44.00%), and *Very difficult* was selected by 9 participants (18.00%). This data is demonstrated in Figure 4.12.

Only participants who selected *Yes* for using ChatGPT in the activity were presented with Question #19, asking the participant for their level of confidence with their ChatGPT response being the correct solution, ranging from *Very unconfident* to *Very confident* on a 5-point Likert scale.

Of the 50 participants presented with this question, 0 selected *Very unconfident*, 6 (12.00%) selected *Somewhat unconfident*, 13 (26.00%) selected *Neutral*, 21 (42.00%) selected *Somewhat confident*, and 10 (20.00%) selected *Very confident*. This data is demonstrated in Figure 4.13.

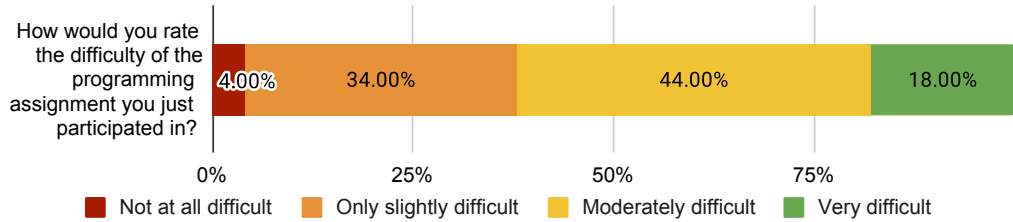


Figure 4.12: Perceived difficulty of programming activity for those using ChatGPT assistance.

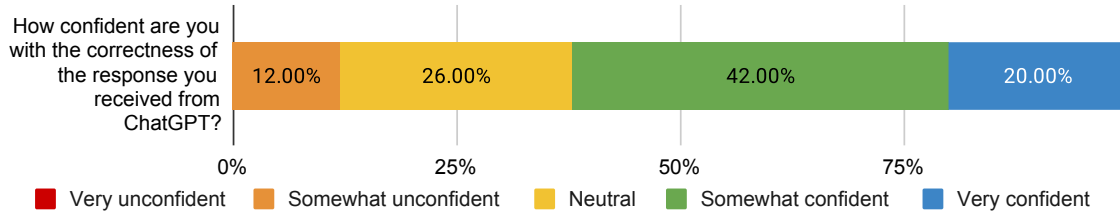


Figure 4.13: Level of confidence in correctness of ChatGPT response.

Additionally, these participants were presented with Question #18, asking the participant to explain why they chose to use ChatGPT. We performed a thematic analysis of their reasoning, categorized below:

- Activity Restrictions:** To better understand how students use AI code generation tools, participants in the study were limited to using ChatGPT for outside assistance. 4 participants (8.00%) listed this restriction as the primary reason for deciding to use ChatGPT in the activity, suggesting they would not have used the tool given other options (e.g., *“I choose to use ChatGPT because the instructions said that I couldn’t use any other resource.”*).
- Coding Assistance:** 25 participants (50.00%) provided reasoning that described AI playing the role of a coding assistant. Participants in this category generally stated benefits relating to productivity (e.g., *“I used it because it allows me to pinpoint the issues within the code quicker.”*) or identifying problems in the code (e.g., *“I was struggling to figure why parts of the code weren’t working so I chose to ask ChatGPT”*).
- Familiarity:** 14 participants (28.00%) cited their familiarity with ChatGPT, or its ease of use, as their primary reason for electing to use the tool (e.g., *“I choose to use ChatGPT because I’m more familiar with it and it could provide answers based on the questions that I asked.”*).
- Knowledge Constraints:** 13 participants (26.00%) provided reasoning that described a gap in the knowledge required for the task, either from inexperience with the material or requiring

Table 4.2: Perceived difficulty of programming activity for those who rated their confidence in ChatGPT responses as *Neutral*

	Count	Percentage
Not at all difficult	0	0.00%
Only slightly difficult	5	38.46%
Moderately difficult	5	38.46%
Very difficult	3	23.08%

some form of a reminder (e.g., “*Asking syntax related questions such as defining variables, etc. Not for directly writing code and copy/pasting*”).

Lastly, participants were presented with Question #20, asking the participant to elaborate on their level of confidence for the correctness of ChatGPT’s responses. Based on the results of Question #19, the majority of our participants selected either *Neutral* (N = 13, 26.00%) or *Somewhat confident* (N = 21, 42.00%). We focus our analysis on the reasoning behind these two groups and their levels of confidence.

For those who selected *Neutral* in response to Question #19, 5 participants (38.46%) rated the activity as *Only slightly difficult*, 5 participants (38.46%) rated it as *Moderately difficult*, and 3 participants (23.08%) rated it as *Very difficult*. These figures are demonstrated in Table 4.2

When asked to elaborate on why they selected *Neutral* confidence, participants described the potential faultiness of ChatGPT, while having a tendency to blame themselves for misuse of the tool (e.g., “*The responses from ChatGPT provided me with the wrong solution or thought process. It could be because of the phrasing of my question.*”) or from their inability to fully understand the solution it provided to the student (e.g., “*The program ended up working as it was intended but at the same time it I did not complete it within my frame of actually understanding how to solve it.*”).

For those who selected *Somewhat confident* in response to Question #19, 1 participant (4.76%) rated the activity as *Not at all difficult*, 4 participants (19.05%) rated it as *Only slightly difficult*, 13 participants (61.90%) rated it as *Moderately difficult*, and 3 participants (14.29%) rated it as *Very difficult*. These figures are demonstrated in Table 4.3

These participants gave similar responses to those who selected *Neutral*, showing awareness of ChatGPT’s inconsistencies and ability to generate faulty code. However, these participants seemed to better understand the concept of prompt engineering and providing all relevant details (e.g., “*ChatGPT works very well in providing solutions containing base definitions of things, however it lacks the ability to piece different pieces of logic/code together.*”). This group, too, described difficulties understanding the code generated by ChatGPT (e.g., “*I do not have enough expertise to judge whether the code chatGPT provided is correct. However, the code can run so I suppose it’s ok.*”).

The results from these two groups suggests most students understand ChatGPT, and AI as a whole, is imperfect and subject to faulty responses. Students also struggle to understand the

Table 4.3: Perceived difficulty of programming activity for those who rated their confidence in ChatGPT responses as *Somewhat confident*

	Count	Percentage
Not at all difficult	1	4.76%
Only slightly difficult	4	19.05%
Moderately difficult	13	61.90%
Very difficult	3	14.29%

solutions generated by ChatGPT, or struggle to provide the necessary prompt engineering to create a working solution through the AI tool. Given that the majority of students rated the activity for our study as either *Moderately difficult* or *Very difficult*, there may be a correlation between the student’s confidence in AI tools relative to their perceived difficulty of the task presented to them. However, this was not explored further in our study.

4.3.2 Students who elected not to use ChatGPT

A total of 20 participants (28.57%) indicated that they chose not to use ChatGPT during the programming activity. Of these individuals, 3 participants (15.00%) indicated the activity was *Not at all difficult*, 7 (35.00%) found it *Only slightly difficult*, 7 found it *Moderately difficult*, and 3 found it *Very difficult*. This data is demonstrated in Figure 4.14.

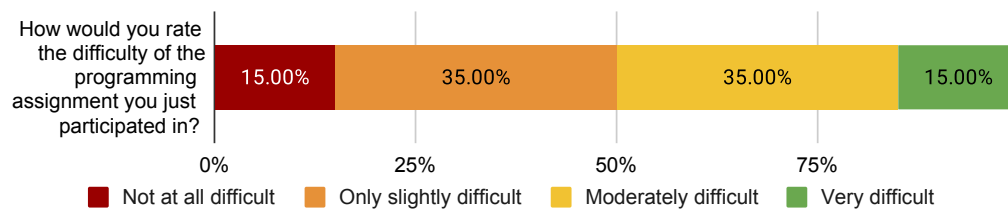


Figure 4.14: Perceived difficulty of programming activity for those *NOT* using ChatGPT assistance.

These participants were presented with Questions #21 and #22 in the questionnaire, as an alternative to Questions #18-20 for those who used ChatGPT. Questions #21 and #22 follow a similar format to Questions #19 and #20, asking participants to rate their level of confidence in their submission, and then elaborating. For Question #21, 2 participants (10.00%) rated themselves as *Not at all confident* in their solution, 5 (25.00%) participants selected *Only slightly confident*, 12 participants (60.00%) selected *Moderately confident*, and only 1 participant (5.00%) selected *Very confident*. This data is demonstrated in Figure 4.15.

Once again, we focus our analysis on the two largest selections to understand the reasoning behind their level of confidence. For the participants who elected not to use ChatGPT, these two groups are those who selected *Only slightly confident* (N = 5, 25.00%) and *Moderately confident* (N = 12, 60.00%).

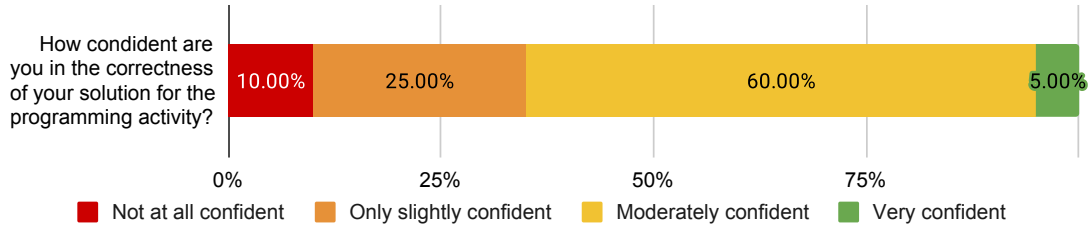


Figure 4.15: Level of confidence in correctness of non-ChatGPT solution.

Table 4.4: Perceived difficulty of programming activity for those who rated their confidence without ChatGPT as *Only slightly confident*

	Count	Percentage
Not at all difficult	0	10.00%
Only slightly difficult	1	20.00%
Moderately difficult	3	60.00%
Very difficult	1	20.00%

For the 5 participants who selected *Only slightly confident* in response to Question #21, 1 participant (20.00%) rated the activity as *Only slightly difficult*, 3 participants (60.00%) rated it as *Moderately difficult*, and 1 participant (20.00%) rated it as *Very difficult*. These figures are demonstrated in Table 4.4.

When asked to elaborate on their level of confidence, participants explained that they were able to make some changes and corrections towards completing the activity, but ultimately were unable to create working code (e.g., “While I did remove the errors present, my code could not run and I did not know why it would.”).

For the 12 participants who selected *Moderately confident* in response to Question #21, 3 participants (25.00%) rated the activity as *Not at all difficult*, 5 participants (41.67%) rated it as *Only slightly difficult*, and 4 participants (33.33%) rated it as *Moderately difficult*. These figures are demonstrated in Table 4.5.

When asked to elaborate on their level of confidence, many participants once again described their ability to complete some, but not all, of the activity. However, these participants had a

Table 4.5: Perceived difficulty of programming activity for those who rated their confidence without ChatGPT as *Moderately confident*

	Count	Percentage
Not at all difficult	3	25.00%
Only slightly difficult	5	41.67%
Moderately difficult	4	33.33%
Very difficult	0	0%

far greater number who were able to complete the activity, or at least progressed to the point of achieving code free of runtime errors and only missing minor features (e.g., “*I’m not completely done with it, as I haven’t finished implementing the displayed score. Other than that, I have plenty of confidence that it works*”). Generally, participants in this grouping who found the activity less difficult than others described a tendency to test their code and aim for higher code quality (e.g., “*I tested my code 3 times before I announced I was finished, and renamed some variables to make the code cleaner.*”).

The results from these two groups of participants suggest a correlation between perceived difficulty of the task, and the confidence in their solution. This is not a surprising result. However, the students elaborating on what behaviors resulted in the varying levels of confidence provides some meaningful insight, namely the value of students thoroughly testing their code, and showing some concern regarding the quality of their code.

Summary for RQ2. When presented with the option of using ChatGPT assistance for the assigned programming activity, the majority of students took it, despite the activity not being graded. For the students who elected to use ChatGPT, most only held a moderate level of confidence in the tool’s performance. These students generally understood the limitations of the tool, but cited trouble in understanding the code it generated. The majority of students who elected not to use ChatGPT felt moderately confident in their own solutions. Among these students, those who placed a higher value on code quality and thorough testing perceived the activity as less difficult than their peers.

4.4 RQ3: To what extent are students using AI code generation tools in an educational setting?

To address this RQ, we analyzed the responses to Questions #23 - 26 from our online questionnaire, regarding the frequency and nature of students using AI code generation tools in their courses.

Question #23 is a single-choice, 5-point Likert-scale question asking *How often do you use AI code generation tools to assist you with school-related activities? This includes activities like homework, projects, exams, etc..* Among the 70 responses we received to this question, 6 students (8.57%) answered with *Never*, 18 students (25.71%) answered *Sometimes*, 18 students (25.71%) answered *About half the time*, 22 students (31.43%) answered *Most of the time*, and 6 students (8.57%) answered *Always*. This data is visually represented in Figure 4.16. Participants who selected *Never* in response to Question #23 proceed to Question #27, while all other participants are presented with Questions #24 - 26.

Interestingly, of all 70 students presented with this question, all 6 who answered *Never* identified as Freshman (1st year) students. Of the 6 students who answered *Always*, the majority (N = 4, 66.67%) identified as Juniors (3rd years), with the remaining two students being a Senior (4th year)

Table 4.6: Tool names used by students for programming purposes.

AI Code Generation Tool	# of students	Percentage of students
ChatGPT	64	100.00%
Gemini (formerly Bard)	8	12.50%
GitHub Copilot	4	6.25%
OpenAI Codex	3	4.69%
Bing Chat	1	1.56%
Replit GhostWriter	1	1.56%
Tabnine	0	0.00%
Amazon GhostWhisperer	0	0.00%
Other	8	12.50%

and a Freshman (1st year).

Question #23: How often do you use AI code generation tools to assist you with school-related activities?

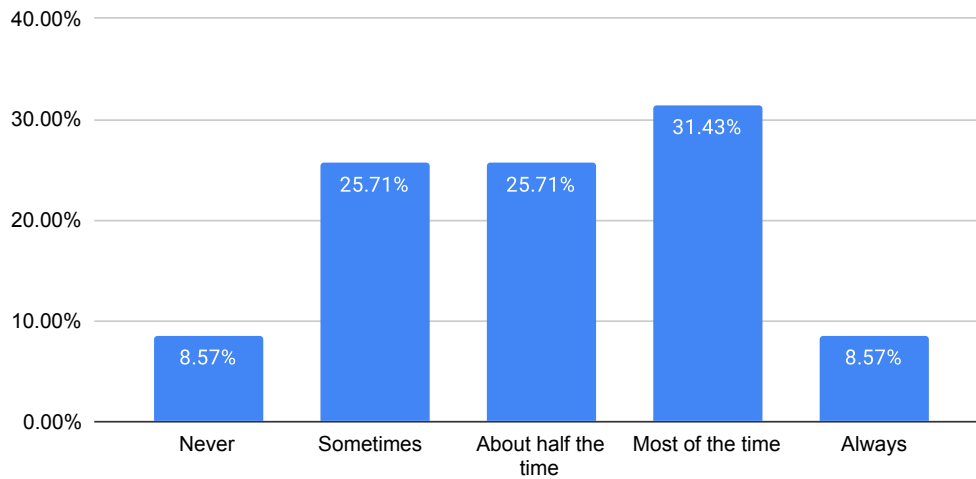


Figure 4.16: Student responses regarding how often they use AI code generation tools to assist with school-related activities.

Question #24 is a multi-choice question listing 8 AI code generation tools, with an additional option for Other, containing text entry to manually include any additional AI tools the participant may use for programming. Though a similar question was asked for Question #14, Question #24 is aimed at seeing which tools students *use*, rather than simply what tools they are familiar with.

Of the 64 students who were presented with this question, all of them (100.00%) selected ChatGPT in their response. All other selections were drastically lower, with the next closest selection being Gemini (formerly Bard) with 8 selections (12.50%). Table 4.6 represents the frequency of each selection presented to the survey participants. Among these participants, 8 selected *Other* in

their responses and included AI tools not originally listed. Of these responses, the AI tools Phind and Claude were the most common, with 2 responses each. Other entries included Blackbox, Julius AI, Ollama, and WRTN, each with a single entry.

Question #25 asks participants to list which courses they've used AI code generation tools in, providing a text-entry box to write their replies in. Question #26 is a follow-up to this question, asking if AI code generation tools were permitted by the instructors of the courses they listed. The results for Question #25 were extremely varied, with 18 responses (28.13%) not listing any specific courses, and 11 responses (17.19%) listing non-CS courses. Of the responses that included CS course numbers, 27 responses (42.19%) listed lower-level courses in the 100-200 range, while 14 responses (21.88%) listed higher-level courses at 300 and above. Of the specific courses listed, ICS 111 and ICS 211 were the most common, these being CS1 and CS2 course equivalents offered at the university. As most of the students for our study were first-year students, the proportion of these courses relative to others was to be expected.

For Question #26, the vast majority (N=45, 70.31%) responded *Yes*, indicating AI code generation tools were permitted in their courses. Among the other responses, 18 students (28.13%) stated that *Some courses, but not all* permitted AI code generation tools, and only 1 student (1.56%) stated that they were *not* permitted.

Summary for RQ3. Our data suggests that students at all academic levels are reliant on AI code generation tools for at least half of their school-related activities. Among the students who claimed to use AI code generation tools in their work, all of them claimed to have used ChatGPT, with all other AI tools appearing in much lower numbers of use or familiarity. Most students claimed that AI code generation tools were permitted in the courses in which they were used. When asked to list which courses the tools were used in, answers ranged from introductory CS1 courses, all the way through graduate-level courses.

4.5 RQ4: What perceptions do students have towards AI code generation tools?

To address this RQ, we analyzed the responses to Questions #29 - 31 from our online questionnaire, touching on the levels of optimism and concern students feel towards AI code generation tools.

Question #29 asks students to identify with one of two beliefs:

1. AI code generation tools are a short-term solution to use while learning to program
2. AI code generation tools are a long-term skill set to be used in the programming industry

Among our 70 responses, 26 students (37.14%) reported AI code generation tools as a short-term solution, while the remaining 44 (62.86%) reported AI code generation tools as a skill set to develop.

We then asked students to rank their levels of concern over a variety of topics for Question #30. Each topic allowed for students to select *I haven't thought about it* if they held no views on the topic, or a 4-point Likert scale of concern, ranging from *Not concerned at all* to *Very concerned*. Here we analyze the results of each topic:

1. **Plagiarism:** The most common level of concern for Plagiarism by our participants was *Somewhat concerned* (N = 27, 39.13%), with the least being *Very concerned* (N = 7, 10.14%). Though students show some concern regarding plagiarism, this topic held the second lowest number of students that were either *Concerned* or *Very concerned*, and the second largest selections for *I haven't thought about it*, suggesting that this is a low area of concern for students relative to other topics.
2. **Over-reliance on AI tools:** The most common level of concern for over-reliance by our participants was *Concerned* (N = 21, 30.43%). However, this was followed closely by both *Somewhat concerned* and *Very concerned*, each having 19 (27.54%) selections. The lowest selection was *I haven't thought about it*, with only 4 (5.80%) ratings, the lowest number of any selection for Question #30, indicating this is a topic students have absolutely thought about, and have a high level of concern for.
3. **Poor quality of AI-generated code:** The most common level of concern over poor quality of AI-generated code by our participants was *Concerned* (N = 22, 31.88%), followed very closely by *Somewhat concerned* (N = 21, 30.43%). The lowest number of selections for this topic was *Very concerned* (N = 5, 7.25%). These results suggest students are only moderately concerned about the quality of AI-generated code, which may be related to student understanding that AI tools can produce faulty code, but it's largely dependent on proper prompt engineering.
4. **Poor student understanding of generated code:** The most common level of concern over students not understanding AI generated code by our participants was *Concerned* (N = 24, 34.78%), followed somewhat closely by *Very concerned* (N = 20, 28.99%). When looking at combined numbers of *Concerned* and *Very concerned* selections, this topic ranks #1 of those listed in the survey, while also containing the lowest number of combined *I haven't thought about it* and *Not concerned at all* ratings. These results suggest that, among the topics we presented to the students, this is the area they are most concerned about.
5. **Students not developing creative problem-solving and critical-thinking skills:** The most common level of concern by our participants regarding students not developing problem-solving and critical-thinking skills was *Very concerned* (N = 22, 31.88%), followed closely by *Somewhat concerned* (N = 20, 28.99%). Though only ranking third in combined *Concerned* and *Very concerned* rankings, this topic held the highest selections of *Very concerned* among

all topics listed in our survey. This suggests that even though this may not be the highest average area of concern for students, it may be a more polarizing one, where students are largely either very concerned or only mildly concerned.

6. **Legal or ethical issues:** The most common level of concern selected by our participants regarding legal or ethical issues for AI tools was *Somewhat concerned* (N = 27, 39.13%). The two lowest levels of concern, interestingly, were *Concerned* (N = 7, 10.14%) and *Very concerned* (N = 9, 13.04%). Additionally, this topic held the highest number of *I haven't thought about it* ratings (N = 15, 21.74%). These results suggest that, among the topics we presented to the students in our survey, this was the area they held the lowest levels of concern, and have spent the least amount of time discussing or thinking about the topic.

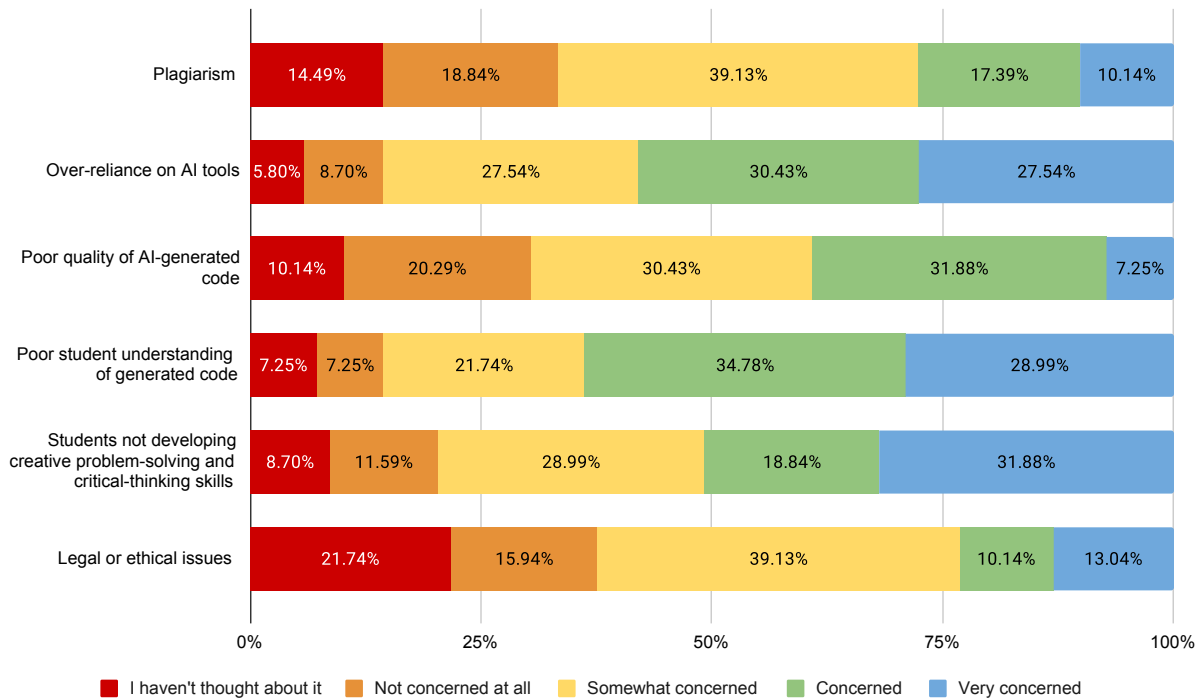


Figure 4.17: Student levels of concern across a variety of topics

Our results for Question #30 suggest that students are primarily concerned with not understanding AI generated code, and end up building too much reliance on AI code generation tools. On the opposite end of the spectrum, students are the least concerned about legal or ethical issues and plagiarism, with these topics having the lowest numbers of ratings for high levels of concern, and the highest numbers of ratings for having not thought about the topics.

Question #31 is formatted similar to Question #30, instead asking students to highlight their

levels of optimism regarding potential outcomes of AI code generation tool usage. Once again, we analyze the results for each topic:

1. **Students being better prepared for future jobs:** The most common rating for this topic among our participants was *Somewhat optimistic* (N = 27, 39.13%). Interestingly, this topic resulted in the lowest number of *Very optimistic* ratings (N = 6, 8.70%) and the highest number of *Not optimistic at all* ratings (N = 11, 15.94%) among all our topics for Question #31. This suggests that students are generally *not* optimistic that using AI code generation tools will help prepare them for their future careers.
2. **Improvements to equity and access in programming:** The most common rating for this topic among our participants was *Optimistic* (N = 25, 36.23%), followed closely by *Somewhat optimistic* (N = 22, 31.88%). This topic resulted in the highest number of *Optimistic* ratings, and tied for the second most *Very optimistic* ratings, suggesting that this is an area students are among the most optimistic about.
3. **Students gaining self-facilitated learning from personalized feedback:** The most common rating for this topic among our participants was *Optimistic* (N = 21, 30.43%), followed closely by both *Somewhat optimistic* and *Very optimistic* (N = 19, 27.54% each). This topic also resulted in the lowest number of *I haven't thought about it* ratings (N = 6, 8.70%). With the lowest number of ratings for low levels of optimism, and the highest number of ratings for high levels of optimism, our results suggest that this is the area students are most optimistic about regarding their usage of AI code generation tools.
4. **Students improving on CS theory and problem-solving skills:** The most common rating for this topic among our participants was *Somewhat optimistic* (N = 29, 42.03%). This topic resulted in the lowest number of *Optimistic* ratings (N = 14, 20.29%) and the second lowest number of *Very optimistic* ratings (N = 10, 14.49%) among our topics listed. When considering this with the results of Question #30's topic on problem-solving and critical-thinking skills, these results confirm that students are generally not optimistic that AI code generation tools will help them develop necessary programming skills.
5. **Students becoming more interested in learning:** The most common rating for this topic among our participants was *Somewhat optimistic* (N = 25, 36.23%). This topic holds the largest number of *I haven't thought about it* ratings (N = 12, 17.39%) among our topics. With relatively moderate ratings for all other levels of optimism, our results suggest that this is a topic students may not have strong feelings about, or perhaps they feel AI tools have had little effect towards their interest in Computer Science fields.

Our results for Question #31 suggest that students are most optimistic about using AI code generation tools to help learn on their own and gather personalized feedback whenever needed. The

high levels of optimism around equity and access also suggests students may feel that these tools help level the playing field, where less competent programmers can better keep up with their more accomplished peers. However, students also displayed low levels of optimism towards these tools preparing them for their careers and helping develop their programming skills, indicating they may feel that these tools help them perform better in their courses, but not in the manner they desire (e.g., better understanding of the material).

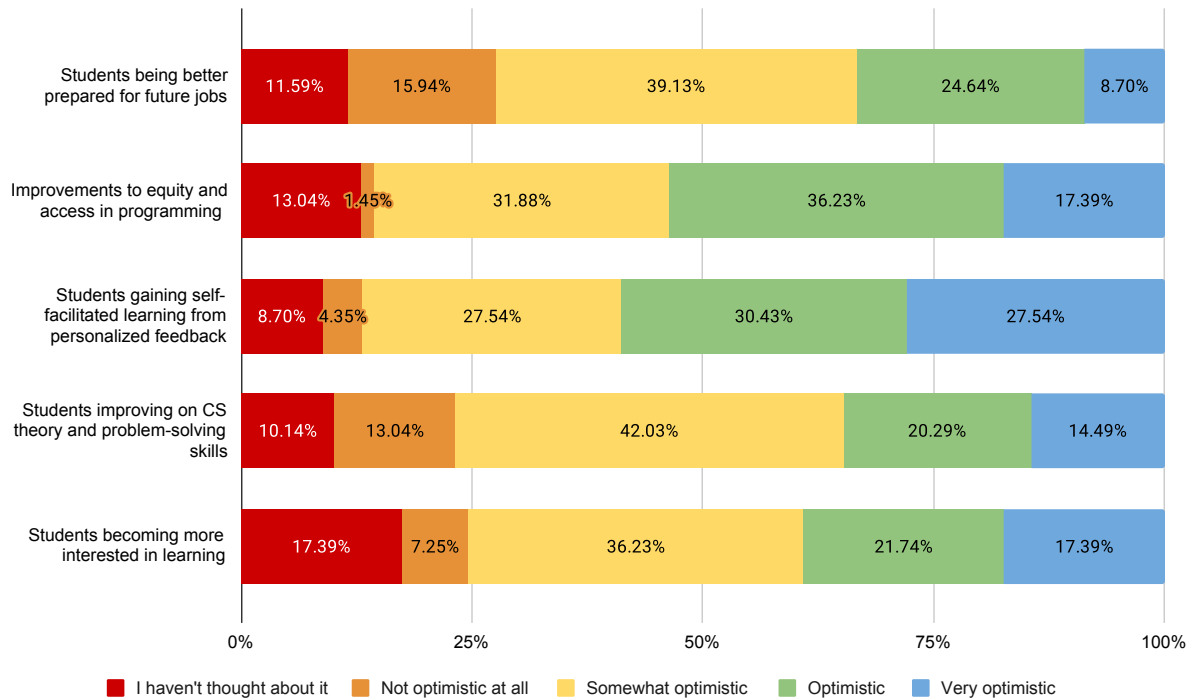


Figure 4.18: Student levels of optimism across a variety of topics

Summary for RQ4. Our data suggests that students largely see AI code generation tools as a skill set they need to develop to further their careers. However, students have mixed feelings when weighing the benefits and consequences of using AI code generation tools. Students are most optimistic towards gaining self-facilitated learning and personalized feedback from AI tools, yet they also showcase high levels of concerns towards over-reliance, failing to develop necessary skills, and not understanding the code generated by these tools. Thus, they show relatively low levels of optimism that these tools will actually prepare them for their future careers.

4.6 RQ5: How do students interact with AI code generation tools?

To address this RQ, we analyzed the responses to Questions #27 and #28 from our online questionnaire, as well as the submitted prompt logs from students who used ChatGPT during the programming activity. Questions #27 and #28 asked participants what they use AI code generation tools for, and were only presented to students who did *not* select *Never* to Question #23, which asked them how often they use AI code generation tools for school-related activities. Based on this requirement, we received 64 responses to Questions #27 and #28.

4.6.1 Questionnaire Results

Question #27 asks participants to enter their single most frequent reason for using the tools, while Question #28 takes that response, and by default lists it as the top ranking item in a list of use-cases for AI code generation tools, where the participant can then reorder the use cases by how likely they are to use AI code generation tools for that purpose. For this ranking, rank 1 depicts the most common use-case, while rank 6 depicts the least common use-case. Here we will analyze some of the free-text answers for Question #27, followed by the participant rankings for AI code generation tool use-cases.

To analyze the results of Question #27, we use a similar categorization for what was used to analyze Question #18, which asked users to explain why they used ChatGPT in the activity. However, since Question #27 is not related to an activity with time or resource restrictions, we remove the *Activity Restrictions* category to be left with the following:

- **Coding Assistance:** Responses under this category expressed use-cases related to productivity, identifying errors, and improving code quality. For the responses that mentioned productivity in some fashion, they primarily focused on AI code generation tools being a faster alternative to other resources available (e.g., *“To get quick answers that might take a little longer to find when searching using a search engine.”*). Most responses in this category stated that AI code generation tools were used primarily when the student becomes stuck, does not know how to proceed, or for assistance with debugging (e.g., *“Last resort for when I’m stuck but the lectures slides aren’t helpful enough and I don’t have anyone else I can ask at the moment.”*). We received 34 responses (53.13%) under this category.
- **Knowledge Constraints:** Responses under this category described using AI code generation tools to explain various programming concepts (e.g., *“Sometimes, I do not understand when instructors explain during lecture. Therefore, I typically use it to teach myself or ask to explain it in simpler terms.”*), or to explain what certain pieces of code do (e.g., *“To help explain concepts and why a block of code works a certain way instead of what I expected”*). We received 21 responses (32.81%) under this category.

Table 4.7: Average rankings of how students use AI code generation tools. A lower numbered ranking signifies a high number of students listing it as a top reason for using the tool.

Use-case for AI code generation tools	Average ranking of participants (out of 6)
I utilize AI code generation tools to explain programming concepts to me with provided examples	3.44
I utilize AI code generation tools to debug and troubleshoot my code	3.53
I utilize AI code generation tools to bridge the gap between theoretical knowledge and practical implementation	3.73
I utilize AI code generation tools to increase my programming productivity	3.94
I utilize AI code generation tools to improve the quality of my code	5.20

- **Familiarity:** Responses under this category stated reasons relating to ease of use (e.g., “*easy to finish your job.*”), convenience, or very generically described using AI code generation tools for help (e.g., “*Help with homework*”). These responses did not provide any other specific reasoning for using AI code generation tools. We received 9 responses (14.06%) under this category.

For Question #28, participants were asked to rank their reasons for why they utilize AI code generation tools, using their response from Question #27 as the default rank 1 item. Participants’ rank 1 item largely remained the same, with only 5 students (7.81%) electing to place another use-case above their original reasoning. Given the varying nature of this response, we instead focus on the average rankings for the other listed use-cases. Of those, the use-case of “*I utilize AI code generation tools to explain programming concepts to me with provided examples*” ranked the highest, with an average ranking of 3.44 out of 6. The lowest ranking use-case of “*I utilize AI code generation tools to improve the quality of my code*” averaged a score of 5.20 out of 6. Table 4.7 lists these rankings in descending order, from the highest ranking to the lowest.

4.6.2 ChatGPT Interactions

According to our questionnaire results, 50 out of our 70 participants (71.43%) used ChatGPT during the programming activity part of our study. To understand how these students interact with ChatGPT, we directed students to submit their discussion log alongside their code repository. Due to the nature of some batch studies performed, we were not able to confirm the correct submission of every participant in our study, resulting in 32 out of our 50 students (64.00%) submitting their ChatGPT discussion log. Among these 32 students, 3 students submitted improper file formats, and could not be properly viewed or analyzed. Our analysis focuses on the 29 correctly submitted ChatGPT discussion logs we received and their contents.

For each discussion log examined, we searched for unique behaviors across three aspects of their

dialogue:

1. **Priming:** This aspect looks at how students initiate their conversation with ChatGPT, including the level of detail they provide from the start in regards to their task and relevant code, as well as any unique verbiage or instructions they include to set the intended behavior of the AI's responses.
2. **Prompt engineering:** This aspect looks at how students react to ChatGPT's results, including adjustments they make to their own prompts in an effort to guide the AI tool in a better direction, or if they make reference to segments of ChatGPT's reply.
3. **General discourse:** This aspect attempts to define any more generalized behaviors present throughout the discussion log, particularly in the absence of any priming or prompt engineering behaviors.

After analyzing all 29 available ChatGPT logs, we identified a number of unique behaviors regarding how students interacted with the AI. Here we provide a list of these behaviors, along with a brief description, and sample prompts to match each behavior. Though some students showcased multiple behaviors in their logs, the majority are represented by only a single behavior.

- **Focus on one error at a time:** The most common behavior observed in our study was the student attempting to understand and correct their code by focusing on singular errors at a time. These students would provide context to the AI with some source code, generally starting with small segments of code and increasing to entire classes or multiple classes depending on the success of ChatGPT's response. The style of their prompts differ slightly, with the majority of the prompts including a code snippet and some direction on what they want the AI to do with the code provided (e.g., *"Could you help me determine what is the issue in this code?"*, *"I am receiving the NullPointerException, here's the entire code:"*). This behavior generally includes no priming of the AI prior to debugging, which may result in the AI being unaware of additional classes and dependencies related to the issue. Occasionally these students would engaged in some manner of prompt engineering. Some students replied directly to the content of ChatGPT's output (e.g., *"I like the second method, how may I go about doing that?"* and *"I'm confused, can you further explain what the issue is and how you are solving it?"*). Others would repeat a similar prompt, but provide additional source code for the AI to make sense of the issue. On rare occasion, a student would reply to ChatGPT, but with poor context, leading to a misunderstanding of intent (e.g., *"How do I implement the classes correctly"*). In these cases, ChatGPT would provide a more generic answer that did not align with the greater needs of the activity, leading students towards compounding issues.

- **Fixing the code all at once with no priming:** A small number of students in our study engaged with ChatGPT by providing large pieces of the source code at a time with only brief instructions (e.g., *“Could you please check this code for me?”* or *“Fix this”*). In reply to these prompts, ChatGPT would identify multiple issues and offer multiple solutions at once. Students would repeat this behavior for all files or classes, until the AI has seen the entire source code. On rare occasion, students would provide the source code for all classes in the first prompt, allowing ChatGPT to identify dependencies between classes and provide more suitable solutions. For students who provided classes one at a time, ChatGPT would sometimes offer solutions that don’t integrate with other classes properly, leading to unresolved errors. In these instances, students would often repeat the process of providing the entire source code of the class, but specify the error that they are still encountering.
- **Fixing the code all at once with priming:** A very small number of students demonstrated behavior suggesting extensive experience at having used ChatGPT before, and understanding how to get the most out of the AI. These students provided detailed priming and prompt engineering during their interactions. Students began by explaining the assignment to ChatGPT (e.g., *“I am trying to find errors in a java program that runs as a quiz function and then assigns a grade based on how many questions were answered correctly.”* and *“This program aims to output a quiz that accepts user input and displays their total score in the form of 4 java files: AbstractQuiz.java, App.java, MathQuiz.java, Question.java.”*). Students would also immediately provide all relevant source code for the activity, in this case all 4 Java files, to give the AI full context of its starting point. With this setup, the students relied on ChatGPT to perform the majority of the debugging process, using its recommended code snippets and corrective actions. Students would then continue to interact with the AI using natural language similar to how they may converse with a peer or mentor (e.g., *“Can you simply describe the inner workings how the program works?”*). In one instance, the student admitted to their own confusion, asking the AI to help them get back on track (e.g., *“I am lost in my assignment, I need to calibrate and understand what is going on, I will reprovide my code, can you assist in explaining it to me what it does, how it ties into each other and then what is wrong with it.”*). The way in which ChatGPT responded to the student’s priming resulted in different behaviors by the student. In one student’s case, ChatGPT provided corrections to singular classes at a time, including code snippets for the student to copy and paste. These code snippets resolved all issues in the code, and only required the smallest levels of intervention by the student. In other cases, ChatGPT would give suggestions for multiple classes in one response, in which case it would not provide code snippets for everything, leading to increased interactions with the student and some apparent trial and error in correcting the code.
- **Copying and pasting without any context:** This was the least common behavior ob-

served in our study. These students would copy and paste the source code and/or error messages to the ChatGPT prompt, and submit, with no additional context or direction provided. The amount of source code would slightly differ, with the majority of interactions being limited to small portions of a single class or the entirety of a class by itself. While ChatGPT does a reasonable job at interpreting the student’s intent, it would sometimes offer more generalized solutions to the student, particularly in the absence of necessary source code such as superclasses. Because of this, students would eventually alter their behavior slightly, either by providing brief instructions (e.g., “*Error with i*” or “*It did not correctly calculate the total score*”) or additional source code, such as multiple classes in a single prompt.

- **Obtaining explanations for core concepts:** This behavior is represented by students whose prompts were primarily composed of questions or concepts they wanted ChatGPT to explain to them. Students who engaged in this behavior did not include any form of priming or prompt engineering (e.g., “*Declaring class variable syntax for Java*”, and “*What is the format to write classes in Java?*”). Furthermore, they did not include any of their source code in their prompts, though they would occasionally submit an error message by itself. Only a small number of students prioritized this behavior, while the majority would only occasionally submit such prompts. For those who prioritized this behavior, it is unclear if they did so intentionally, with a greater emphasis on learning rather than completing the assignment, or if they were simply unaware of ChatGPT’s capabilities to accept source code and provide more specified feedback.
- **Repeating the same prompts:** On rare occasion, the student would encounter a series of prompts where ChatGPT would provide a solution that did nothing to solve their error, and the student would continue to tell the AI that its solution didn’t work (e.g., “*It works now but it doesn’t print the question, it just prints [...]*”, “*Still only prints this*”, and “*Still doesn’t say the question*”). In these instances, the AI appears to lack the necessary information to identify the problem or implement the correct solution, and the student does not know how to continue forward. Instead of attempting a new prompt or adding more source code for reference, the student seems to assume the AI will backtrack further, or provide guidance to the student on what it needs to help further. In our study, this phenomena occurred when the error the student was attempting to resolve was occurring at a line in a class separate from the root cause. For instance, their error message pointed to line 50 in class A, but the reason for the issue occurring was due to the design in class B. The student would then focus on class A, while never providing class B’s code to assist the AI.

The students within our study displayed a wide variety of behaviors when interacting with ChatGPT, with some students providing significant context to the AI by describing the objective and including all of their source code, and others doing the complete opposite and only asking for

explanations of programming principles or Java syntax. The most common behavior we observed was the tendency to focus on correcting singular errors at a time, by providing only the necessary source code and a description of the error. The least common behavior observed was students simply copying and pasting the source code without explanation. These outcomes show some promise that students are still focused on learning the material, despite the potential for AI code generation tools to simply do the work for them.

Summary for RQ5. According to our questionnaire results, students use AI code generation tools primarily as a personalized tutor, providing explanations and sample code for concepts or issues they have questions on. Following that, they use AI code generation tools when they are stuck or having difficulty with an assignment or activity. During our study, students that used ChatGPT during the assigned programming activity displayed a wide variety of behaviors, with the most common emphasizing correcting one error at a time with minimal context. Furthermore, the majority of students submit prompts that are more in line with understanding the material as they go, rather than simply completing the assignment. Students generally did not utilize any form of AI priming or prompt engineering, though this may be intentional behavior depending on the student's motivation behind using ChatGPT, and how large of a role the student desired to play in completing the assignment.

CHAPTER 5

DISCUSSION

AI code generation tools are a powerful resource with potential to significantly shape the future of both the software development industry and computer science education alike. Thus, our study aims to provide insight on how students perceive and interact with these tools to better understand the effect these tools have on their education.

By and large, ChatGPT is the AI code generation tool that students are the most familiar with and use the most often, according to our findings in RQ1 and RQ3. Given the popularity of ChatGPT among the general population, this is not necessarily surprising, though the lack of familiarity with alternative tools may indicate that students are not well-versed with this AI code generation technology as a whole. This can be concerning, as our findings in RQ1, RQ2, and RQ3 all point to the fact that students are using these tools regularly, more-so than traditional resources used in the past for learning, such as Stack Overflow, online documentation, or even tutorial videos. At the same time, these findings show some promise, as students report they use ChatGPT mostly for explanations and examples, rather than to have the AI write code for them, the latter appearing to be the concern for the majority of educators [14, 16, 29, 32]. Our RQ4 results demonstrate some level of student concern that over-reliance on these tools will hinder their education. At the same time, they believe AI code generation tools will be necessary for their future careers, which match the findings of similar studies on student perspectives [1, 14, 27, 32].

While some studies have examined in-depth how students interact with other AI tools [26], there's a lack of research in how students interact with ChatGPT specifically. This knowledge is crucial given the dominant usage of this AI tool, as indicated by our previous RQs. This paper sheds new light into the behaviors of students as they engage with ChatGPT, while providing supporting evidence for the results of our questionnaire and others in the intentions students have for using AI tools. More explicitly, students appear to use ChatGPT to improve their own comprehension of programming material, as opposed to having the AI simply do the work for them. However, our findings indicate that students may not have a firm understanding of how to use ChatGPT to meet their needs, as demonstrated by a lack of context given in their prompts. These observations lead us to our main takeaways and suggestions for the various stakeholders in this study.

Students

Students are wise to be cautious about over-reliance on AI code generation tools and the potential negative effects on their learning. However, they are correct in recognizing that familiarity with these tools may be necessary for their future careers. Therefore, we recommend that students experiment with using AI code generation tools for various purposes. Students already use these

tools to ask questions and generate examples, which is beneficial, and they should continue doing so while remaining aware of the possibility of AI hallucinations. They should also practice prompt engineering to determine which tasks AI can successfully generate working code snippets for and where it struggles. When uncertain about how they are permitted to use AI code generation tools in their courses and assignments, students should communicate with their instructors to ensure that learning objectives are appropriately met.

Educators

Educators should be aware that nearly every student is familiar with ChatGPT, and some are experimenting with other AI code generation tools available on the market. Additionally, students report using these tools for more than half of their schoolwork. Given their desire to succeed in their courses, students are likely to continue relying on AI tools if they believe these tools are beneficial. Therefore, educators should clearly communicate their policies and beliefs regarding AI usage in the classroom, considering students' priorities with empathy. For those who discourage the use of AI tools, it is important to explain the value of mastering course material without AI assistance and its impact on future careers and coursework. Our RQ4 findings indicate that students are concerned about the long-term consequences of using these tools, so a gentle reminder and explanation may be sufficient to reduce reliance on them. However, educators should remain open to the idea that there may be areas where the use of AI tools does not significantly impact learning objectives.

For educators who support the use of AI tools, the focus should be on helping students understand how these tools work, their strengths and weaknesses, and how to use them effectively. Our RQ3 findings suggest that students are familiar with these tools and use them regularly. However, our RQ5 data demonstrates their use is often limited in scope. Educators should provide examples of effective versus ineffective prompts for various tasks and demonstrate how students can use priming and prompt engineering to tailor the AI to their needs. Some examples of techniques instructors could introduce are defining a role for the AI to fulfill and specifying the format of the AI output. These techniques will help prepare students for their future courses and career.

Industry

Employers should recognize the degree to which their new hires may depend on generative AI technology and adapt their interview processes and on-boarding programs accordingly. For employers who regularly use AI code generation tools within their departments, understanding which tools are employed, their specific applications, and any specialized techniques for their use is highly valuable to academic circles. For example, developers might primarily use AI tools for documentation or writing test cases rather than for debugging or explaining concepts. These are use cases not commonly taught to students, potentially leading to gaps in the skills necessary for their careers. While developers are not expected to broadcast their daily activities and tool usage, we encourage

them to participate in research aimed at providing insights into these practices. For developers collaborating closely with academia, leveraging AI tools can create valuable programs and systems to support both students and educators [9, 17, 21, 22]. Engaging in discussions with educators to explore the potential applications of this technology can further the development of tools that advance Computer Science education.

CHAPTER 6

THREATS TO VALIDITY

The findings of our study may not accurately reflect the situation with students across the globe given our method of recruiting, the location of our study, and the number of participants we were able to analyze. Over half of our participants were within their freshman (1st) year, thus our findings may not hold true for students belonging to academic years that were less represented in our survey.

By design of our study, students may have been more inclined to discuss ChatGPT and show their familiarity with it. Given the limited resources available during the programming activity, we acknowledge that students may have elected to use ChatGPT for assistance, even if they would look for alternative means on a normal basis. Within the programming activity, students were tasked with debugging and completing pre-written Java code, which may result in observed behaviors with ChatGPT not representing the typical use-case for students. Additionally, due to the timed nature of the programming activity, students may have felt pressured to prioritize expediency, which may differ from their behavior when working on assignments and assessments with more generous timelines. A number of participants were also unable to correctly submit their code repository and ChatGPT log, resulting in a loss of data that may have been necessary to accurately represent the population of our study.

Though we designed our questionnaire to be as easily understood and free from bias as possible, students may have still interpreted questions and concepts differently, affecting their ratings and responses. Though we provided *Other* options for many of our questions, the absence of specific items, such as AI code generation tools, may have prevented students from remembering or discussing them. Lastly, participants were excused from our study upon completion of the questionnaire, which may have encouraged students to provide less detailed responses in an effort to expedite their completion.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This study explores the various experiences and perceptions of programming students with AI code generation tools, providing valuable insight into the familiarity, reliance, and behaviors students have with them. Our findings show predominant usage of ChatGPT among students, with conflicting beliefs regarding the benefits and detriments AI tools induce. However, students demonstrate a desire to develop good practices with AI code generation tools, and do not wish to become overly reliant on them. As the capabilities of AI code generation tools continue to grow, further studies should ensure best practices are maintained to address the many concerns held by those in the software engineering field.

Future work in this regard includes longitudinal studies examining the long-term outcomes of AI tool usage, in-depth studies of students interacting with these AI code generation tools for additional use-cases, studies with software developers in the industry to understand how students should tailor their use of AI code generation tools, and studies with instructors to identify positive behaviors and lesson plans to accommodate this technology in the classroom. Lastly, it should be noticed that we live in a time where the capabilities of AI tools and systems are rapidly changing, and with them are societal standards and perspectives towards AI tools. These factors may significantly affect the results of any future studies in this field of research.

APPENDIX A

ACTIVITY INSTRUCTIONS

Activity Instructions

For this experiment, you will be given some pre-built Java code aimed at creating a console-based quiz application. Your objective is to ensure that the code runs per the requirements described below. You may complete this activity on your own, or with the use of ChatGPT, but no other resources will be allowed.

About the Code

The Java code provided is the basis for a console-based quiz application. The questions and answers are hard-coded into the files and should remain hard-coded.

In total, there are 10 single-choice questions. When the program is run, each question is shown individually, and prompts the user for their answer.

```
Question #2: What is 8 - 3?  
A) 11  
B) 5  
C) 24  
D) -1  
Please enter your answer: █
```

Once the last question has been answered by the user, the results of the quiz should be displayed.

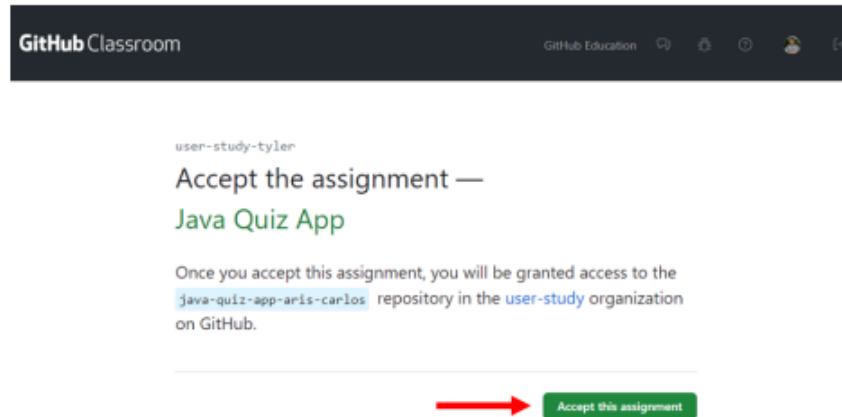
```
Question #10: What is -9 / -3?  
A) -27  
B) -3  
C) 3  
D) 27  
Please enter your answer: C  
  
-----  
Your score: A | 100.00 % | 50/50 points
```

Steps

1. Log into your GitHub account.
2. Click the following link to access the code:
https://classroom.github.com/a/Ut_PoMHp

Figure A.1: Page #1 of the instructions provided to students at the start of the programming activity.

3. Click the **Accept this assignment** button.



4. Give the repository a few moments to copy to your account before refreshing the page.
5. Once the repository is copied, you should see the below view. Click the **Open in GitHub Codespaces** button.

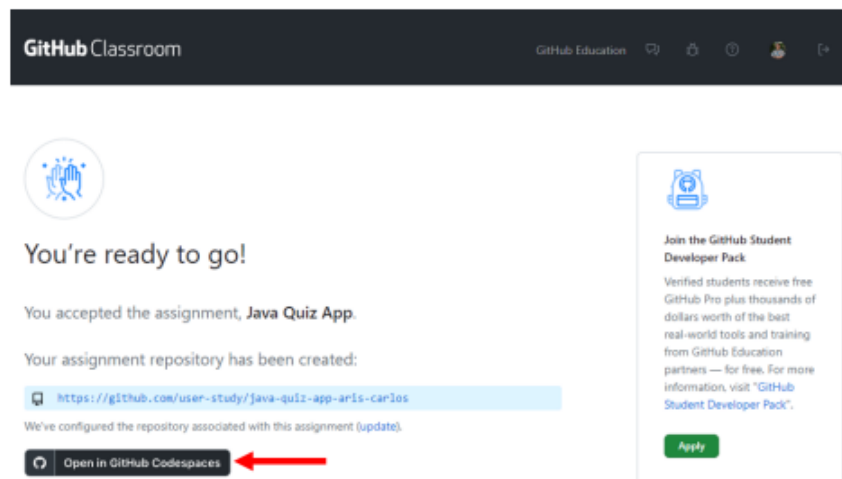
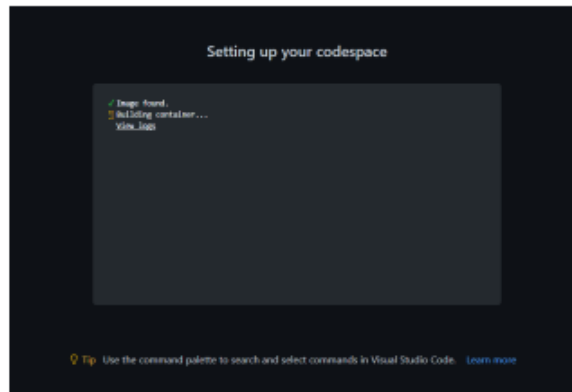


Figure A.2: Page #2 of the instructions provided to students at the start of the programming activity.

6. Give it a few moments to set up your Codespace. Please do not exit this page while it is loading.



7. Once the setup is complete, locate the required files inside of the folder "src/main/java"

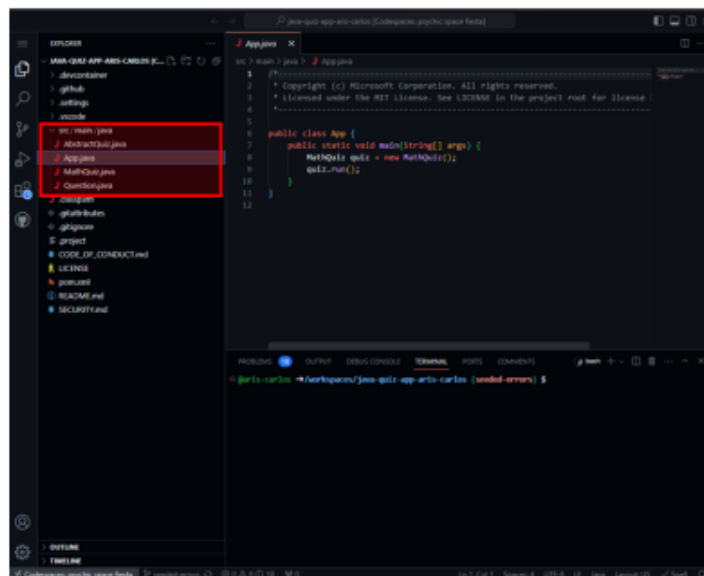


Figure A.3: Page #3 of the instructions provided to students at the start of the programming activity.

8. Find and correct the files as needed to meet the activity objective.

Please feel free to use ChatGPT to assist you in this task. All other outside resources are prohibited for this activity.

When you are done

When the time for the experiment has ended, or you have successfully met the activity objective, please raise your hand to have a proctor assist you. You will be asked to commit and push all changes you made to the Java code, as well as include your ChatGPT log if it was used.

Once your commit has been pushed to GitHub, please continue past this page to complete our survey.

Thank you!

Figure A.4: Page #4 of the instructions provided to students at the start of the programming activity.

APPENDIX B

SOURCE CODE WITH ERRORS

```
1  /*-----*/
2  * Copyright (c) Microsoft Corporation. All rights reserved.
3  * Licensed under the MIT License. See LICENSE in the project root for license information.
4  *-----*/
5
6  public class App {
7      public static void main(String[] args) {
8
9          // Initialize the appropriate quiz type and execute it
10         AbstractQuiz quiz = new MathQuiz();
11         quiz.run();
12     }
13 }
```

Figure B.1: Source code of App.java. This was the driving class for the application, and did not contain any seeded errors. File available at <https://github.com/tylera211/thesis/blob/main/App.java>

```
1
2
3 public class Question {
4
5     // Class variables
6     int points;
7     String question;
8     String[] answers;
9     char correctAnswer;
10    char selectedAnswer;
11
12    public Question(int points, String question, String[] answers, char correctAnswer) {
13        points = points;
14        question = question;
15        answers = answers;
16        correctAnswer = correctAnswer;
17    }
18
19 }
```

Figure B.2: Source code of Question.java. Class variable assignments within the constructor use incorrect references. File available at <https://github.com/tylera211/thesis/blob/main/Question.java>

```

1
2
3 import java.util.Scanner;
4
5 public abstract class AbstractQuiz {
6
7     // Class variables
8     Question[] a;
9     int c;
10
11     // Returns a letter grade based on score (out of 100)
12     public char grade(double score) {
13         if(score > 90) return 'A';
14         if(score > 80) return 'B';
15         if(score > 70) return 'C';
16         if(score > 60) return 'D';
17         return 'F';
18     }
19 }
20
21 // Executes the console-based quiz by presenting the questions one at a time, listing the possible answers,
22 // waiting for and accepting the user input, and displaying their score at the end
23 public void run() {
24     Scanner input = new Scanner(System.in);
25     int userScore = 0;
26
27     for(int i = 0; i < a.length; i++) {
28         System.out.printf("Question #d: ", i+1);
29         System.out.println(a[i].question);
30         for(int j = 0; j < a[i].answers.length; j++) {
31             System.out.printf("%s) %s\n", (char) ('A' + j), a[i].answers[j]);
32         }
33         System.out.print("Please enter your answer: ");
34
35         System.out.println();
36     }
37
38     System.out.println("-----");
39     double score = ((double) userScore / c) * 100;
40     System.out.printf("\nYour score: ");
41
42     input.close();
43 }
44
45 }

```

Figure B.3: Source code of AbstractQuiz.java. Seeded+ errors include an incorrectly placed closing bracket '}', an extra semicolon ';', poor naming conventions for class variables, and missing code calculating and displaying the user score. File available at <https://github.com/tylera211/thesis/blob/main/AbstractQuiz.java>

```

1 public class MathQuiz extends AbstractQuiz {
2
3     public MathQuiz() {
4
5         // Total # of possible points
6         c = 50;
7
8         // Temporary variables to define Question components before passing them to the Question constructor
9         // and assigning the Question to the quiz's array of questions
10        int points; String question; String[] answers; char correctAnswer;
11
12        // Question 0
13        points = 5;
14        question = "What is 2 + 5?";
15        answers = new String[] {"25", "10", "7", "-3"};
16        correctAnswer = 'C';
17        this.questions[0] = new Question(points, question, answers, correctAnswer);
18
19        // Question 1
20        points = 5;
21        question = "What is 8 - 3?";
22        answers = new String[] {"11", "5", "24", "-1"};
23        correctAnswer = 'B';
24        this.questions[1] = new Question(points, question, answers, correctAnswer);
25
26        // Question 2
27        points = -5;
28        question = "What is 2 * 9?";
29        answers = new String[] {"11", "16", "18", "81"};
30        correctAnswer = 'C';
31        this.questions[2] = new Question(points, question, answers, correctAnswer);
32
33        // Question 3
34        points = 5;
35        question = "What is 10 / 5?";
36        answers = new String[] {"2", "5", "7.5", "10"};
37        correctAnswer = 'A';
38        this.questions[3] = new Question(points, question, answers, correctAnswer);

```

Figure B.4: Source code of MathQuiz.java, part 1. Seeded errors include a mismatch of class variable names from its superclass, AbstractQuiz.java, an uninitialized array prior to index assignments, an incorrect points assignment of -5, and an incorrect correctAnswer assignment of 'c' instead of 'C'. File available at <https://github.com/tylora211/thesis/blob/main/MathQuiz.java>

```

39
40     // Question 4
41     points = 5;
42     question = "True or false: 9 + 9 = 99";
43     answers = new String[] {"true", "false"};
44     correctAnswer = 'B';
45     this.questions[4] = new Question(points, question, answers, correctAnswer);
46
47     // Question 5
48     points = 5;
49     question = "True or false: 9 + 9 = 18";
50     answers = new String[] {"true", "false"};
51     correctAnswer = 'A';
52     this.questions[5] = new Question(points, question, answers, correctAnswer);
53
54     // Question 6
55     points = 5;
56     question = "What is 2*2*2?";
57     answers = new String[] {"4", "6", "8", "10"};
58     correctAnswer = 'C';
59     this.questions[6] = new Question(points, question, answers, correctAnswer);
60
61     // Question 7
62     points = 5;
63     question = "What is 3*3?";
64     answers = new String[] {"6", "8", "9", "12"};
65     correctAnswer = 'C';
66     this.questions[7] = new Question(points, question, answers, correctAnswer);
67
68     // Question 8
69     points = 5;
70     question = "What is 5 * -3?";
71     answers = new String[] {"-15", "-2", "2", "8"};
72     correctAnswer = 'A';
73     this.questions[8] = new Question(points, question, answers, correctAnswer);
74
75     // Question 9
76     points = 5;
77     question = "What is 9 / 3?";
78     answers = new String[] {"-27", "-3", "3", "27"};
79     correctAnswer = 'c';
80     this.questions[9] = new Question(points, question, answers, correctAnswer);
81
82
83 }
84 }

```

Figure B.5: Source code of MathQuiz.java, part 2. Seeded errors include a mismatch of class variable names from its superclass, AbstractQuiz.java, and an uninitialized array prior to index assignments. File available at <https://github.com/tylera211/thesis/blob/main/MathQuiz.java>

BIBLIOGRAPHY

- [1] Matin Amoozadeh, David Daniels, Daye Nam, Aayush Kumar, Stella Chen, Michael Hilton, Sruti Srinivasa Ragavan, and Mohammad Amin Alipour. Trust in Generative AI Among Students: An Exploratory Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 67–73, 2024.
- [2] Sebastian Baltes and Paul Ralph. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering*, 27(4):94, 2022.
- [3] Marco Barenkamp, Jonas Rebstadt, and Oliver Thomas. Applications of AI in classical software engineering. *AI Perspectives*, 2(1):1, 2020.
- [4] Mehul Bhattacharyya, Valerie M Miller, Debjani Bhattacharyya, and Larry E Miller. High Rates of Fabricated and Inaccurate References in ChatGPT-Generated Medical Content. *Cureus*, 15(5), 2023.
- [5] Avijeet Biswal. 24 cutting-edge artificial intelligence applications in 2024. <https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/artificial-intelligence-applications>, Jun 2024.
- [6] Som Biswas. Role of ChatGPT in computer programming. *Mesopotamian Journal of Computer Science*, 2023:9–15, 2023.
- [7] Niccolo Conte. Ranked: The Most Popular AI Tools. <https://www.visualcapitalist.com/ranked-the-most-popular-ai-tools/>, Jan 2024.
- [8] Begum Karaci Deniz, Chandra Gnanasambandam, Martin Harrysson, Alharith Hussin, and Shivam Srivastava. Unleashing developer productivity with generative AI. <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai>.
- [9] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A Becker, and Brent N Reeves. Prompt Problems: A New Programming Exercise for the Generative AI Era. *arXiv preprint arXiv:2311.05943*, 2023.
- [10] Robin Emsley. ChatGPT: These are not Hallucinations—They’re Fabrications and Falsifications. *Schizophrenia*, 9(1):52, 2023.
- [11] FlexOS. Generative AI Top 150: The World’s Most Used AI Tools (Feb 2024). <https://www.flexos.work/learn/generative-ai-top-150>, Feb 2024.

- [12] Katherine Haan. 24 top AI statistics and Trends in 2024. <https://www.forbes.com/advisor/business/ai-statistics/>, Jun 2024.
- [13] Ishika Joshi, Ritvik Budhiraja, Harshal Dev, Jahnvi Kadia, M Osama Ataullah, Sayan Mitra, Dhruv Kumar, and Harshal D Akolekar. Chatgpt in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions. *arXiv preprint arXiv:2304.14993*, 2023.
- [14] Ishika Joshi, Ritvik Budhiraja, Pranav Deepak Tanna, Lovenya Jain, Mihika Deshpande, Arjun Srivastava, Srinivas Rallapalli, Harshal D Akolekar, Jagat Sesh Challa, and Dhruv Kumar. “with Great Power Comes Great Responsibility!”: Student and Instructor Perspectives on the influence of LLMs on Undergraduate Engineering Education. *arXiv e-prints*, pages arXiv–2309, 2023.
- [15] Eirini Kalliamvakou. Research: Quantifying GitHub Copilot’s Impact on Developer Productivity and Happiness. <https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/>, Sep 2022.
- [16] Sam Lau and Philip Guo. From “Ban It Till We Understand It” to “Resistance is Futile”: How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*, pages 106–121, 2023.
- [17] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J Malan. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 750–756, 2024.
- [18] Kamil Malinka, Martin Peresíni, Anton Firc, Ondrej Hujnák, and Filip Janus. On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree? In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, pages 47–53, 2023.
- [19] Bernard Marr. The 20 Generative AI Coding Tools Every Programmer Should Know About. <https://www.forbes.com/sites/bernardmarr/2024/05/23/the-20-generative-ai-coding-tools-every-programmer-should-know-about/>, May 2024.
- [20] John McCarthy et al. *What is artificial intelligence*. Stanford University, 2007.
- [21] Atharva Mehta, Nipun Gupta, Dhruv Kumar, Pankaj Jalote, et al. Can ChatGPT Play the Role of a Teaching Assistant in an Introductory Programming Course? *arXiv preprint arXiv:2312.07343*, 2023.

- [22] Ha Nguyen and Vicki Allan. Using GPT-4 to Provide Tiered, Formative Code Feedback. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 958–964, 2024.
- [23] Oluwatimilehin Ogidan. Top 10 AI Tools for Developers in 2024. <https://code.pieces.app/blog/top-10-ai-tools-for-developers>, Feb 2024.
- [24] OpenAI. Introducing ChatGPT. <https://openai.com/index/chatgpt/>, Nov 2022.
- [25] Tracy Phillips. AI Code Tools: The Ultimate Guide in 2024. <https://codesubmit.io/blog/ai-code-tools/>, Jul 2024.
- [26] James Prather, Brent N Reeves, Paul Denny, Brett A Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. “it’s Weird That it Knows What I Want”: Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, 31(1):1–31, 2023.
- [27] Michael P Rogers, Hannah Miller Hillberg, and Christopher L Groves. Attitudes Towards the Use (and Misuse) of ChatGPT: A Preliminary Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 1147–1153, 2024.
- [28] Jaromir Savelka, Arav Agarwal, Christopher Bogart, Yifan Song, and Majd Sakr. Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?(2023). *arXiv preprint arXiv:2303.09325*, 2023.
- [29] Judy Sheard, Paul Denny, Arto Hellas, Juho Leinonen, Lauri Malmi, and Simon. Instructor Perceptions of AI Code Generation Tools-A Multi-Institutional Interview Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 1223–1229, 2024.
- [30] Alex Singla, Alexander Sukharevsky, Lareina Yee, and Michael Chui. The state of AI in early 2024: Gen AI adoption spikes and starts to generate value — mckinsey.com. <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>, May 2024.
- [31] Nicole Willing. Top 10 Most Popular AI Tools That You Need to Use in 2024. <https://www.techopedia.com/top-10-most-popular-ai-tools>, June 2024.
- [32] Cynthia Zastudil, Magdalena Rogalska, Christine Kapp, Jennifer Vaughn, and Stephen MacNeil. Generative AI in Computing Education: Perspectives of Students and Instructors. In *2023 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2023.