Science of Computer Programming 235 (2024) 103111

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

"Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools

Mohammad Amin Kuhail [a,*], Sujith Samuel Mathew [a], Ashraf Khalil [b], Jose Berengueres [c], Syed Jawad Hussain Shah [d]

[a] *College of Interdisciplinary Studies, Zayed University, Abu Dhabi, UAE*
[b] *College of Technological Innovations, Zayed University, Abu Dhabi, UAE*
[c] *College of Information Technology, UAE University, Al Ain, UAE*
[d] *School of Science and Engineering, University of Missouri, Kansas City, USA*

ARTICLE INFO

Keywords:
ChatGPT
Programmer assistant tools
AI tools
Chatbot

ABSTRACT

ChatGPT is a language model with artificial intelligence (AI) capabilities that has found utility across various sectors. Given its impact, we conducted two empirical studies to assess the potential and limitations of ChatGPT and other AI tools in software development. In the first study, we evaluated ChatGPT 3.5′s effectiveness in generating code for 180 coding problems from LeetCode, an online coding interview preparation platform. Our findings suggest that ChatGPT 3.5 is more effective in solving easy and medium coding problems but less reliable for harder problems. Further, ChatGPT 3.5 is somewhat more effective at coding problems with higher popularity scores. In the second study, we administered a questionnaire ($N = 99$) to programmers to gain insights into their views on ChatGPT and other AI tools. Our findings indicate that programmers use AI tools for various tasks, such as generating boilerplate code, explaining complex code, and conducting research. AI tools also help programmers to become more productive by creating better-performing, shorter, and more readable code, among other benefits. However, AI tools can sometimes misunderstand requirements and generate erroneous code. While most programmers are not currently concerned about AI tools replacing them, they are apprehensive about what the future may hold. Our research has also revealed associations between AI tool usage, trust, perceived productivity, and job security threats caused by the tools.

1. Introduction

Programmer assistant tools have revolutionized the software development landscape as they can automate testing and debugging [1,2], analyze [3], optimize [4], and generate code [5]. All these tools have contributed to software development by automating monotonous activities and providing suggestions to improve programmers' productivity and efficiency.

Artificial Intelligence (AI) tools such as OpenAI Codex [6] and ChatGPT (2022) have emerged, ushering in a new era of AI programmer assistant tools. The performance of these tools is promising. For example, Codex, which has 12 billion model parameters and is trained on 54 million GitHub software repositories, solved over 70% of 164 Python programming problems with 100 examples [7].

* Corresponding author.
E-mail addresses: mohammad.kuhail@zu.ac.ae (M.A. Kuhail), sujith.mathew@zu.ac.ae (S.S. Mathew), ashraf.khalil@zu.ac.ae (A. Khalil), jose@uaeu.ac.ae (J. Berengueres), shs6g7@umsystem.edu (S.J.H. Shah).

https://doi.org/10.1016/j.scico.2024.103111
Received 6 August 2023; Received in revised form 25 February 2024; Accepted 22 March 2024
Available online 22 March 2024

ChatGPT, in particular, has had a massive impact across various domains. With 100 million active visitors in its first two months of availability, ChatGPT has become the fastest-growing AI tool in history [8]. ChatGPT is a type of chatbot, a conversational agent that emulates human-like conversations. ChatGPT has proliferated since its debut due to its applications in various domains, including software development [9], marketing [10], and healthcare [11], amongst others.

Recent research suggests that AI tools, like ChatGPT, can provide code recommendations for programmers [12] and solve coding problems [13]. The effectiveness of AI tools in solving coding problems appears to be influenced by the complexity of the problems. For example, Nascimento et al. [14] demonstrated that AI tools can produce solutions that outperform those created by novice programmers in the case of simple and moderately complex coding problems. Similarly, Nikolaidis et al. [13] reported that AI tools effectively solve easy coding problems. However, it is also plausible that the availability of solutions to these coding problems and their popularity may influence the effectiveness of AI tools. Past studies have suggested that the accuracy of AI tools, such as ChatGPT, can be influenced by the training data they are fed [15].

The effectiveness of AI tools for coding problems is not the only crucial aspect in determining their impact on the software industry. Equally significant is programmers' perception of these tools. Understanding this perception can provide important insights into using AI tools in the industry. Research has begun to emerge on the perceptions and attitudes of programmers towards AI tools. Among the limited research available, a qualitative study conducted by Ge & Wu [16] indicates that the utility of the tools influences the adoption of ChatGPT. Another study by Liang and Myers [17] assessed programmers' perceptions of AI tools (such as Copilot) and found that the tools have limitations in addressing certain functional and non-functional requirements and that programmers have little control over the output generated. Similar findings were reported by Ciniselli et al. [18]. AI tools often generate erroneous code that is difficult to understand or does not conform to project coding standards [19]. These studies provide valuable insights into how AI tools are perceived by programmers, highlighting both their strengths and limitations.

Beyond the strengths and limitations of AI tools, other aspects of programmers' perceptions of AI tools involve trust and perceived productivity. These factors are essential to understanding AI tools' impact on the software industry. A recent study has emphasized the need to understand better whether programmers trust these tools [20]. According to a qualitative study, programmers cite productivity as a reason for trusting AI tools [21]. However, while some studies have reported hopes that AI tools can increase productivity [22,23], others have raised concerns about job security [24]. In short, the existing literature sheds light on programmers' perspectives towards AI tools. Nevertheless, the research on this topic remains scant, and further investigation is required to comprehend the factors that impact programmers' viewpoints towards the tools and their interrelationships.

Given the rise in AI tools, their influence on the software industry, and the limited research on the topic, we conducted this study to evaluate the capabilities, limitations, and perceptions of AI tools to understand their impact on the software industry. In this paper, we present the results of two studies. The first study assessed ChatGPT 3.5′s code generation effectiveness for 180 selected coding problems by LeetCode [25], a popular coding interview preparation tool. We analyzed the results of this study to answer the question, "*How effective is ChatGPT 3.5 for coding problems of varying complexity and popularity?*" (RQ1). We then conducted the second study, where we surveyed 99 professionals with programming backgrounds to gather their views on AI tools, including use cases, trust, perceived productivity, and concerns regarding potential job security threats. We analyzed the results of the second study to answer the question, "*What are the programmers' perceptions of ChatGPT and other AI tools concerning the capabilities, limitations, trust, perceived productivity, job security threats, and the future of AI tools and computer science education?*" (RQ2).

Our study contributes to the software engineering community by extensively evaluating ChatGPT's effectiveness in solving coding problems of different levels of complexity and popularity. Additionally, we present a quantitative and qualitative survey-based study that provides insights into the perceptions of programmers regarding ChatGPT and other AI tools, which can help improve the user experience and adoption of these tools. Our findings offer valuable information to researchers and professionals in software engineering, paving the way for further advancements in AI-assisted coding.

2. Background

2.1. Chatbots

Chatbots are becoming increasingly popular in various fields, including education [26], consumer services [27], and healthcare [28], as they can mimic human conversations and automate services. In 2019, the global chatbot market was predicted to reach 1.2 billion dollars in 2024 [29]. In 2022, ChatGPT's owner, OpenAI, was alone projected to reach a revenue of 1 billion dollars in 2024 [30].

The growth of chatbots has not been sudden; they are almost 50 years old. ELIZA, Weizenbaum's first chatbot (1966), mimicked a psychotherapist talking to a patient using pattern matching. Mid-1990s chatbot ALICE employed knowledge records and Artificial Intelligence Markup Language (AIML) to find a correct answer to user input [31]. SmarterChild [32] was an early-2000s chatbot that used artificial intelligence to communicate with users on messaging platforms. SmarterChild predates current virtual chatbot-based assistants such as Alexa [33] and Siri [34], which are accessible on messaging apps and may imitate conversations with rapid access to data services. Virtual chatbot-based assistants are typically limited to specific commands and are not intended to engage in back-and-forth dialogue. The next section covers the new generation of chatbots powered by generative AI.

2.2. Quick history of chatGPT

In the realm of natural language processing (NLP), the development of generative pre-trained transformer (GPT) language models has driven significant progress. The GPT architecture, originally introduced by Goodfellow et al. [35], has since evolved, with the transformer model playing a crucial role in capturing long-range dependencies between words in the input [36]. Following the inception of the first-generation GPT model, GPT-2 emerged with minor enhancements and an expanded training dataset of 40 GB. These refinements allowed GPT-2 to generate increasingly coherent and realistic output [37]. The subsequent GPT-3 model garnered considerably more attention, primarily due to its massive training dataset, which was 100 times larger (45 TB) than its predecessor. Consequently, GPT-3 can generate human-like, coherent sequences spanning words, code, and other data types [38]. The wide-ranging potential applications of GPT-3 have been the subject of recent scholarly investigations, encompassing fields as diverse as medicine and computer science [39].

2.3. Programmer assistant tools

Machine learning-based program synthesis approaches have been developed to generate computer programs according to specified functionality [40,41]. These techniques utilize input-output examples [42,43], partial implementations [44,45], or natural language descriptions [44,46,47] as input for the target program. Genetic programming methods, such as linear [48], stack-based [49], and grammar-guided [50] approaches, are frequently explored in program synthesis research. Machine learning has also been employed to assist programmers, with extensive research conducted in areas such as program error correction [1,2], program correctness evaluation [3], variable type inference [51], and test script generation [52]. These advancements have significantly shaped the landscape of programmer assistant tools.

Recently, AI-driven programmer assistant tools, or simply AI tools, have gained widespread availability due to the emergence of industrial AI solutions, including OpenAI Codex [6], DeepMind AlphaCode [53], and Amazon CodeWhisperer [54], among others. OpenAI Codex, a derivative of GPT, is an AI model designed by OpenAI to generate code based on natural language. Serving as the foundation for GitHub Copilot [55], an integrated development environment tool for auto-completion assistance, Codex is trained on nearly 50 million GitHub projects, encompassing the majority of GitHub Python code and totaling 159 GB [56]. Codex can generate Python code from English questions, translate code across programming languages, provide English explanations of input code, report time complexity, and develop API-based code for tasks such as email drafting and database access. Finnie-Ansley et al. [57] conducted a noteworthy empirical evaluation of Codex, particularly in the context of beginner programming problems. Their research was pioneering in assessing the accuracy of Codex when applied to CS1-level (introductory computer science course) problems. The methodology involved 23 programming questions that comprised summative assessments from two invigilated lab-based tests, presenting the problem statements to Codex exactly as they were presented to students. Furthermore, they explored the applicability of Codex to multiple variations of the classic Rainfall Problem. With all executions conducted on the "Davinci" Codex model at a temperature setting of 0.9, the study aimed to elicit creative and diverse solutions from Codex. To evaluate performance, the researchers submitted Codex-generated solutions to the same testing software utilized in student examinations. If necessary, as per their 'repeated sampling' approach, Codex was prompted up to ten times for each question before deeming a problem unsolved. This approach of reattempting up to ten solutions aligns with the abandonment threshold used by the developers of Codex in their testing, albeit with a lower threshold. In certain instances where minor formatting issues were the only defects in otherwise correct responses, these were manually corrected, underlining the practical reality of how students typically address such minor submission errors. This study found that Codex outperformed about 80% of the students from these courses on summative test questions, revealing a powerful demonstration of its potential as a programming aid for beginners.

AlphaCode, introduced in 2022, employs a transformer-based model to generate computer programs at a competitive level. Trained on 715 GB of GitHub code in languages such as Python, JavaScript, C++, and C#, AlphaCode has successfully tackled novel challenges requiring critical thinking, logic, algorithmic design, coding, and natural language comprehension [58].

3. Related work

3.1. Effectiveness of AI tools

Several studies have evaluated the effectiveness of AI tools, offering a diverse range of insights into their capabilities and limitations. Some studies focused on assessing ChatGPT's ability to solve coding problems. For instance, Nascimento et al. [14] offered a unique perspective by using recent LeetCode contest problems to test ChatGPT's capabilities. They found that ChatGPT's solutions were more efficient than those of novice programmers for easy and medium-level problems. They also showed better memory efficiency than experienced and novice programmers. Similarly, Nikolaidis et al. [13] explored the efficacy of ChatGPT and Copilot using 50 random coding problems from LeetCode. This study highlighted that ChatGPT was effective for easier problems but was susceptible to syntax and semantic errors.

Other studies focused on assessing AI tools' capabilities by comparing them to human performance. For instance, Koubaa et al. [59] focused on comparing human programmers with ChatGPT in IEEExtreme programming competitions. They found that humans maintained an edge in some problem-solving respects, though their study did not delve into the types of problems used or the efficiency of the solutions. Lertbanjongngam et al. [60] examined whether AI-generated codes, specifically those from AlphaCode, resembled human codes and how they performed in comparison. Their findings showed significant similarity between AI and human codes, with AI's performance being on par or slightly inferior in execution time and memory usage. In contrast, Imai [61] compared human pair programming with programming alongside GitHub Copilot, revealing that while Copilot increased code quantity, it fell short in quality compared to human pair programming.

AI tools based on Large Language Models (LLMs) have inherent limitations. For example, AI tools like ChatGPT have token limitations, restricting the information that can be processed simultaneously [62]. These limitations lead to the truncation or premature ending of the generated code. This code truncation can, therefore, lead to functional or logical errors. These restrictions occur due to limited computational capabilities like memory capacity and processing power. The issue is often mitigated by breaking down complex and large tasks into smaller tasks and getting the LLM to solve these tasks. It is also important to note that the non-determinism of ChatGPT is due to the lack of control of several hyperparameters like temperature, nucleus sampling, max tokens, and other model settings that OpenAI provides. However, while these hyperparameters can be controlled, it is often difficult to find the right balance between these parameters to generate high-quality results. With LLM models like Claude 2, Llama 2, and PaLM that provide better performances and capabilities, we can be assured that these models will soon provide better quality results and code.

Despite the existence of studies that evaluate the effectiveness of AI tools, several shortcomings have been identified. First, these studies lack details on the coding problems used to evaluate the effectiveness of the AI tools. Such details include problem complexity, types of problems, data structures used, and problem popularity on coding platforms. Additionally, the studies on AI tool effectiveness do not include statistical analysis of variables such as problem complexity and popularity, which could influence the success rate of AI tools. Therefore, given the limited number of studies analyzing the effectiveness of AI tools to solve coding problems and the gaps in the current studies, we propose conducting a study to investigate the effectiveness of ChatGPT. Our study addresses the identified gaps and explores the effectiveness of ChatGPT in depth. Furthermore, the study analyzes the influence of variables such as problem complexity and popularity on the success rate of ChatGPT. The study also compares the efficiency of the tool's output with output generated by human code.

3.2. *Programmers' perceptions of AI tools*

The literature on the adoption and perceptions of AI tools in software development presents a multifaceted view of how these technologies influence the software industry. Ge & Wu [16] conducted 50 semi-structured interviews to examine the factors influencing the adoption of ChatGPT for bug fixing in software development. Their study highlights that trust and utility are crucial for adopting these tools. Other studies have reported comparable results regarding the perceived utility and trust of AI tools, contributing to users' intention to use them [63,64]. Wang et al. [21] interviewed 17 developers about the utilitarian reasons behind trusting AI code-generation tools. The study revealed a mixed response: while some developers trust these tools and find them helpful in increasing productivity and improving code quality, others prefer to use them for more mundane tasks, like writing unit tests or generating boilerplate code, but not for complex tasks involving business logic. Stavridis & Drugge [65] broadened the scope by examining AI tools' perceived utility and productivity across various levels of the Software Development Life Cycle (SDLC). They concluded that AI tools could enhance resource efficiency in time, thereby increasing the value derived from software development projects. However, they also noted that while AI tools can produce fast and functional code, human understanding of the generated code is essential for maintaining quality and identifying issues.

Liang & Myers [17] further explored programmers' perceptions of AI tools by assessing the perceived usability of AI tools among 410 developers. They found that developers are motivated to use AI programming assistants primarily because it increases productivity by reducing keystrokes, expediting task completion, and assisting in recalling syntax. However, they also identified limitations, such as the inability of these tools to address certain functional or non-functional requirements and difficulties in controlling the tool to generate desired outputs. Lastly, Zhan et al. [66] surveyed 717 participants, not limited to programmers, to assess concerns about job replacement and existential threats from AI tools. They found that using AI technology can reduce the fear of bias and job replacement. Another study found that implementing AI tools has been observed to positively impact the workforce, as it tends to restructure tasks, leading employees to focus more on creative and social tasks. This shift in focus can enhance productivity without the risk of job replacement [67].

In short, a limited number of studies were conducted on the perceptions of programmers regarding AI tools and their impact on the software industry. These studies have identified variables of interest, such as utility, limitations, trust, productivity, and job security. The studies show that these variables tend to be interrelated. However, despite their contribution to the body of knowledge, the existing studies are few in number, and some of the variables affecting programmers' perception of AI tools, such as productivity and job security, trust, and job security, require further exploration. Additionally, the existing studies are either qualitative or quantitative and may provide an insufficient picture of the programmers' perceptions of AI tools. To address this gap in research and further contribute to the existing body of knowledge, we propose a study that is both quantitative and qualitative in nature. Our study surveys professional programmers' perceptions of AI tools concerning their power and limitations, as well as the potential threats to trust, productivity, and job security caused by these tools.

4. Research methodology

Our research comprises two studies. The first study involves a quantitative evaluation of ChatGPT's effectiveness in solving coding problems of varying levels of complexity and popularity. This study aims to ascertain the success rate of ChatGPT in solving coding problems and to determine how the complexity and popularity of the programming problems affect the success rate (RQ1). Subsequently, in the second phase, we conduct a survey-based quantitative and qualitative study to assess the perceptions of programmers about AI tools' power and limitations, as well as programmers' trust in AI, perceived productivity, perceived threat to job security, and view of the future of computer science education (RQ2).

4.1. Study 1: evaluation of chatGPT's effectiveness

4.1.1. Design and procedure

We evaluated ChatGPT 3.5 in terms of its effectiveness in solving coding problems. The evaluation process involved three steps: *(1) Problem selection:* We carefully selected 180 coding questions from LeetCode. We chose LeetCode as it is a popular platform for interview preparation for software engineer roles [68] as well as the platform used by other related studies such as [14,69]. Our problem selection spanned various categories of programming questions, such as sets, hash tables, and strings (Table 1). We ensured an equal percentage of easy, medium, and hard problems within each category. LeetCode categorized the complexity levels. We also ensured that there was a representation of problems with various popularity scores. The popularity score represents the number of problems available for that topic on LeetCode (Table 1). AI tools such as ChatGPT have limitations regarding the number of tokens they can process. This limitation could be an issue in a study that aims to evaluate ChatGPT's effectiveness in generating code. However, we chose coding problem descriptions that were short and minimal in their requirements to avoid truncation of the prompt and solution provided by ChatGPT. *(2) Problem submission:* To evaluate the effectiveness of ChatGPT 3.5 in solving the coding problems, each author worked on an equal set of problems. We copied the description of each coding problem by LeetCode and pasted it into ChatGPT 3.5′s web-based interface. We did not change the problem description or add a specific prompt. In return, ChatGPT 3.5 produced a solution to the coding problem using Python 2.0. After that, we submitted the solutions produced by ChatGPT 3.5 to LeetCode's coding problem submission system. Because ChatGPT 3.5 operates in a non-deterministic manner, we made three different submissions at different times by different authors to ChatGPT 3.5 for each coding problem at various times to ensure diversity in the responses. *(3) Data collection:* Afterward, we collected the following data: (a) Did the solution pass all the test cases? (Pass/Fail). (b) How did the performance of the solutions provided by ChatGPT 3.5 compare to the submissions by participants on LeetCode? (c) What was the memory usage of the solutions provided by ChatGPT 3.5 compared to the submissions by participants on LeetCode?

4.1.2. Measurements

We assessed the effectiveness of ChatGPT 3.5 based on three factors: (1) its success rate in solving coding problems, (2) the time performance of its coding solutions compared to human solutions, and (3) the efficiency of memory usage of its coding solutions compared to human solutions.

We measured the success rate of the coding problems provided by ChatGPT 3.5 by collecting the data provided by the LeetCode submission system that indicated whether the solutions provided by ChatGPT 3.5 passed all the required test cases, failed some or all the test cases, or exceeded the allotted time. To address the non-determinism of ChatGPT 3.5 and improve the reliability of our study, we conducted three separate measurements for every coding challenge. Following this, we calculated an aggregated score for each coding problem using a specific criterion: if at least two out of the three attempts successfully met the test case requirements, the aggregated score for that coding problem was marked as a pass. If not, the coding problem was marked as a fail. This approach of multiple measurements and the computation of an aggregated score aligns with the guidelines from a recent study [70] that highlighted the issue of non-determinism in LLMs like ChatGPT 3.5.

We measured the performance of coding solutions by ChatGPT 3.5 by collecting the percentage of human solutions that had a poorer performance than the one by ChatGPT 3.5. Like assessing the success rate, we maintained three individual performance measurements for each coding problem, along with an aggregated score representing the average of these three measurements.

Likewise, we measured the memory usage of the coding solutions by ChatGPT 3.5 by collecting the percentage of human solutions that had poorer memory usage than the one by ChatGPT 3.5. For each coding problem, we recorded three separate memory measurements and calculated an aggregated score by averaging these three measurements, akin to how we determine the success rate.

We measured time performance and memory usage, two key attributes of quality software [71]. It is worth noting that LeetCode has a known issue where the runtime and memory usage of the same code can vary due to the resources available on the virtual platform. However, it is important to mention that we compared the performance of ChatGPT-generated code to human solutions in the next section to reveal the results of these performance comparisons.

4.2. Study 2: evaluation of programmers' perception of AI tools

4.2.1. Design and procedure

The questionnaire was announced on professional networks, and we also contacted our professional contacts. Fig. 1 shows an overview of the questionnaire content.

Table 1
Information about the LeetCode coding problem experiment.

Item	Information
Problem topics	Ordered sets ($N = 15$), hash tables ($N = 15$), greedy algorithms ($N = 15$), matrix ($N = 15$), backtracking ($N = 15$), bit manipulation ($N = 15$), arrays ($N = 15$), tries ($N = 14$), dynamic programming ($N = 11$), recursion ($N = 10$), topological sort ($N = 9$), number theory ($N = 5$), shortest path ($N = 5$), linked lists ($N = 4$), other ($N = 2$).
Problem complexity	Easy ($N = 60$), medium ($N = 60$), hard ($N = 60$)
Problem popularity Scores	1279 ($N = 15$), 595 ($N = 15$), 446 ($N = 15$), 412 ($N = 11$), 278 ($N = 15$), 184 ($N = 15$), 167 ($N = 1$), 152 ($N = 15$,), 70 ($N = 4$), 51 ($N = 15$), 44 ($N = 25$), 30 ($N = 9$), 26 ($N = 5$), 17 ($N = 5$)

First, the participants completed a form explaining the purpose of the project, the confidentiality agreement, and the participant's consent to take part in the questionnaire. Second, the participants filled in demographic questions (age, sex, job, and country of residence). Third, the participants rated their expertise in programming on a 1–5 Likert scale, their domain of expertise, programming languages expertise, and use. Fourth, we asked the participants about the AI tools they use, including ChatGPT, frequency of usage, and how helpful the tools are (on a 1–5 Likert scale). The participants were asked to identify use cases of the tools and how they improve their productivity. Further, the participants rated how much the tools improved their productivity. We also asked the participants to highlight AI tools' limitations and their future predictions for their evolution. The participants rated how AI tools influenced their job security as programmers today and in the future. Finally, the participants shared their thoughts on how aspiring programmers should learn coding in light of these tools. Fifth, we asked the participants to rate their trust in ChatGPT and other AI tools. The participants were asked to rate their perception of whether the tool is competent, effective, provides trustworthy help, and acts in their best interest. All the questionnaire questions can be found in Appendix A.

4.2.2. *Sample*

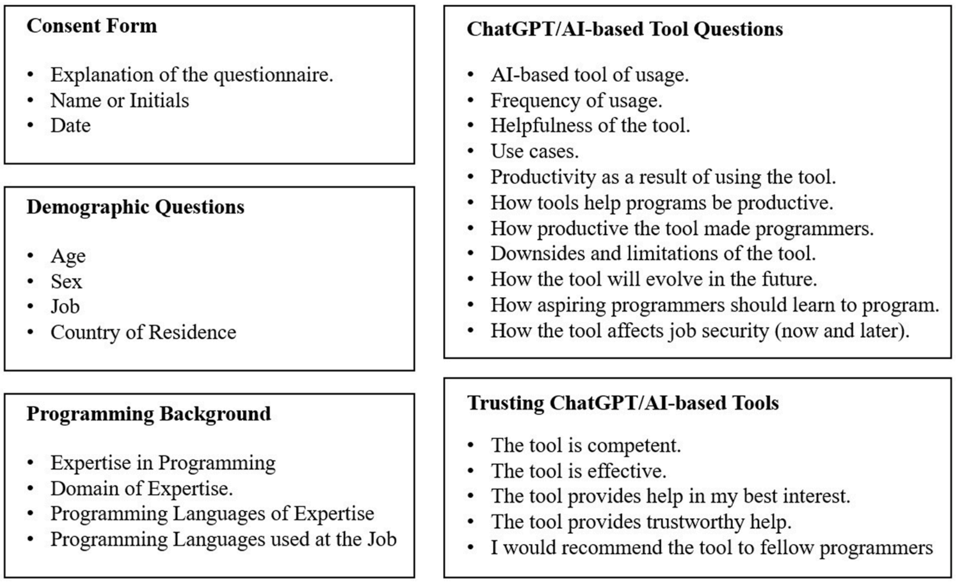
One hundred six professionals with a programming background completed the questionnaire. However, we removed 7 responses due to missing data or simply because they had not used ChatGPT or other AI tools that assist programmers. Table 2 shows participant demographics. Most participants ($N = 47$) were 26–35 years old, whereas 20 participants were 18–25, and 21 were 36–45. The remaining participants were 46–55 ($N = 8$) and 56–65 ($N = 3$). 67% of the participants were male ($N = 67$), and 32% were female ($N = 32$). Concerning country of residence, 47 participants live in the UK, 30 in the US, 11 in the UAE, and 11 elsewhere. Participants reported expertise in backend development ($N = 51$), software engineering ($N = 50$), frontend development ($N = 35$), database administration ($N = 23$), software architecture ($N = 22$), data science ($N = 17$), data engineering ($N = 15$), mobile development ($N = 9$), gaming development ($N = 3$), DevOps engineering ($N = 3$), and others. Developers ($N = 33$), software engineers ($N = 23$), technical managers ($N = 12$), and IT consultants ($N = 6$) were among the participants' current occupations.

4.2.3. *Measurements*

Several items used in the questionnaire were adapted from previously validated scales. We adapted the trust scale from [72,73,74]. Trusting ChatGPT/AI tools was assessed using a Likert scale, which measures three key attributes: integrity, benevolence, and competence. Integrity is measured by the question, *"I feel ChatGPT/AI tools provide trustworthy help,"* while benevolence is measured by the question, *"I feel ChatGPT/AI tools provide help in my best interest."* Competence is evaluated using the questions *"I feel ChatGPT/AI tools are competent in the help they provide"* and *"I feel ChatGPT/AI tools are effective in the help they provide."* Lastly, the question *"I would recommend ChatGPT/AI tools to fellow programmers"* measures the overall level of trust in the tools.

To evaluate productivity, we asked, "In what ways have ChatGPT or other AI tools helped you become more productive?" We then provided a list of various options, such as "completing coding tasks faster," "writing code with fewer errors," "writing more concise code," and "understanding colleagues' code more efficiently." These productivity options are associated with traditional productivity metrics like output, efficiency, and value-added per worker, as outlined by [75]. We also asked the general Likert-scale productivity question, "How much more productive have ChatGPT/AI tools made you as a programmer?".

To measure programmers' sense of job security threat from AI tools, we asked the Likert-scale questions "Do you think ChatGPT/AI tools affect your job security as a programmer now? (reduce your value as a programmer)" and "Do you think ChatGPT/AI tools affect



Consent Form

- Explanation of the questionnaire.
- Name or Initials
- Date

Demographic Questions

- Age
- Sex
- Job
- Country of Residence

Programming Background

- Expertise in Programming
- Domain of Expertise.
- Programming Languages of Expertise
- Programming Languages used at the Job

ChatGPT/AI-based Tool Questions

- AI-based tool of usage.
- Frequency of usage.
- Helpfulness of the tool.
- Use cases.
- Productivity as a result of using the tool.
- How tools help programs be productive.
- How productive the tool made programmers.
- Downsides and limitations of the tool.
- How the tool will evolve in the future.
- How aspiring programmers should learn to program.
- How the tool affects job security (now and later).

Trusting ChatGPT/AI-based Tools

- The tool is competent.
- The tool is effective.
- The tool provides help in my best interest.
- The tool provides trustworthy help.
- I would recommend the tool to fellow programmers

Fig. 1. An overview of the questionnaire content.

Table 2
The demographic information of the participants of study 2.

Characteristic	Description
Age	18–25 ($N = 20$), 26–35 ($N = 47$), 36–45 ($N = 21$), 46–55 ($N = 8$), 56–65 ($N = 3$)
Sex	Male ($N = 67$), female ($N = 32$)
Country of residence	UK ($N = 47$), USA ($N = 30$), UAE ($N = 11$), Other ($N = 11$)
Domain of expertise	Backend development ($N = 51$), software engineering ($N = 50$), frontend development ($N = 35$), database administration ($N = 23$), software architecture ($N = 22$), data science ($N = 17$), data engineering ($N = 15$), mobile development ($N = 9$), gaming development ($N = 3$), DevOps engineering ($N = 3$), other ($N = 8$).
Occupation	Developer ($N = 33$), software engineer ($N = 23$), technical manager ($N = 12$), IT consultant ($N = 6$), DevOps engineer ($N = 4$), professor ($N = 3$), student ($N = 2$), Data scientist ($N = 2$), support manager ($N = 2$), other ($N = 13$).

your job security as a programmer in the future?" These questions are similar to those used in related research studies on perceptions of AI and its impact on employment, such as the one cited in [66].

We asked three open-ended questions to obtain qualitative data. We asked, *"Why do you think ChatGPT/AI tools affect (or not) your job security as a programmer now or in the future?"* to collect qualitative data about the job security threat. The question *"How should aspiring programmers learn coding in light of ChatGPT/AI tools?"* provides perspectives on future computer science education in light of AI tools. At last, the question *"How do you think ChatGPT/AI tools will evolve in the future?"* sheds light on the evolution of AI tools.

5. Results

In the following sections, we present the results of Study 1 and Study 2. The data from both studies can be found in [76].

5.1. Study 1: chatGPT's effectiveness

This section will present the results of testing ChatGPT 3.5′s effectiveness. The results focus on the three factors of effectiveness: success rate, time performance, and memory usage.

5.1.1. Descriptive statistics

Table 3 shows the descriptive statistics of the data obtained from this study. Out of the total 540 measurements taken (3 measurements per coding problem), 57% ($N = 308$) of the problems passed all the test cases, while 39.4% ($N = 213$) failed and 3.5% ($N = 19$) exceeded the set time limit. When looking at the aggregated measurements, out of a total of 180 problems tested, 44.4% ($N = 80$) passed all the test cases, while 51.1% ($N = 92$) failed, and 4.4% ($N = 8$) exceeded the set time limit. Regarding time performance, ChatGPT outperformed human solutions by an average of 58.5% among the passing problems in the total measurements. Regarding the aggregated measurement, ChatGPT outperformed human solutions by an average of 59.1%. Concerning memory usage, ChatGPT used less memory than 69.6% of human solutions on average among the total measurements and less than 69.7% of human solutions on average in the aggregated measurements.

In the next subsections, we will share the impact of problem complexity and popularity on the effectiveness factors (success rate, time performance, and memory usage).

5.1.2. Problem complexity vs. success rate

Fig. 2 shows the relationship between the complexity of problems solved by ChatGPT 3.5 and the success rate. Regarding the total measurements, it was observed that most of the easy problems (86.1%, $N = 155$) passed all the test cases, while a few either failed (12.7%, $N = 23$) or exceeded the set time limit (1.1%, $N = 2$). In contrast, for medium-level problems, around two-thirds of them ($N = 118$) passed all the test cases, while the remaining failed or exceeded the time limit. When it comes to hard problems, only a few of them (19.4%, $N = 35$) passed the test cases, while most of them failed (71.6%, $N = 129$) or exceeded the time limit (8.8%, $N = 16$).

The overall trend in the aggregated measurements is similar, with most easy problems passing (96.6%, $N = 58$) and more than two-thirds of medium-level problems passing (71.6%, $N = 43$). However, only a few hard problems passed (20%, $N = 12$).

We conducted a Chi-square test to assess whether the relationship between problem complexity and the success rate is significant [77]. A chi-square test is a statistical method used to determine if there is a significant difference between categorical variables. We

Table 3
Descriptive statistics of Study 1′s results.

Item	Data
Success rate	Total: Passed ($N = 308$), failed ($N = 213$), exceeded time limits ($N = 19$) Aggregate: Passed ($N = 113$), failed ($N = 67$)
Percentage of ChatGPT solutions with better time performance than human solutions	Total: $N = 308$, μ=0.588, SD= 0.274 Aggregate: $N = 113$, μ=0.591, SD= 0.197
Percentage of ChatGPT solutions with more efficient memory usage than human solutions	Total: $N = 308$, μ=0.696, SD= 0.236 Aggregate: $N = 113$, μ=0.697, SD= 0.149

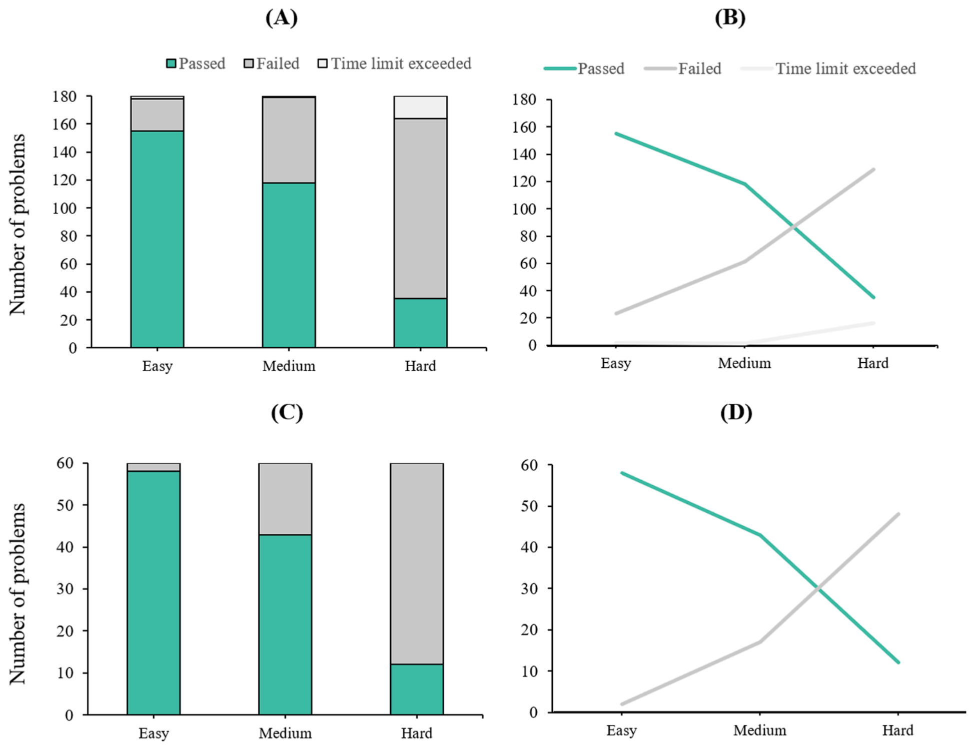


Fig. 2. Problem complexity vs. success rate: Total measurements: (A) stacked chart (B) trend lines. Aggregated measurements: (C) stacked chat (D) trend lines.

used the test to compare the frequencies of pass and fail cases for (1) easy problems, (2) medium problems, (3) and hard problems. We considered the problems that exceeded the time limit as failing problems for simplicity. As a result of the Chi-square test, we report the values of χ^2 (which is a measure that determines how well-observed data fit an expected distribution in categorical data analysis) and p (a value that measures the probability of observing results as extreme as those obtained, assuming the null hypothesis is true. A p-value

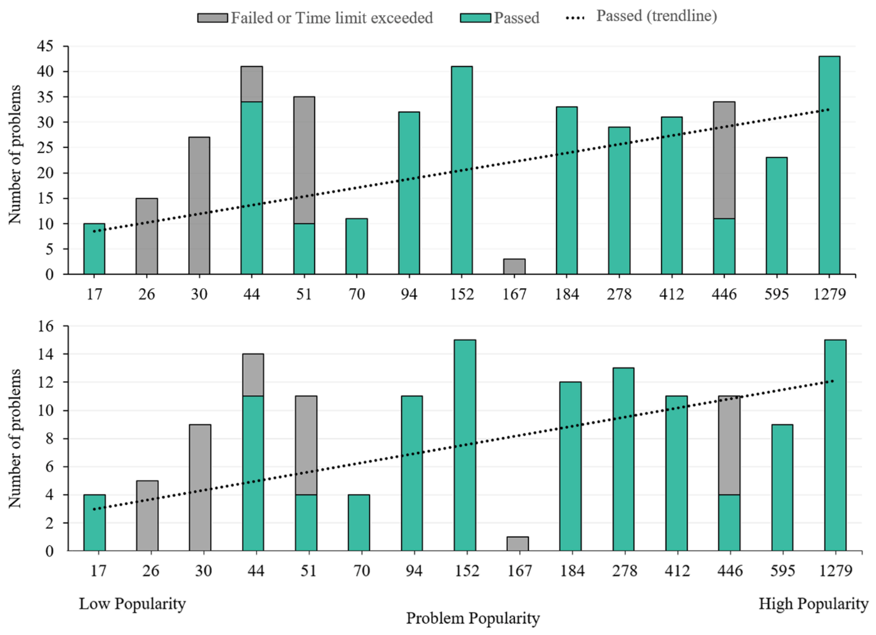


Fig. 3. The relationship between problem popularity and success rate of coding problems solved by ChatGPT 3.5: The total measurements (top), the aggregated measurements (bottom).

less than 0.5 indicates that the observed data is unlikely under the null hypothesis, suggesting a statistically significant effect).

We conducted two Chi-square tests: one for the total measurements ($N = 540$) and another for the aggregated measurements ($N = 180$). For the total measurements, the χ^2 statistic was 171.22, and the p-value was very small (6.57×10^{-38}). We used Cramér's V [78] to test for the effect size. The value was 0.56, indicating a substantial relationship between problem complexity and success rate. For the aggregated measurement, the χ^2 statistic was 78.50, and the p-value was very small (8.97×10^{-18}). The value of Cramér's V was 0.66, indicating a strong relationship between problem complexity and success rate. Both results lead us to reject the null hypothesis that the pass/fail frequencies are independent of the complexity levels of the problems. As such, the data suggests that the probability of passing or failing a test case is significantly associated with the problem's complexity level.

5.1.3. Problem popularity vs. success rate

Fig. 3 shows the relationship between the popularity scores of problems solved by ChatGPT 3.5 and the success rate for the total and aggregated measurements. A Spearman correlation [79] between problem popularity and success rate indicates a moderate correlation that is approaching significance for the total measurements and a moderate and significant correlation for the aggregated measurements (total: correlation=0.486, $p = 0.065$, aggregate: correlation=0.597, $p = 0.018$). Fig. 3 (top) for the total measurements shows a weak trend, showing that as the popularity score increases, so does the success rate. Overall, for problems below or equal to a popularity score of 180, 144 (26.6%) problems failed, and 138 (25.5%) passed. On the other hand, for problems with a popularity score above 100, 170 (31.4%) problems passed, and 88 (16.2%) failed. Fig. 3 (bottom) for the aggregated measurements also shows a similar weak trend, showing that the success rate increases with the increase of popularity score. For problems below or equal to a popularity score of 180, 45 (25%) problems failed, and 49 (27.2%) passed. On the other hand, for problems with a popularity score above 100, 64 (35.5%) problems passed, and 22 (12.2%) failed.

To assess the differences between the two groups (problems with a popularity score <=180 and others with a score >180) for both total and aggregated measurements, we conducted a Mann–Whitney U test [80]. However, the test results indicate no significant differences in the success rates of problems solved by ChatGPT 3.5 ($p > 0.05$) based on their popularity scare for the total and aggregated measurements.

5.1.4. Time performance

We conducted an ANOVA test to assess whether problem complexity or popularity significantly impacted time performance compared to other submissions on LeetCode. This test is a statistical method used to compare the means of three or more independent groups to determine if there is a significant difference among them. The groups here represent time performance percentages categorized based on problem complexity or popularity. In both total and aggregated measurements, the results showed no significant difference in the effect of problem complexity or popularity on time performance compared to other submissions on LeetCode.

5.1.5. Memory usage efficiency

We conducted an ANOVA test to assess if the problem's complexity and popularity significantly impact memory usage compared to other submissions on LeetCode. Based on the results obtained from the analysis, it can be confidently concluded that problem

Table 4
Descriptive statistics of programmers' perceptions of AI tools.

Item	Data
Programming language of expertise	Python ($N = 53$), JavaScript ($N = 37$), Java ($N = 32$), C# ($N = 25$), PHP ($N = 10$), C++ ($N = 9$), C ($N = 7$), Swift ($N = 4$), VBasic ($N = 4$), Bash ($N = 2$), Delphi ($N = 2$), Other ($N = 12$)
Programming language of daily usage	Python ($N = 43$), JavaScript ($N = 32$), Java ($N = 21$), C# ($N = 21$), PHP ($N = 7$), C++ ($N = 4$), Bash ($N = 4$), Swift ($N = 4$), C ($N = 3$), VBasic ($N = 2$), Delphi ($N = 2$), Other ($N = 8$)
Tools used	ChatGPT ($N = 71$), Copilot ($N = 32$), OpenAI Codex ($N = 8$), DeepCode ($N = 6$), Polycoder ($N = 2$), Tabnine ($N = 2$).
Frequency of using AI tools	*Statistics:* μ=2.08, SD=1.09 *Details:* Very rarely ($N = 37$), rarely ($N = 32$), sometimes ($N = 19$), frequently ($N = 7$), very frequently ($N = 4$).
Use cases of AI tools	Generate boilerplate code ($N = 48$), explain code ($N = 38$), research ($N = 36$), find errors in code ($N = 33$), comment code ($N = 31$), write documentation ($N = 21$), improve code performance ($N = 21$), Test cases ($N = 18$), generate regexes ($N = 17$), rewrite code with new conventions ($N = 11$), other ($N = 2$)
Perceived productivity with AI tools	*Statistics:* μ=2.53, SD=1.14 *Details:* Not productive at all ($N = 23$), a little productive ($N = 25$), somewhat productive ($N = 30$), moderately productive ($N = 4$), highly Productive ($N = 5$).
Ways AI tools helped with productivity	Code faster ($N = 58$), better-performing code ($N = 27$), shorter code ($N = 25$), better docs ($N = 24$), easier-to read code ($N = 23$), code conforming to standards ($N = 19$), fewer bugs ($N = 19$), understand colleagues' code faster ($N = 19$), architectural options ($N = 1$)
Issues and limitations with AI tools	Misunderstanding the requirements ($N = 49$), missing the requirement context ($N = 41$), code errors ($N = 38$), code not working for test cases ($N = 29$), inefficient code ($N = 26$), incorrect explanation ($N = 24$), erroneous solutions ($N = 23$), vulnerabilities ($N = 21$), low-quality documentation ($N = 15$), other code issues ($N = 7$), other non-code issues ($N = 3$)
Programmers' perceived job security threats now and in the future	*Statistics (now):* μ=1.89, SD=1.08, *(future):* μ=2.41, SD=1.16 *Details:* High threat (now: $N = 3$; future: $N = 4$), moderate threat (now: $N = 7$, future: $N = 14$), some threat (now: $N = 14$, future: $N = 29$), little threat (now: $N = 28$, future: $N = 24$), no threat (now: $N = 47$, future: $N = 28$)
Trust in AI tools	*Statistics:* μ=3.65, SD=0.75

complexity and popularity hold no significant impact on memory usage compared to other submissions on LeetCode (in both total and aggregated measurements).

5.2. *Study 2: assessment of programmers' perception of AI tools*

5.2.1. *Descriptive statistics*

We ran initial descriptive analyses to understand the data better (Table 4). The participants also reported their expertise and daily usage of various programming languages. Python is the most common language of expertise ($N = 53$) and daily usage ($N = 43$), followed by JavaScript (expertise: $N = 37$, daily usage: $N = 32$) and Java (expertise: $N = 32$, daily usage: $N = 21$). The participants reported using several AI tools to assist them with their jobs. By far, ChatGPT was the most used ($N = 71$, 71.7%), followed by Copilot ($N = 32$, 32.3%) and OpenAI Codex ($N = 8$, 8.1%), among others. Meanwhile, the study participants' average usage frequency was 2.08 on a scale from 1 to 5, where 1 indicates rare usage, and 5 indicates very frequent usage. Most participants, 37.3% ($N = 37$) and 32.3% ($N = 32$), reported using AI tools very rarely and rarely, respectively. A smaller percentage of the participants, that is, 19.1% ($N = 19$), 7% ($N = 7$), and 4% ($N = 4$), reported using the tools sometimes, frequently, and very frequently, respectively.

Participants use ChatGPT and AI tools for a variety of use cases, including the generation of boilerplate code ($N = 48$, 48.5%), explanation of code ($N = 38$, 38.4%), research ($N = 36$, 36.4%), finding errors in code ($N = 33$, 33.3%), commenting on the code ($N = 31$, 31.3%), writing documentation ($N = 21$, 21.2%), improvement of code performance ($N = 21$, 21.2%), generating test cases ($N = 18$, 18.2%), generating regex expressions ($N = 17$, 17.2%), rewriting code with new conventions so that it is understood by other programmers ($N = 11$, 11.1%), and others ($N = 2$, 2%).

When asked about what other ways they use AI tools, a few participants cited that the tools provide them with novel solutions ($N = 3$, 3%), check the quality of their solutions ($N = 3$, 3%), and convert their code syntax from one language to another ($N = 2$, 2%), besides other tasks such as code modification, language learning, and code simplification, among others. From participants' perspectives, AI tools like ChatGPT find their use more during the implementation phase of a project life cycle than other phases.

Based on a scale from 1 to 5 where 1 indicates no productivity at all and 5 indicates high productivity, the average perceived productivity of participants with AI tools was 2.53. A considerable number of participants reported that the tools made them somewhat more productive ($N = 30$, 30.3%), moderately more productive ($N = 17$, 17.2%), and highly more productive ($N = 4$, 4%). However, some participants mentioned that the tools only slightly improved their productivity ($N = 25$, 25.3%) or didn't improve it at all ($N = 23$, 23.2%). Concerning how the tools improved their productivity, the participants stated that the tools helped them code faster ($N = 58$, 58.6%), write better-performing code ($N = 27$, 27.3%), write shorter code ($N = 25$, 25.3%), compile better documentation ($N = 24$, 24.2%), write easier to read code ($N = 23$, 23.2%), and develop code conforming to standards ($N = 19$, 19.1%) and with fewer bugs ($N = 19$, 19.2%). Additionally, the tools helped the participants understand colleagues' code faster ($N = 19$, 19.2%) and identify architectural options ($N = 1$, 1%).

Concerning the limitations of the tools, the participants cited that the tools sometimes produce erroneous ($N = 38$, 38.4%) or inefficient code ($N = 29$, 29.3%). The participants also reported code not working for test cases ($N = 29$, 29.3%), code with security vulnerabilities ($N = 21$, 21.2%), and other code issues ($N = 7$, 7%). Other issues not directly related to code include the misunderstanding of requirements ($N = 49$, 49.5%), missing the requirement context ($N = 41$, 41.4%), erroneous solutions ($N = 23$, 23.2%), low-quality documentation ($N = 15$, 15.2%), and other non-code issues ($N = 3$, 3%).

Concerning the current job security threat caused by the tools currently (Fig. 4), the study found that on a scale of 1 to 5, where 1 meant no threat at all and 5 meant high threat, the average current job security threat among participants was 1.89. While 47.5% ($N = 47$) of the participants didn't view AI tools as a job security threat at all, 28.3% ($N = 28$) saw them as a minor threat. Some participants ($N = 14$, 14.1%) cited AI tools as a potential job security threat, while only 3% saw them as a high threat. However, the future outlook was different, with an average job security threat of 2.41. The numbers showed that participants were more concerned about their job security, with 29.3% feeling some threat, 14.1% moderate, and 4% high.

5.2.2. *Thematic analysis*

We conducted a thematic analysis following the Forbes [81] method to examine the qualitative data provided by the participants on job security threats, the future of computer science education, and the evolution of AI tools. This process involved several steps, including becoming familiar with the data, generating initial codes, searching for themes, defining themes, iteratively reviewing

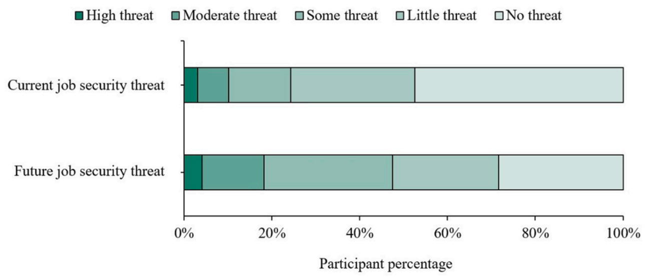


Fig. 4. Participants' perceived current and future job security threats.

themes, and writing up the results.

For participants not concerned about their job security concerning the tools (Table 5), they cited that domain knowledge is required for development ($N = 14$, 14%). Therefore, humans will always have an edge as their experiences are invaluable ($N = 11$, 11.1%). Some participants stated that the tools are only helpful for smaller computing problems but not for larger software projects ($N = 5$, 5.1%). Others mentioned that requirements are individualized and understanding the context requires a human ($N = 3$, 3%). One participant cited that these tools rely on programmers' prior knowledge and experience ($N = 1$, 1%) to submit meaningful queries.

On the other hand, participants concerned about their job security (Table 6) cited that there will be no need for coders in the future ($N = 13$, 13.1%), and those tools are already good at programming ($N = 5$, 5.1%). One participant cited that the tools will be able to write full programs in the future, while another cited that the tools will process the requirements independently.

Concerning the future of computer science education in light of AI tools, some participants did not see a need to change how aspiring programmers should learn how to code (Table 7). In their views, solid foundations are still needed regardless of the tools ($N = 14$, 14.1%), and the tools should be used as a last resort for learners ($N = 1$, 1%).

On the other hand, some participants saw a need to change how programming is learned in light of the tools (Table 8). For example, some participants encouraged teaching the tools ($N = 19$, 19.2%), while others emphasized the teaching of foundations alongside the tools ($N = 13$, 13.1%) as well as problem-solving skills ($N = 4$, 4%). A few participants cited the need to be careful with the tools due to their limitations ($N = 9$, 9.1%).

Concerning how the tools will evolve (Table 9), many participants think AI tools will improve ($N = 23$, 23.2%). In particular, AI tools will be more accurate with answers ($N = 20$, 20.2%), help with productivity ($N = 6$, 6.1%), understand the requirements better ($N = 4$, 4%), answer advanced questions ($N = 3$), and automate more tasks ($N = 3$, 3%). A few participants cited that the tools will become developer assistants ($N = 3$, 3%) and cover more applications in the future ($N = 2$, 2%), among others.

On the other hand, a few participants think there will be negative consequences as the tools evolve (Table 10). For example, some participants ($N = 7$, 7.1%) think the tool will replace entry-level IT jobs. Other participants ($N = 2$, 2%) stated that over-reliance on the tools might cause programmers to lose creativity. Interestingly, one participant thought the tool would never understand requirements sufficiently, while another stated that the tools would reach a point of diminishing returns.

5.2.3. *Correlative analysis and statistical tests*

To understand the variables influencing the programmers' perceptions of AI tools, we conducted various Spearman correlations [79] between variables identified in the questionnaire (Table 11). Noticeably, there are weak but significant positive correlations between the frequency of using AI tools and trusting the tools, between sensing a current threat to job security due to AI tools and trusting the tools, and between productivity associated with AI tools and sensing a current job security threat. Further, a strong correlation exists between sensing a current and a future job security threat.

To understand the relationships between variables, we conducted Mann–Whitney U tests to assess the differences between various groups. We assessed whether the frequency of using AI tools affects the trust and job security threat (Table 12). The results show a significant difference between participants' trust in AI tools based on the frequency of using the tools (participants with a frequency>=3 vs. participants with a frequency<3). We used Cohen's d [82] to test for the effect size. The value was 0.815, indicating a large size effect. Fig. 5 (left) shows that those who use the AI tools more frequently trust the tools more than other participants. However, usage frequency does not appear to significantly influence programmers' feelings of job security threats based on the frequency of using AI tools ($p > 0.05$).

We assessed whether trusting AI tools affects the job security threat as perceived by participants (Table 13). The results show a significant difference between participants' feelings of job security threat based on trusting AI tools (participants with a trust>=3.5 vs. participants with a trust<3.5). The Cohen's d value was 0.974, indicating a large size effect. Fig. 6 (left) shows that those who trust the tools more have a higher sense of job security threat from the tools.

We also found a significant difference between participants' perceived level of job security threat currently and in the future directly due to AI tools. Cohen's d was 0.432, indicating a small to medium size effect. Fig. 6 (right) shows that participants sense higher job security threats from AI tools in the future.

We assessed whether the productivity associated with AI tools affects trust in AI tools (Table 14). The results show a significant difference between participants' trust in AI tools based on the productivity associated with the tools (participants with productivity<=2 vs. participants with productivity>=3). Cohen's d was 0.843, indicating a strong size effect. Fig. 7 (left) shows that participants with higher productivity achieved with AI tools trust the tools more.

Table 5
Reasons why some participants do not sense a job security threat from AI tools.

Theme	Frequencies	Example
Domain knowledge is needed	14	*"Randomly selected person off the street would not be able to do my job even if given access to chatgpt and source code I work with. Too much domain knowledge required to catch up."*
Humans will always have an edge	11	*"I think human validation is still required to ensure code is correct and meets local rules and regulations/standards."*
Only isolated problems	5	*"ChatGPT provides solutions for isolated problems, not a complete project that requires integration."*
Requirements are individualized	3	*"Because the requirements are always individualized depending on the job and context will be needed."*
The tool depends on prior programmers' experience	1	*"I do not think that AI tools will affect my job security, it will actually emphasize it because these tools depend on the existence of programmers that are able to train and maintain them."*

Table 6
Reasons why some participants sense a job security threat from AI tools.

Theme	Frequencies	Example
No need for coders in the future	13	*"In the future, the need will be for software architects and designers, not coders/programmers."*
It is already good at programming	5	*"it is better than the some of the best programmers already"*
Write full programs	1	*"because they will become better and potentially could lead to being able to write entire programs by themselves"*
Process requirements on their own	1	*"it might become smarter in the future to process business requirements on its own"*

Table 7
Reasons for not changing the education of programmers in light of AI tools.

Theme	Frequencies	Example
Solid foundations are still needed	14	*"Foundations are necessary irrespective of any AI tool. every programmer should build solid foundations."*
Last resort	1	*"I would recommend that new programmers use chatGPT as a last resort to verify their code. Having chatGPT generate code will make them way too reliant and in turn there will be a lack of creativity."*

Table 8
Reasons for changing the education of programmers in light of AI tools.

Theme	Frequencies	Example
Teach the tool	19	*"Show them the result ad how it will make their life easier"*
Teach foundations and use the tool	13	*"ChatGPT will always have a defect of not understanding the requirements properly. Regardless of this fancy technology, everyone should learn how to code and add ChatGPT as a good tool to be used whenever needed."*
Be careful with the tools	9	*"These kind of tools should be used carefully; misunderstanding software foundations is at stake, with all the repercussions that may have on code quality"*
Focus on problem solving	4	*"Learning new better ways of solving problems"*

Table 9
Themes for how AI tools will positively evolve in the future.

Theme	Frequencies	Example
The tools will improve.	23	*"They will become smarter"*
More accurate/Fewer errors	20	*"become better at giving correct solutions and also making bigger responses"*
Faster code writing	6	*"I think they will help us become even more productive"*
Better understanding of requirements	4	*"The interpretation side will improve - development end will all depend on the specification inputs "*
Answer advanced questions	3	*"I think that it will become more advanced in applying and learning algorithms"*
Automate more tasks	3	*"They will just do the code"*
Developer assistants	3	*"I believe they will become an "assistant" like entity that will write the majority of code for developers"*
More applications	2	*"in the future, we can expect better responses and wider applications."*

Table 10
Themes for how the tools will negatively evolve in the future.

Theme	Frequencies	Example
Job replacement	7	*"They'll replace many entry jobs for sure, why keep a staff of customer support when an AI can be trained and do it 24/7 basically without pay?"*
Reliance on the tool/Drop in creativity	2	*"Over reliance in it will cause a drop in creativity"*
It will never understand requirements	1	*"I don't think AI can ever understand requirements well enough to do coding required for work"*
Diminishing returns	1	*"it will reach a point of diminishing returns. "*
Legacy code	1	*"on the other hand as the technologies evolve more deprecated libraries or even more complex functions within libraries, the model might mix those versions more often (I faced this issue multiple times) "*

We did not find a significant difference between perceived job security threat levels based on the productivity associated with the tools ($p > 0.05$). Fig. 7 (right) shows that participants with higher productivity achieved with AI tools tend to sense a higher job security threat from the tools.

6. Discussion

Our study offers valuable insights into the potential impact of AI tools, such as ChatGPT, on software programmers. Specifically, we delve into how ChatGPT can effectively assist programmers in streamlining tedious coding tasks and enhancing their productivity.

Table 11
Spearman correlations between the variables identified in Study 2.

Variables	Correlation coefficient	p-value
Usage frequency vs. Trust	0.378	0.000*
Usage frequency vs. Job security threat (current)	0.175	0.082
Job security threat (current) vs. Trust	0.255	0.010*
Job security threat (current) vs. Job security threat (future)	0.752	2.74×10^{-19}*
Productivity vs. Trust	0.442	4.62×10^{-6}*
Productivity vs. Job security threat (current)	0.260	0.009 *
* Significant correlation		

Table 12
Testing the significance of the relationship between trust and usage frequency of AI tools.

Hypothesis	U-value	p-value	Cohen's d
Trusting AI tools significantly differs between participants based on their usage frequency. • Group 1: Trust values for participants who use AI tools highly or moderately frequently (frequency>=3) (N = 30, μ=4.04, SD=0.470) • Group 2: Trust values for participants who use AI tools less frequently (frequency<3) (N = 69, μ=3.467, SD=0.779)	1470.5	0.000*	0.815

* Hypothesis supported.

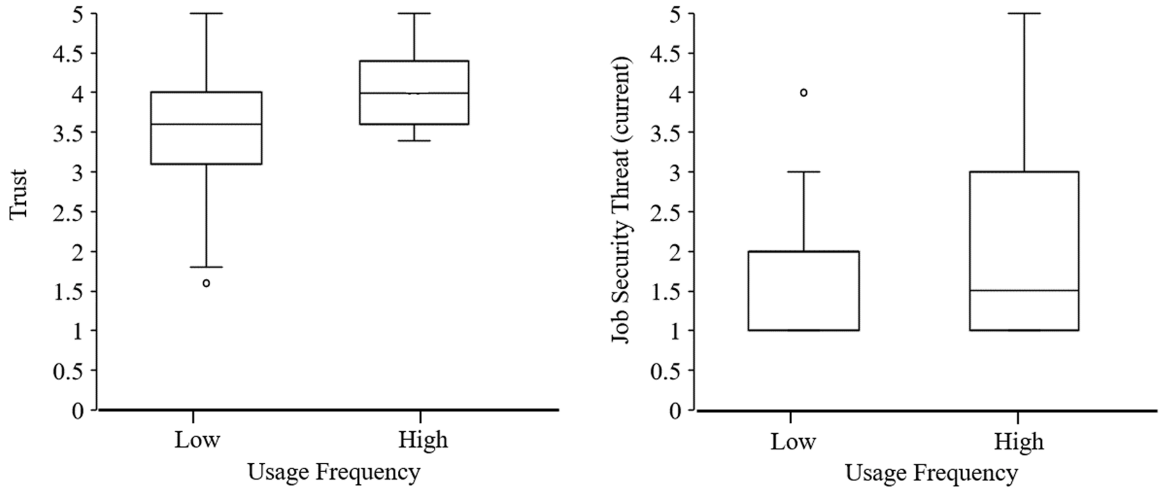


Fig. 5. (left) Box plot showing the trust difference between participants with a higher and lower usage frequency. (right) Box plot showing the perceived job security threat difference between the same two groups.

Table 13
Testing the significance of two relationships: (1) job security and trust of AI tools, (2) job security now and in the future.

Hypothesis	U-value	p-value	Cohen's d
Sensing a job security threat because of AI tools significantly differs between participants based on their trust in the tools. • Group 1: Job security threat values for participants who trust the tools highly or moderately (trust>=3.5) (N = 67, μ=2.11, SD=1.187) • Group 2: Job security threat values for participants who trust tool less (trust<3.5) (N = 32, μ=1.43, SD= 0.618)	1412.0	0.006*	0.974
There is a significant difference between the level of job security threat felt by the participants currently because of the AI tools and in the future. • Group 1: Job security threat values for participants (future) (N = 99, μ= 2.383, SD=1.157) • Group 2: Job security threat values for participants (current) (N = 99, μ=1.898, SD=1.083)	3685.0	0.001*	0.432

*Hypothesis supported.

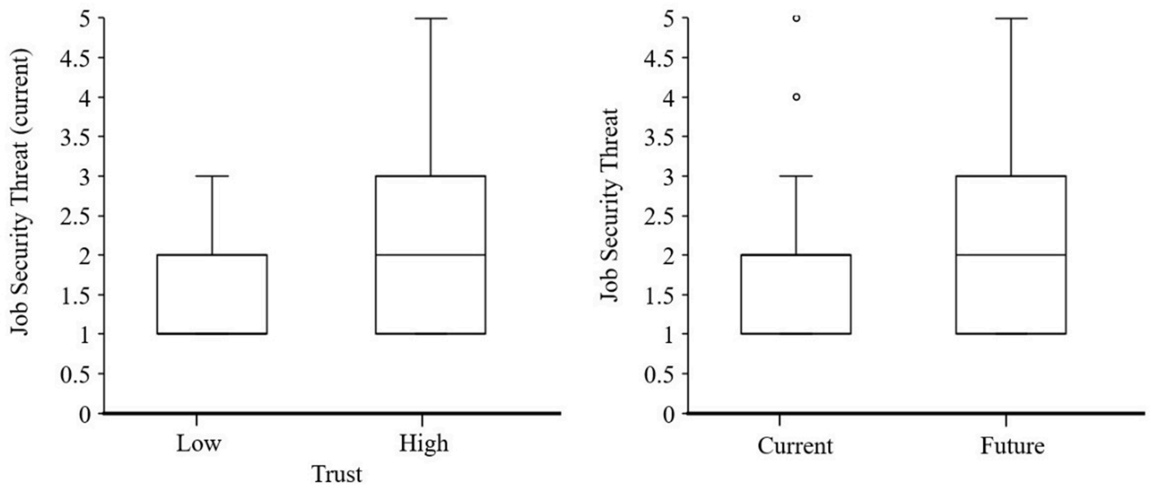


Fig. 6. (left) Box plot showing the perceived job security threat difference between participants with a higher and lower trust of AI tools. (right) Box plot showing the difference between the job security threat now and in the future.

Table 14
Testing the significance of relationships between the trust and perceived productivity variables.

Hypothesis	U-value	p-value	Cohen's d
Trusting the AI tools significantly differs based on the productivity level associated with the tool. • Group 1: Trust values for participants who have achieved higher productivity with the tool (productivity>=3) ($N = 50$, μ=3.929, SD= 0.509) • Group 2: Trust values for participants who have not achieved high productivity with the tool (productivity<=2) ($N = 49$, μ=3.346, SD= 0.835) * Hypothesis supported	1709.0	0.0006*	0.843

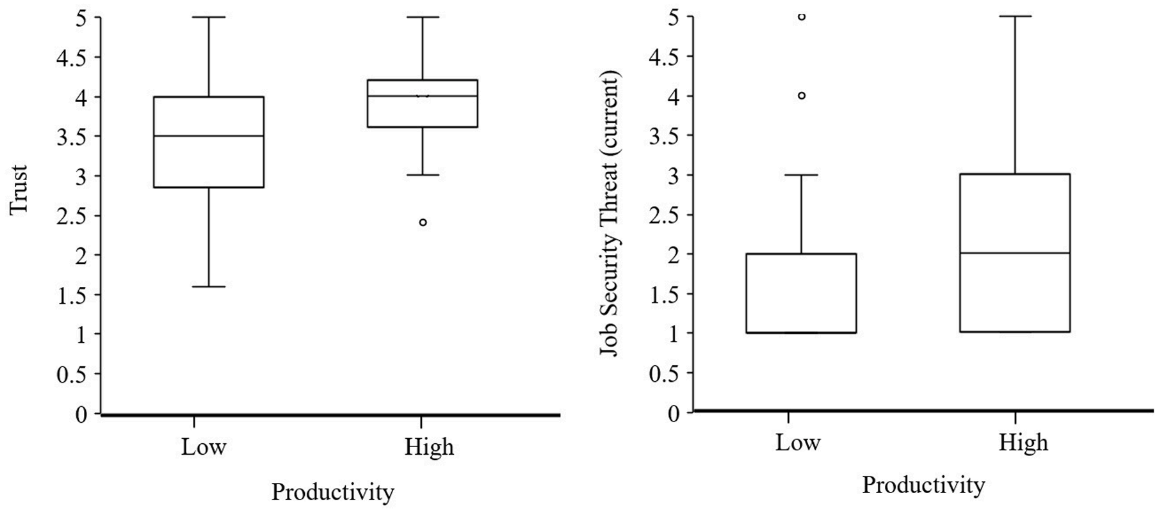


Fig. 7. (left) Box plot showing the differing levels of trust in AI tools among participants based on the productivity associated with the tools. (right) Box plot showing differing perceptions of job security threats related to AI tools among participants based on the productivity associated with the tools.

While using AI to generate code is an evolving field, we also explore the current limitations of these tools. Additionally, we provide expert perspectives from the software industry on the impact of AI tools on jobs and the future of coding education.

6.1. Effectiveness of chatGPT 3.5

In answering RQ1, we found a substantial connection between the problems' complexity level and the success rate of the programs generated using ChatGPT 3.5. The likelihood of failure of the code created by ChatGPT 3.5 increases in proportion to the complexity of the tasks. In most instances, code failure means the code compiles and runs but does not meet the expected results against various test cases or, in some instances, exceeds the time allotted for the problems by LeetCode. We did not discover any significant correlation between the popularity of the problem and the success rate of the programs generated by ChatGPT 3.5. Nevertheless, we observed a slight inclination suggesting that the problems with higher popularity scores tend to have a higher success rate than those with lower scores. Our study did not show any significant relationship between the time or memory performance of the programs written by ChatGPT-3 (compared to other submissions on LeetCode) and the complexity of the problems.

Our results of testing the effectiveness of ChatGPT 3.5 in solving programming problems on LeetCode are similar to those reported by OpenAI (2023), where the researchers also found that the passing rate of ChatGPT (versions 3 and 3) decreases as the problem complexity increases. Similar findings are reported by a recent study [83]. The researchers also found that the new version of ChatGPT (version 4) outperforms other language models and is comparable to human performance. The research on this topic remains scant. Nonetheless, anecdotal evidence based on some developers' experience provides more details on the tool's effectiveness. For instance, a developer reported that although some of ChatGPT's solutions to hard LeetCode problems could have syntactical problems, they are still valuable and can be fine-tuned until they pass all the test cases [84].

Concerning the relationship between the popularity of the problems and the success rate, we speculate that ChatGPT performs better at popular problems because solutions to such problems are widely available on the internet. This finding implies that the quality of the produced solutions is directly proportional to the quality and variety of the training data. However, more exploration is needed in future studies.

Finally, we can say that the time performance and memory usage of ChatGPT 3.5′s solutions are comparable to those of humans. However, we found no significant correlation with problem complexity or popularity. It is important to note that the data on time performance and memory usage is variable and depends on LeetCode's server resources. Therefore, our results should be interpreted with caution and are only indicative.

In the next subsections of the discussion, we discuss our answer to RQ2.

6.2. Power and limitations of AI tools

This research has shown that AI tools are being used to assist programmers in the generation of boilerplate code, explanation of code, research, finding errors in code, commenting on the code, documentation, improving code performance, and writing test cases, among others. Some other interesting use cases, though reported only rarely, include converting the code syntax to conform to other standards and checking the quality of the solutions.

In short, the most prominent use cases involving AI tools are:

1. Troubleshooting: AI tools can assist users in debugging code, diagnosing issues, and understanding error messages by providing guidance on possible causes and suggesting solutions to fix the problem.
2. Code Explanation: AI tools help users seek explanations for specific code snippets, functions, or algorithms by clarifying how the code works, its purpose, and potential use cases.
3. Syntax Assistance: AI aids users in addressing code syntax questions across various programming languages such as Python, JavaScript, Java, and C++.
4. Best Practices: AI offers advice on programming best practices, design patterns, and efficient ways to implement specific features based on common industry practices and standards.
5. Concept Clarification: AI helps users understand complex IT or coding concepts, such as data structures, algorithms, programming paradigms, and software architecture.
6. Tool Recommendations: AI provides recommendations for tools, libraries, or frameworks relevant to users' projects or tasks, suggesting popular and suitable options based on their requirements.

These six main use cases are considered an enhanced user interface for searching knowledge bases more efficiently than the so-far dominant PageRank paradigm [85]. Specifically, use cases 2 and 3 are competitors to tasks already semi-automated by existing development environment plugins. Meanwhile, use cases 1, 4, 5, and 6 are simply new search interfaces that offer enhanced search power functionality compared to knowledge bases such as Stack Overflow (2023), and Google, thanks to the generative power of large language models. In other words, by abstracting, they can answer a larger variety of questions, even if those are not answered in the existing knowledge base. Nonetheless, the literature shows that existing academic AI tools have been proposed to fulfill many of these use cases. For example, AI tools use neural networks to assist programmers in finding code based on natural language queries [86]. Other AI tools can be used in code annotation [87], error correction [88], code documentation generation [89], and the generation of test cases [90]. However, among our sample of programmers, only recently available commercial AI tools such as ChatGPT (2022), Copilot [55], OpenAI Codex [6], and DeepCode (2023) have been more widely adopted. Further, compared to other commercial AI tools, more than two-thirds of the participants reported the usage of ChatGPT. We speculate that the publicity and power of the tool contributed to this rapid adoption.

Despite their demonstrated utility, AI tools are not perfect. According to the participants in this study, AI tools generate faulty and inefficient code, code with security issues, or code not matching all test cases. Further, AI tools can misunderstand or miss the requirements' context and generate poor documentation, among other issues. Extent literature largely agrees with these findings. For instance, a recent study on Copilot reports that sometimes the AI tool generates wrong solutions or misunderstands the requirements [12]. A recent article on ChatGPT [9] reported some tool limitations, including a lack of reliability and logical reasoning. Another article also judged AI tools to lack creativity in their solutions [10].

To conclude, even though ChatGPT and other AI tools have made great strides in recent years, exercising care and not overly relying on them is still vital. AI tools may give significant insights and recommendations but should not be depended upon without human supervision. The critical thinking and problem-solving abilities of human programmers remain highly valuable.

6.3. *Job security of programmers in the context of AI tools*

This study shows more than two-thirds of the participants did not see an imminent job security threat directly related to AI tools. However, the picture is different regarding the future, with about half of the participants feeling a partial to high job security threat. The difference between participants' perceived current and future job security threats is significant. Concerning reasons behind job security threats, participants cited that AI tools can now write code, and it will potentially replace entry-level software programming jobs. On the other hand, participants who did not feel a job security threat cited that AI tools will not be able to replace human programmers as domain knowledge and human ingenuity are still needed to write effective software.

The impact of AI on employability is an ongoing debate among scholars and economists. Some authors speculate that AI tools will replace various jobs in the future [91]. There are speculations that even the tasks of software developers could soon be automatable [92]. Indeed, machine learning progress enables programmers to delegate optimization of sophisticated parameters and designs to a computer program [93] and detect bugs with reliability that humans cannot match [94]. Nonetheless, even with such tools, programmers reportedly want to stay in the loop and not over-rely on them [95]. On the bright side, a study argues that AI tools can create new jobs and industries by generating demand for complementary skills and occupations [96]. Another study reports that while some tasks within jobs may be automated, the overall impact on employment will depend on how tasks can be redefined and redistributed among future employees [97].

6.4. *Learning coding and the evolution of AI tools*

Mixed perspectives were gathered regarding learning coding in the era of AI tools. On the one hand, some participants saw no need to change how aspiring programmers learn to code. Foundational computer science topics must be learned, and learners should only utilize AI tools as a last option. On the other hand, other participants think that the traditional methods used to teach programming are no longer appropriate in this new AI era. These participants think that foundational topics must be taught alongside AI tools while emphasizing problem-solving skills and the limitations of the tools.

The findings from this study align with the recent literature on the rise of AI tools, especially ChatGPT, in education. Some educators are worried about learners overusing the tool and over-relying on it for their assignments [98]. These concerns have resulted in some universities banning ChatGPT to curb plagiarism [99]. On the other hand, there are calls for teaching alongside the tool [100] and implementing a tactical approach to education that is aware of AI [10].

6.5. *Productivity enhancement through AI tools*

Our research indicates that approximately half of the participants experienced a productivity increase ranging from somewhat to highly productive when using AI tools. Various factors contributed to this enhanced productivity, including accelerated coding, the generation of shorter, more efficient, easily readable, and standard-compliant code, as well as facilitating the understanding of colleagues' code.

These findings are consistent with existing studies on AI-driven productivity improvements. For instance, GitHub Copilot's autocomplete functionality has been associated with a 50% increase in productivity [101,102]. Moreover, Noy & Zhang [103] conducted a study involving 444 participants and discovered that ChatGPT-3 significantly boosted average productivity, reducing coding time by 0.8 standard deviations (SDs) and enhancing output quality by 0.4 SDs. The AI tool also improved programmers' self-efficacy, a factor linked to mental health [104].

Our research demonstrates a moderate positive correlation between AI tools' productivity impact and trust levels. Consequently, participants who experienced increased productivity with AI tools exhibited slightly higher trust levels towards these tools than their counterparts. This observation suggests that trust in AI tools may rise as they become more powerful and productivity gains are further realized. While research on this subject is limited due to the novelty of AI tools, a recent study supports our findings, indicating that perceived trust in emerging AI tools could be bolstered by improvements in productivity and task execution [105].

7. Study limitations

Several factors may affect the generalizability of the findings of Study 1. (1) We tested the effectiveness of ChatGPT 3.5 between

January 1, 2023, through February 20, 2024. Our results do not account for ChatGPT-4, launched in March 2023. (2) We selected 180 coding problems representing various computer science topics. However, our selected set of problems may have missed topics that might be significant to other studies. (3) The scope of AI tools assessed in this study was limited to one tool (ChatGPT). The diversity of AI tools on the market, each with unique features and functionalities, suggests that our findings may not apply to other tools not included in our investigation. (4) We retrieved our coding problems from LeetCode, which is just one among many platforms programmers use to prepare for programming job interviews. (5) The assessment of ChatGPT3.5′s effectiveness solely based on coding problems may not necessarily align with everyday programming tasks such as bug fixing, feature enhancement, or generating test cases on top of an existing codebase. Therefore, our findings may not be widely applicable to ChatGPT3.5′s impact on all aspects of software engineering. However, it is also worth noting that high-tech companies, such as Google [106] recommend LeetCode as a platform for software engineer job interview preparation. (6) The time performance and memory usage data obtained from LeetCode are variable and change depending on the LeetCode's server resources. This issue is inevitable in any submission system; therefore, no distinct inference can be made by comparing the time utilization and memory usage. As such, it would be an interesting research problem to explore in the future. (7) It is known that AI tools like ChatGPT can produce different outputs for the same prompt, which is referred to as non-determinism [70]. To address this issue, we conducted three measurements for each coding problem and calculated an aggregated score by combining the three measurements. We also used Cramér's V to test for the effect size. Some of these techniques were recommended by a recent study to counter the non-determinism of ChatGPT [107]. However, inevitably, the non-deterministic nature of ChatGPT may still affect the validity of our study. (8) It's worth noting that AI-powered tools like ChatGPT have limitations regarding the number of tokens they can process. However, we selected coding problems that were short and minimal in their requirements. As such, the coding problems were not truncated by ChatGPT 3.5. Additionally, the solutions generated by ChatGPT 3.5 weren't truncated either. Nonetheless, it's important to remember that the token limitation of LLMs could pose a challenge regarding more complex and larger coding problems.

We also identified several factors that may affect the generalizability of the findings of Study 2. (1) Our questionnaire targeted individuals with programming experience, many of whom worked in the software development industry. However, a few participants serve in other industries despite having a programming background. As such, our sample may not accurately represent the entire developer community. (2) It is important to note the possibility of confounding variables that were not controlled within the survey, which might impact the validity and interpretation of the results. One significant potential confounder that we must acknowledge is the varying experience levels of participants. The effectiveness of AI tools like ChatGPT could be influenced by how proficiently individual users can leverage these tools to solve programming problems. (3) The potential for measurement bias is a limitation of the subjectivity inherent in the self-reported data gathered through our questionnaire may have introduced systematic errors due to recall inaccuracies, personal biases, or misunderstandings of the questions. For example, some participants might have unintentionally overestimated their productivity improvements or underestimated it. Further, the 99 participants demographic might be skewed in unforeseen ways. (4) Finally, an additional constraint of our study was the absence of control over how long they have known or used the AI tool. Controlling for "experience" length in time would have allowed us to control this aspect and time factors that might influence attitudes. For example, it is documented that employees initially resist change to news tools [108].

8. Conclusion

Considering the wide array of job functions that fall under the software development umbrella (system engineering, solution architecture, usability, user experience, and so on), in this study we focused on the programming or analyst job function. This research aimed to evaluate the effectiveness of ChatGPT 3.5 and gather programmers' perspectives on trust, job security, and perceived productivity gains. Our findings indicate that ChatGPT 3.5 effectively solves easy LeetCode problems, while its performance diminishes for harder problem categories. The study shows a weak trend indicating that ChatGPT 3.5 is more effective with coding problems with higher popularity scores. Additionally, the study highlights the value of AI tools for various use cases, including troubleshooting, code explanation, syntax analysis, and more. However, these tools are not flawless, as they may generate incorrect, inefficient, or insecure code, struggle with context understanding, or produce inadequate documentation, among other issues. The trade-off between speedier coding and secure code is worth further attention.

Regarding fear of being replaced, most participants did not perceive an immediate threat to job security but expressed concerns about future job stability. Opinions were divided regarding the impact of AI tools on programming education. Some participants did not see the need to change traditional teaching methods, while others advocated incorporating AI tools alongside foundational computer science topics. Over half of the participants reported that AI tools contributed to increased productivity, ranging from somewhat to highly productive. Furthermore, the study revealed that participants who experienced greater productivity with AI tools exhibited higher trust levels in these tools but also felt a heightened sense of job security threat.

Future studies should consider incorporating controls for participant experience or conducting stratified analyses to determine the AI tool's effectiveness across varying levels of programming expertise. This would provide a more accurate reflection of the tool's capabilities and a deeper understanding of its utility across the programming community, from novices to seasoned professionals.

Funding

This research has been partially funded by Zayed University, UAE, under grant number 23014.

Institutional review board statement

This study was approved by the Research Ethics Committee of the university.

Informed consent statement

Informed consent has been obtained from the participants to publish this study.

Declaration of competing interest

The authors declare no conflict of interest.

Appendix

A. Questionnaire

No.	Question
1.	Age:______
2.	Job:______
3.	Country of Residence:__________
4.	Sex: ○ Male ○ Female ○ I prefer not to say.
5.	Rate your expertise in programming:
	I'm a beginner. I just know the basics 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ I'm a seasoned programmer.
6.	What is your domain of expertise (choose all that apply)?
	☐ Backend developer ☐ Frontend developer ☐ Database administrator ☐ Software architect ☐ Data scientist ☐ Data engineer ☐ Gaming developer ☐ Software engineer ☐ Mobile application developer ☐ Other
7.	What programming language(s) are you an expert at?
	☐ Python ☐ Java ☐ JavaScript ☐ PHP ☐ C ☐ C++ ☐ C# ☐ Perl ☐ Visual Basic ☐ Other
8.	What programming language(s) do you use in your daily job?
	☐ Python ☐ Java ☐ JavaScript ☐ PHP ☐ C ☐ C++ ☐ C# ☐ Perl ☐ Visual Basic ☐ Other
9.	What AI tools do you use to help you with programming (Choose all that apply)?
	☐ ChatGPT ☐ OpenAI Codex ☐ Tabnine ☐ CodeT5 ☐ Polycoder ☐ Cogram ☐ GitHub Copilot ☐ DeepCode ☐ Other. Please specify:________
10.	How frequently do you use ChatGPT/AI tools to help you with programming?
	Rarely (almost never) 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Very frequently (everyday)
11.	How helpful is ChatGPT/AI tools with programming?
	Not helpful at all 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Very helpful
12.	What do you use ChatGPT/AI tools for (Choose all that apply)?
	☐ Generate boilerplate code (Ask the tool to create a skeleton of the app using the technologies, and frameworks you choose).
	☐ Research (building different proofs of concept and compare them).
	☐ Explain the code.
	☐ Comment the code.
	☐ Write test cases.
	☐ Write documentation.
	☐ Generate regexes.
	☐ Rewrite the code using certain coding conventions.
	☐ Find errors in the code.
	☐ Improve the performance of a function.
	☐ Other. Please specify:________
13.	What other ways has ChatGPT/other tools helped you with your coding job? Please elaborate:__________________________________
14.	In what ways has ChatGPT/other AI tools helped you become more productive (Choose all that apply).
	☐ I'm able to finish my coding tasks faster.
	☐ I'm able to write shorter code.
	☐ I'm able to write better-performing code.
	☐ I'm able to write code that has fewer bugs.
	☐ I'm able to write code with better documentation.
	☐ I'm able to write code that is easier to read.
	☐ I'm able to write code that conforms to the standards better.
	☐ I'm able to understand the code of my colleagues faster.
	☐ Other. Please specify:________
15.	How much more productive has ChatGPT/AI tools made you as a programmer?
	Same productivity. Nothing has changed. 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ I've become very productive with ChatGPT/AI tools.
16.	If you use ChatGPT for coding, what are some of the downsides/limitations that ChatGPT/AI tools have (Check all that apply)?
	☐ Errors in the code.
	☐ Inefficient code.
	☐ Code that doesn't match the requirements.
	☐ Code that doesn't work for all test cases.

(*continued on next page*)

(*continued*)

☐ Incorrect explanation of code.
☐ Low quality documentation.
☐ Code with security vulnerabilities.
☐ Other. Please specify:________

17. If you use ChatGPT for other than coding, what are some of the downsides/limitations that ChatGPT/AI tools have (Check all that apply)?
☐ Low quality documentation.
☐ Missing the context of the requirements.
☐ Misunderstanding the requirements of the problem.
☐ Erroneous solutions.
☐ Other. Please specify:________

18. How do you think ChatGPT/AI tools will evolve in the future?
____

19. How should aspiring programmers learn coding in light of ChatGPT/AI tools?
____

20. Do you think ChatGPT/AI tools affect your job security as a programmer now? (reduce your value as a programmer)
Not at all. 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ I feel my job is threatened by ChatGPT/AI tools.

21. Do you think ChatGPT/AI tools affect your job security as a programmer in the future?
It will not affect my job in the future. 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ There is a high chance that ChatGPT/AI tools will affect my job security as a programmer in the future.

22. Why do you think ChatGPT/AI tools affects (or not) your job security as a programmer now or in the future?
____

23. Please answer the following questions:
a) I feel ChatGPT/AI tools are competent in the help they provide.

I totally disagree ○ I disagree ○ I neither agree nor disagree ○ I agree ○ I totally agree ○

a) I feel ChatGPT/AI tools are effective in the help they provide.

I totally disagree ○ I disagree ○ I neither agree nor disagree ○ I agree ○ I totally agree ○

a) I feel ChatGPT/AI tools provide help in my best interest.

I totally disagree ○ I disagree ○ I neither agree nor disagree ○ I agree ○ I totally agree ○

a) I feel ChatGPT/AI tools provide trustworthy help.

I totally disagree ○ I disagree ○ I neither agree nor disagree ○ I agree ○ I totally agree ○

a) I would recommend ChatGPT/AI tools to fellow programmers.

I totally disagree ○ I disagree ○ I neither agree nor disagree ○ I agree ○ I totally agree ○

References

[1] Z. Chen, et al., Sequencer: Sequence-to-sequence learning for end-to-end program repair, IEEE Transactions on Software Engineering 47 (9) (2019) 1943–1959.
[2] M. Yasunaga, P. Liang, Break-it-fix-it: Unsupervised learning for program repair. s.l, in: Proceedings of the 38th International Conference on Machine Learning, 2021, pp. 11941–11952.
[3] J. Yao, et al., Learning nonlinear loop invariants with gated continuous logic networks. s.l, in: The 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, 2020, pp. 106–120.
[4] M. Basso, A. Rosà, L. Omini, W. Binder, Java Vector API: Benchmarking and Performance Analysis, in: Proceedings of the 32nd ACM SIGPLAN International Conference on Compiler Construction (CC 2023), New York, NY, USA, 2023, pp. 1–12.
[5] M. Rahmouni, M. Bouzaidi, S. Mbarki, Approach by modeling to generate an e-commerce web code from laravel model, Indonesian Journal of Electrical Engineering and Computer Science 30 (1) (2023) 257–266.
[6] Codex, O., 2022. [Online] Available at: https://openai.com/blog/openai-codex[Accessed 6 4 2023].
[7] Chen, M. et al., 2021. Evaluating Large Language Models Trained on Code. ArXiv.
[8] Hu, K., 2023. ChatGPT sets record for fastest-growing user base - analyst note. [Online] Available at: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/[Accessed 6 4 2023].
[9] J. Zhou, et al., ChatGPT: potential, prospects, and limitations, Front Inform Technol Electron Eng (2023).
[10] Y.K. Dwivedi, et al., So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy, Int. J. Inf. Manage 71 (2023) 102642.
[11] M. Cascella, J. Montomoli, V. Bellini, E. Bignami, Evaluating the Feasibility of ChatGPT in Healthcare: An Analysis of Multiple Clinical and Research Scenarios, J. Med. Syst. 47 (1) (2023).
[12] M. Wermelinger, Using GitHub Copilot to Solve Simple Programming Problems, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education, Toronto, 2023, pp. 1–10.
[13] Nikolaidis, N. et al., 2023. The End of an Era: Can Ai Subsume Software Developers? Evaluating Chatgpt and Copilot Capabilities Using Leetcode Problems. SSRN.
[14] N. Nascimento, P. Alencar, D. Cowan, Artificial Intelligence vs. Software Engineers: An Empirical Study on Performance and Efficiency using ChatGPT, in: Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering (CASCON '23), Las Vegas, USA, 2023.
[15] Zuccon, G. & Koopman, B., 2023. Dr ChatGPT, tell me what I want to hear: How prompt knowledge impacts health answer correctness. ArXiv.
[16] H. Ge, Y. Wu, An Empirical Study of Adoption of ChatGPT for Bug Fixing among Professional Developers, Innovation & Technology Advances 1 (1) (2023).

[17] J.Y.J. Liang, B. Myers, A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges, in: IEEE/ACM 46th International Conference on Software Engineering (ICSE), Lisbon, Portugal, 2024.
[18] M. Ciniselli, et al., An Empirical Study on the Usage of BERT Models for Code Completion, in: IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), Madrid, Spain, 2021, pp. 21–29.
[19] P. Vaithilingam, T. Zhang, E.L. Glassman, Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models, in: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22), New York, NY, USA, 2022, pp. 1–7.
[20] S.L. Tanimoto, Five Futures with AI Coding Agents, in: Companion Proceedings of the 7th International Conference on the Art, Science, and Engineering of Programming (Programming '23), Tokyo, Japan, 2023.
[21] Wang, R., Cheng, R., Ford, D. & Zimmermann, T., 2023. Investigating and Designing for Trust in AI-powered Code Generation Tools. ArXiv.
[22] A. Agrawal, J.S. Gans, A. Goldfarb, Artificial Intelligence: The Ambiguous Labor Market Impact of Automating Prediction, J. Econ. Perspect. 33 (2) (2019) 31–50.
[23] J. Kleinberg, et al., Human Decisions and Machine Predictions, Q. J. Econ. 133 (1) (2018) 237–293.
[24] D. Acemoglu, P. Restrepo, The Race between Man and Machine: Implications of Technology for Growth, Factor Shares, and Employment, American Economic Review 108 (6) (2018) 1488–1542.
[25] LeetCode, 2023. [Online] Available at: https://leetcode.com/[Accessed 6 4 2023].
[26] M.A. Kuhail, N. Alturki, S. Alramlawi, Interacting with educational chatbots: A systematic review, Educ Inf Technol 28 (2023) 973–1018.
[27] A. Xu, et al., A new chatbot for customer service on social media. s.l, in: Proceedings of the 2017 CHI conference on human factors in computing systems, 2017, pp. 3506–3510.
[28] K. Fitzpatrick, A. Darcy, Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial, JMIR. Ment. Health 4 (2) (2017).
[29] D. Kaczorowska-Spychalska, Chatbots in marketing, Management 23 (1) (2019).
[30] Dastin, J., Hu, K. & Dave, P., 2022. Exclusive: ChatGPT owner OpenAI projects $1 billion in revenue by 2024. [Online] Available at: https://www.reuters.com/business/chatgpt-owner-openai-projects-1-billion-revenue-by-2024-sources-2022-12-15/[Accessed 20 4 2023].
[31] B. AbuShawar, E. Atwell, Alice chatbot: Trials and outputs. Computación y Sistemas, Computación y Sistemas 19 (4) (2015) 625–632.
[32] O. Chukhno, N. Chukhno, K. Samouylov, S. Shorgin, A chatbot as an environment for carrying out the group decision making process, s.l., ITTMM (Selected Papers) (2019) 15–25.
[33] Alexa, 2023. What Is Alexa?. [Online] Available at: https://developer.amazon.com/en-US/alexa [Accessed 6 4 2023].
[34] Siri, 2023. [Online] Available at: https://www.apple.com/siri/[Accessed 6 4 2023].
[35] I. Goodfellow, et al., Generative Adversarial Nets. s.l, in: Advances in Neural Information Processing Systems, 27, NIPS, 2014, p. 2014.
[36] s. Vaswani, et al., Attention is all you need. s.l, in: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), 2017.
[37] T.B. Brown, et al., Language models are few-shot learners. s.l, in: Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20), 2020.
[38] Z. Shao, et al., Tracing the evolution of AI in the past decade and forecasting the emerging trends, Expert. Syst. Appl. 209 (2022) 118221.
[39] M. Hutson, Robo-writers: the rise and risks of language-generating AI, Nature 591 (7848) (2021) 22–25.
[40] S. Gulwani, O. Polozov, R. Singh, Program Synthesis. Foundations and Trends, now Publishers Inc, Hanover, MA 02339, 2017.
[41] D. Sobania, D. Schweim, F. Rothlauf, A comprehensive survey on program synthesis with evolutionary algorithms, IEEE Transactions on Evolutionary Computation 27 (1) (2022) 82–97.
[42] R. Bavishi, et al., AutoPandas: neural-backed generators for program synthesis. s.l, in: Proc. ACM Program, Lang, 2019.
[43] S. Gulwani, Automating string processing in spreadsheets using input-output examples, in: Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '11), York, NY, USA, 2011.
[44] D. Fried, et al., InCoder: A Generative Model for Code Infilling and Synthesis, ArXiv. (2022).
[45] A. Solar-Lezama, et al., Combinatorial sketching for finite programs, ACM SIGARCH Computer Architecture News 34 (5) (2006) 404–415.
[46] S. Kulal, et al., Spoc: Search-based pseudocode to code, in: Advances in Neural Information Processing Systems, 32, 2019.
[47] P. Yin, et al., Learning to mine aligned code and natural language pairs from stack overflow. s.l, in: Proceedings of the 15th International Conference on Mining Software Repositories, 2018, pp. 476–486.
[48] E. Dolson, A. Lalejini, C. Ofria, Exploring Genetic Programming Systems with MAP-Elites, in: W. Banzhaf, L. Spector, L. Sheneman (Eds.), Genetic Programming Theory and Practice XVI, Springer, Cham, 2019, pp. 1–16.
[49] K. Stoffel, L. Spector, High-performance, parallel, stack-based genetic programming. s.l, in: Proceedings of the 1st annual conference on genetic programming, 1996, pp. 224–229.
[50] S. Forstenlechner, M. Nicolau, D. Fagan, M. O'Neill, Grammar Design for Derivation Tree Based Genetic Programming Systems, s.l, in: European Conference on Genetic Programming, 2016.
[51] M. Pradel, G. Gousios, J. Liu, S. Chandra, TypeWriter: neural type prediction with search-based validation, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), New York, NY, USA, 2020, pp. 209–220.
[52] S. Lu, et al., CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation, ArXiv. (2021).
[53] AlphaCode, D., 2023. [Online] Available at: https://www.deepmind.com/blog/competitive-programming-with-alphacode[Accessed 7 4 2023].
[54] CodeWhisperer, A., 2023. [Online] Available at: https://aws.amazon.com/codewhisperer/[Accessed 7 4 2023].
[55] Copilot, G., 2023. [Online] Available at: https://github.com/features/copilot[Accessed 7 4 2023].
[56] B.A. Becker, et al., rogramming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education, New York, NY, USA, 2023.
[57] J. Finnie-Ansley, P. Denny, B.A. Becker, The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming, Australasian Computing Education Conference, NY NY, USA, 2022, pp. 10–19.
[58] Y. Li, et al., Competition-level code generation with AlphaCode, Science (1979) 378 (6624) (2022) 1092–1097.
[59] A. Koubaa, et al., Humans are still better than ChatGPT: Case of the IEEEXtreme competition, Heliyon. 9 (11) (2023).
[60] S. Lertbanjongngam, et al., An Empirical Evaluation of Competitive Programming AI: A Case Study of AlphaCode, in: IEEE 16th International Workshop on Software Clones (IWSC), Limassol, Cyprus, 2022.
[61] S. Imai, Is GitHub copilot a substitute for human pair-programming? an empirical study, in: Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22), Pittsburgh, USA, 2022.
[62] S. Thakur, et al., Benchmarking Large Language Models for Automated Verilog RTL Code Generation, in: IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2023.
[63] J. Balakrishnan, S. Abed, P. Jones, The role of meta-UTAUT factors, perceived anthropomorphism, perceived intelligence, and social self-efficacy in chatbot-based services? Technol. Forecast. Social Change 180 (2022) 121692.
[64] J. Balakrishnan, Y. Dwivedi, Conversational commerce: Entering the next stage of AI-powered digital assistants, Ann. Oper. Res. (2021) 1–35.
[65] A. Stavridis, A. Drugge, The Rise of Intelligent System Development: A Qualitative Study of Developers' Views on AI in Software Development Processes, Umeå University, Umeå, Sweden, 2023.
[66] E.S. Zhan, M.D. Molina, M. Rheu, W. Peng, What is There to Fear? Understanding Multi-Dimensional Fear of AI from a Technological Affordance Perspective, International Journal of Human–Computer Interaction (2023).
[67] A. Milanez, The impact of AI on the workplace: Evidence from OECD case studies of AI implementation. Employment and Migration Working Papers, OECD Social, Paris, France, 2023.

[68] J. Harper, A Software Engineer's Guide to Seniority: A Guide to Technical Leadership, 1st ed., Apress, Berkeley, 2022.
[69] Nikolaidis, N. et al., 2023. The End of an Era: Can Ai Subsume Software Developers? Evaluating Chatgpt and Copilot Capabilities Using Leetcode Problems. SSRN.
[70] Ouyang, S., Zhang, J.M., Harman, M. & Wang, M., 2023. LLM is Like a Box of Chocolates: The Non-determinism of ChatGPT in Code Generation.. https://arxiv.org/.
[71] M.T. Goodrich, R. Tamassia, M.H. Goldwasser, Data Structures and Algorithms in Python, 1st edition, John Wiley & Sons Inc, Hoboken, 2013.
[72] D. McKnight, V. Choudhury, C. Kacmar, Developing and validating trust measures for e-commerce: An integrative typology, Inf. Syst. Res. 13 (2002) 334–359.
[73] L. Qiu, I. Benbasat, Evaluating anthropomorphic product recommendation agents: A social relationship perspective to designing information systems, J. Manag. Inf. Syst. 25 (2009) 145–182.
[74] F. Reinkemeier, U. Gnewuch, Match or mismatch? How matching personality and gender between voice assistants and users affects trust in voice commerce, in: Proceedings of the 55th Hawaii International Conference on System Sciences, Lahaina, HI, USA, 2022.
[75] C. Syverson, What determines productivity? J. Econ. Lit. 49 (2) (2011) 326–365.
[76] Authors, 2023. Github. [Online] Available at: https://github.com/kuhailamin/ChatGPT_data[Accessed 27 11 2023].
[77] W.G. Cochran, The Chi-square Test of Goodness of Fit, The Annals of Mathematical Statistics 23 (3) (1952) 315–345.
[78] H. Cramér, Mathematical Methods of Statistics. s.l, Princeton University Press, Princeton, 1946.
[79] G.W. Corder, D.I. Foreman, Nonparametric Statistics: A Step-By-Step Approach. s.l, Wiley, 2014.
[80] F. Ruland, The Wilcoxon-Mann-Whitney Test—An Introduction to Nonparametrics with Comments on the R Program Wilcox.Test, Independently Published, Chicago, IL, USA, 2018.
[81] M. Forbes, Thematic analysis: A practical Guide, Sage Publications, London, England, 2022.
[82] M.J. Diener, Cohen's d, Wiley, 2010.
[83] Bubeck, S.C.V.E.R. et al., 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. ArXiv.
[84] XQ, ?. Solving Leetcode Hard Problems with ChatGPT, 2023 [Online] Available at: https://medium.com/the-research-nest/solving-leetcode-hard-problems-with-chatgpt-1399b0fa432d [Accessed 7 4 2023].
[85] L. Page, S. Brin, R. Motwani, T. Winograd, The Pagerank Citation ranking: Bring order to the web, s.l, Stanford University, 1998.
[86] X. Gu, H. Zhang, S. Kim, Deep code search, in: Proceedings of the 40th International Conference on Software Engineering (ICSE '18), New York, NY, USA, 2018.
[87] Z. Yao, J.R. Peddamail, H. Sun, CoaCor: Code Annotation for Code Retrieval with Reinforcement Learning, in: he World Wide Web Conference (WWW '19), New York, NY, USA, 2019.
[88] L. Huang, et al., AI Coding: Learning to Construct Error Correction Codes, IEEE Transactions on Communications 68 (1) (2020) 26–39.
[89] A.Y. Wang, et al., Documentation Matters: Human-Centered AI System to Assist Data Science Code Documentation in Computational Notebooks, CM Trans. Comput.-Hum. Interact. 29 (2) (2022).
[90] M. Bhavya, A. Damodaran, S. Ranganath, An AI based Smart Test Case Generator for Embedded Device, in: Second International Conference on Power, Control and Computing Technologies (ICPC2T), Raipur, India, 2022.
[91] TheGuardian, 2023. US experts warn AI likely to kill off jobs – and widen wealth inequality. [Online] Available at: https://www.theguardian.com/technology/2023/feb/08/ai-chatgpt-jobs-economy-inequality [Accessed 7 4 2023].
[92] C.B. Frey, M.A. Osborne, The future of employment: How susceptible are jobs to computerisation? echnological Forecasting and Social Change 114 (2017) 254–280.
[93] H.H. Hoos, Programming by optimization, Communications of the ACM 55 (2) (2012) 70–80.
[94] S. Hangal, M.S. Lam, Tracking down software bugs using automatic anomaly detection, in: Proceedings of the 24th International Conference on Software Engineering (ICSE '02), New York, NY, USA, 2002.
[95] Winter, E. et al., 2022. How Do Developers Really Feel About Bug Fixing? Directions For Automatic Program Repair. IEEE Transactions on Software Engineering, pp. 1–20.
[96] J.E. Bessen, AI and Jobs: The Role of Demand, Boston Univ. School of Law, Law and Economics Research, 2017, pp. 17–46.
[97] M. Arntz, T. Gregory, U. Zierahn, The Risk of Automation for Jobs in OECD Countries: A Comparative Analysis, OECD Social, Employment and Migration Working Papers, 2016.
[98] Marche, S., 2022. The College Essay Is Dead: Nobody is prepared for how AI will transform academia.. [Online] Available at: https://www.theatlantic.com/technology/archive/2022/12/chatgpt-ai-writing-college-student-essays/672371/[Accessed 7 4 2023].
[99] Reuters, 2023. Top French university bans use of ChatGPT to prevent plagiarism. [Online] Available at: https://www.reuters.com/technology/top-french-university-bans-use-chatgpt-prevent-plagiarism-2023-01-27/[Accessed 7 4 2023].
[100] Roose, K., 2023. Don't Ban ChatGPT in Schools. Teach With It.. [Online] Available at: https://www.nytimes.com/2023/01/12/technology/chatgpt-schools-teachers.html[Accessed 7 4 2023].
[101] Peng, S., Kalliamvakou, E., Cihon, P. & Demirer, M., 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. ArXiv.
[102] M. Wermelinger, Using GitHub Copilot to Solve Simple Programming Problems, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), New York, NY, USA, 2023, pp. 172–178.
[103] Noy, S. & Zhang, W., 2023. Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence. SSRN.
[104] A. Bandura, C. Pastorelli, C. Barbaranelli, G.V. Caprara, Self-efficacy pathways to childhood depression, J. Pers. Soc. Psychol. 76 (2) (1999) 258–269.
[105] O. Bitkina, et al., Perceived trust in artificial intelligence technologies: A preliminary study, Hum Factors Man 30 (2020) 282–290.
[106] Google, 2023. Tech Dev Guide. [Online] Available at: https://techdevguide.withgoogle.com/resources/sources/leetcode/?no-filter=true[Accessed 28 11 2023].
[107] OuYang, S., Zhang, J., Harman, M. & Wang, M., 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. ArXiv.
[108] L. Smirani, J. Boulahia, Using the unified theory of acceptance and use of technology to investigate the adoption of open educational resources by faculty members, International Journal of Information Technology 14 (6) (2022) 3201–3211.