DOI: 10.71317/jgst.2.6.2026.284
Journal of Global Social Transformation
Journal homepage: https://rjsaonline.org/index.php/JGST

The Influence of AI-Assisted Programming Tools on Coding Competency Among Computer Science Students

Huzaifa Javed[1], Faheem Ahmed[2], Afrasiyab Ali[3], Kanza Zahra[4]
[1] COMSATS University Wah Campus
Email: huzaifaj247@gmail.com
[2] Assistant Professor, Department of Information Technology University of Sindh, Jamshoro
Email: faheem.abbasi@usindh.edu.pk
[3] Chicago College, USA
Email: mi44199@mail.harpercollege.edu / afrasiyabali2003@gmail.com
[4] Berkeley City College, USA
Email: 30083051@cc.peralta.edu / kanzazahra.624@gmail.com

ARTICLE INFO

Received:
April 16, 2026
Revised:
May 04, 2026
Accepted:
May 29, 2026
Available Online:
June 20, 2026

Keywords:

artificial intelligence, AI-assisted programming tools, coding competency, computer science education, debugging skills, cognitive offloading, student perceptions, independent problem-solving.

Corresponding author:
huzaifaj247@gmail.com

ABSTRACT

This study examined the influence of AI-assisted programming tools on the coding competency of computer science students, with particular attention to how such tools affect debugging, independent problem-solving, and overall skill development. A descriptive correlational research design was utilized based on a structured questionnaire. The survey was built using validated scales on AI programming tools and coding. This survey was distributed with a few adjustments and was tested on undergraduate computer science students from universities in the Punjab and Sindh regions of Pakistan. Convenience sampling was applied for the survey, and ultimately, a sample of 268 students was surveyed. Respondents had to provide informed consent, and the data were collected through Google Forms. The survey was shared in the respective university class groups and through official student email and social media. The privacy of the respondents was assured, and the data were collected using quantitative analysis. The AI programming tools and coding skills research focused on the reliability analysis and the descriptive and principal component analysis. The results of this study will benefit students of computer science and coding. This will help educators structure tools in a way that will help students improve AI programming and maintain coding skills that offer students tools to remain independent coders and aid researchers to explain the use of AI in the coding and programming skills of university students for further research.

Introduction

Generative AI has transformed how computer science students approach programming. Whenever new technologies emerge, they go from being seen as experimental to becoming important tools that are at the forefront of coding education. These tools provide coding tips on-the-spot or help debug coding problems and explain complicated programming concepts in layman's terms. There are AI chatbots that are able to code fluently, so they are able to generate code, debug code, and explain code (Yen et al., 2025). With these AI technologies, higher education institutions are conducting more research to study the impact that these tools have on coming generations of computer science students. These tools are forcing computer

science departments to rethink the teaching and learning frameworks and the assessments that they use (Alanazi, M. et al., 2025). arxiv

The first studies that looked at the impact of these technologies on education produced findings that were jaw-dropping for the average computer science educator. GitHub Copilot, in its infancy, was already grading computer science students in the first 25 percentile of all students that take AP Computer Science courses (Denny et al., as cited in Hubwieser et al., 2025). This was troubling for educational institutions that were trying to figure out how to teach computer science in the age of Assisted-AI programming. After this initial study, other studies looked at how these technologies affect the cognitive processes and task completion of novice programmers. Gardella (2024) examined the impact of AI-assisted development tools on the performance, learning, and attitudes of students towards programming by including a sample of undergraduate students enrolled in an introductory programming course. After an introductory lesson, students completed a series of programming exercises both with AI assistance and independently.

The growing evidence base from a range of disciplines and contexts has produced mixed findings. For some, the introduction of AI programming tools has provided a notable pedagogical advancement. Research surrounding the use of AI programming generators in entry level programming courses have found that novice programmers using the generators spent less time and effort to reach a solution, and, even more critically, did not negatively impact the novice programmers' ability to perform manual code edits or complete tasks outside of the AI programming support (Passey, 2017). In a similar study, which focused on the Gemini assistant embedded in the Google Colab environment, an overwhelming majority of the study participants (91 percent) reported that the AI assist surpassed their expectations for educational support (Paiva et al.,, 2022). Recently, more controlled research has shown a growing concern for the gaze of skill mastery. For one randomized controlled study, participants who relied on AI assistance had a significantly lower mastery of the concepts, showing a difference of more than 17 percent on a quiz of the covered concepts, compared to peers who completed the programming exercises independently (Tan, C et al, 2024).

Researchers have claimed that relying on technology, which provides instant gratification and stimulation, can reduce the likelihood that individuals are willing to take on mentally taxing challenges. This is likely the case for students who rely on AI to provide instant solutions because these students could be less likely to engage in difficult coding problems. This could hinder students' ability to develop basic coding skills (Aru & Rozgonjuk, 2022). At the same time, a related study found that learners mostly use generative AI for help in coding and writing, and use these rapid research aids for help in accelerating the completion of projects. Although, some students have said that this reliance could adversely affect the development of the ability to learn and retain new ideas (Tanay, et al. , 2004). When combined, the findings show that there is a complicated relationship between AI and coding tools in the context of a student's coding ability. The nature of this relationship can be either positive, or negative depending on the context of the specific computer science educational environment.

Early Research on AI-Assisted Coding Performance

We can cite impressive and disturbing findings from the first research on this phenomenon. For instance, Denny and others said GitHub Copilot, using its first version, placed it among the 25% top students in first-year programming at an undergraduate level. That sent many shocks through the education system and helped raise many questions regarding the possible effects AI-assisted programming might have on the development of the programming subsystem (Denny et al., 2024). On the basis of these (and similar) primary concerns, research started to analyze the tools' implications beyond task fulfillments. For example, Fan, G et al. (2025) examined how AI-assisted programming tools affect learning, performance, and attitude toward programming of first-year undergraduate students and compared performance of students in an AI-assisted task to that of students in an unassisted task, using a repeated measures research design.

Mixed Evidence on Learning Outcomes

Research on AI code generators for beginner programmers has shown that, when compared to coding without AI support, the use of AI code generators has not shown any negative impact on manual code modification or completing programming tasks without the use of AI (Long, X, et al., 2024). AI code generators have enabled beginners to code better, quicker, and with less frustration. In a study about the use of the Gemini assistant in the Google Colab environment, 91% of the subjects said the AI support was better than most of the support they received for learning (Masetty, S et al., 2025). Recently, AI tools have been shown to negatively affect the retention of skills learned. One such study reported that, in a programming task where participants were asked to code the same concept, the participants who received AI support scored 17% lower on a quiz about the coding task (Chang, H, et al., 2025; as discussed in InfoQ, 2026).

Cognitive Engagement and Motivation Concerns

It has been suggested that extensively used technologies that reward users in the short term and provide novelty could demotivate users from performing cognitively challenging and effortful tasks. This suggests that students who frequently use AI to get answers may find working on difficult coding tasks too demanding and may lack the motivation to work on such tasks, hence compromising the development of important skills (Butt, D, et al. 2026). A related study suggested that students mainly use generative AI to assist them with programming and writing to help them complete research and develop projects in a time-efficient manner. This study noted that some students believed that using such tools may negatively impact their skills to learn and retain new concepts (Valový, M, et al. 2023). Considering all that has been mentioned, the findings indicate that the relationship between the use of AI to assist programming and the development of coding skills is neither positive nor negative, but rather is dependent on the context and manner within which students use AI tools throughout their computer science courses.

Research Objectives

- To examine the extent to which AI-assisted programming tools influence the coding competency of computer science students.
- To identify the specific ways AI-assisted programming tools affect students' debugging, problem-solving, and independent coding abilities.
- To assess students' perceptions of AI-assisted programming tools and their reported impact on learning outcomes in computer science education.

Research Questions

- To what extent do AI-assisted programming tools influence the coding competency of computer science students?
- How does the use of AI-assisted programming tools affect students' debugging, problem-solving, and independent coding abilities?
- What are computer science students' perceptions of AI-assisted programming tools, and how do these perceptions relate to their learning outcomes?

Problem Statement

Computer Science Education has rapidly embraced AI-assisted programming tools. Their actual impact on students' coding proficiency is largely unknown. One line of research argues that these tools lessen frustration and improve the efficiency with which tasks are completed, while not impacting the ability and willingness to code independently. Conversely, other studies argue that using AI tools creates a poor learning environment, and a lack of cognitive effort, which is essential for mastering the fundamental concepts and skills of programming. Given the contradictory findings, the effect of AI-assisted tools on programming skills is poorly understood and creates an urgent challenge for educators to develop strategies and guidelines on the appropriate degree of encouragement, restriction, and scaffolding of AI-assisted tools in programming curricula.

Significance of the Study

The value of this research is largely based on the consideration of the impact of AI-assisted programming tools, now ubiquitous in teaching coding to students, on an important and unresolved question in computer science education. For educators and developers of instructional programs, this research provides the opportunity to justify the controlled use of AI tools in programming courses. Students may learn to use AI tools to assist with feedback on the effects on their ability to code and debug. Researchers and educators gain an understanding of the effects of AI on computing education, and skill development.

The Evolution and Adoption of AI-Assisted Programming Tools in Computer Science Education

AI-assisted programming tools have undergone rapid improvements over the years, transforming what started as a basic code-completion feature into a more complex artificial intelligence that can write, explain, and debug code. According to Thoughtworks (2026), GitHub Copilot was first revealed in June 2021 and then made available publicly in June 2022. GitHub Copilot became the first preferred generative AI coding tool for coders of all skill levels. The first version of Copilot was an easy inline suggestion tool. As stated, it progressed into a code assistant, then Copilot Chat, and most recently, Agent Mode. It became a popular tool among software engineers across the globe. This progression is part of a larger trend in software

development. The AI coding tools have become more sophisticated, more powerful, and more integrated into a developer's workflow.

Software development tools and educational tools have advanced and integrated at a comparable pace. Recent feedback from the 2025 Stack Overflow developer survey indicates that, on average, over half of professional software developers use AI tools on a daily basis. The integration of AI tools in educational settings has progressed at a significantly faster pace. According to Engineering Students' Usage and Perceptions of GitHub Copilot in Open-Source Projects (2026), the use of AI technology in higher education increased from 53% in 2024 to 88% in 2025, suggesting that the norm rather than the exception was rapid adoption among computer science students. Industry statistics indicate that the same pattern occurs, but at a more drastic level. For instance, Second Gao (2025) notes that GitHub Copilot went from being an experimental tool during its 2022 commercial launch, to being an enterprise standard tool used by 90% of Fortune 100 companies, and with 15 million users around the world.

There are several reasons for the rapid adoption of GitHub Copilot and similar tools among computer science students and software developers, but the primary reasons are the increase in productivity and the appeal of these tools. For instance, Peng et al. (2023) conducted research that suggested that participants that used GitHub Copilot as an AI pair programmer completed tasks 55.8% faster, and this finding has been replicated in many studies that followed. Liang et al. (2024) also described some of the benefits of Copilot. These include a reduction in the number of keystrokes, completion of tasks at a faster rate, and a quicker recall of coding syntax. These benefits may not always relate to faster outcomes or higher task completion success rates, but according to Vaithilingam et al. (2022), Copilot is preferred for mundane coding tasks because it brings clarity by producing a good first draft rather than requiring time-consuming and laborious research. These results indicate that AI-augmented tools have attractive qualities that exceed basic quantitative aspects and have to do with cognitive and workflow aspects.

With the continuous adoption of these tools, researchers have attempted to understand the reasons and mechanisms that cause students to continue using them. In a study Li et al., (2025) the role of AI-based coding assistants in computer education is examined in terms of trust, learning motivation, and perceived reliability of the assistants, and the opportunities and challenges that they represent. Building on the Technology Acceptance Model in this case, motivation to learn, goals, perceived reliability, trust, and elements of doubt and insecurity were relevant in the acceptance and perception of AI coding assistants by undergraduate computer science students in Mexico. There is a greater focus on the role of trust in the adoption of technologies. This focus is supported by an investigation of the evolution of students' attitudes over time. In the study of Shah et al. (2016), trust increased among the 71 upper-division computer science students, although students understood that GitHub Copilot requires a skilled programmer to complete certain tasks.

Effects of AI-Assisted Programming Tools on Students' Coding Competency

Studying the impact of AI programming tool assistants on teaching students how to program shows that the results are mostly in opposition to each other. Multiple studies have shown performance improvements for students when coding with AI help. In one controlled study, ten undergraduate computer science students completed brownfield programming tasks with and without GitHub Copilot, and students using Copilot completed tasks 34.9% faster and made 50.0% more solution progress (Shihab et al., 2025). The same study found that students spent significantly less time composing code and conducting web searches when Copilot was available, indicating a fundamental shift in how they engaged with programming tasks (Shihab et al., 2025). The results of this study, along with others, show how much AI help can lessen the difficulties of working with code that is not familiar, something that many new employees have to deal with and is a task new students will have to do when they enter the job market.

While these studies show performance gains with AI coding help, they also show that not all students gain equally and that how students interact with the AI suggestions matters. A study published in the Proceedings of the 2025 ACM Conference on International Computing Education Research found that higher-performing students tend to use Copilot more selectively, suggesting critical engagement with AI suggestions is beneficial, while students overall expressed concerns about potential superficial understanding despite improved efficiency (Shihab et al., 2025). Understanding how competency effects differ requires noticing this type of distinction, where the use of AI does not uniformly improve or diminish skills, but rather reveals differences in how students approach problem-solving.

While efficiency-related evidence is at the core of these findings, other studies centered on skill retention and mastery present stronger arguments against the costs associated with AI dependency. According to Shen and Tamkin (2026), a randomized controlled trial found that using AI assistance led to a statistically significant decrease in mastery, with participants in the AI-assisted group scoring 17% lower, or nearly two letter grades, on a quiz covering concepts they had used just minutes before, compared to participants who coded by hand. The same researchers noted that this effect appeared specific to skill

acquisition, since their earlier work had shown AI can speed up tasks by as much as 80% when users already possess the relevant skills. It is very likely that this erosion of competency is especially noticeable when students are learning new libraries or tools that are more advanced or complex, than when students are using skills they have already mastered.

Understanding the context is greatly complicated by the fact that even when AI assistance clearly benefits student performance, it does not necessarily damage students' fundamental abilities. According to Kazemitabaar et al. (2023), a controlled experiment with 69 novice learners found that using an AI code generator significantly increased code-authoring performance without decreasing performance on manual code-modification tasks, and access to the code generator reduced student frustration and improved their progression through programming tasks (Kazemitabaar et al., 2023). These findings contrast with the mastery-reduction results reported elsewhere, underscoring that AI programming tools may have competency effects that are highly sensitive to learning task design, the novelty of the learning content, and the timing of the competency assessment relative to AI-augmented learning.

Impact of AI Assistance on Debugging, Problem-Solving, and Independent Coding Skills

Some researchers have begun to argue that the convenience offered by AI assistance may reduce the effort traditionally required to develop debugging and problem-solving abilities through productive struggle. In AI-assisted programming environments, students may increasingly avoid manually interpreting tasks, diagnosing errors, identifying process bottlenecks, and refining code independently. Such dependence can weaken learners' ability to interpret compiler messages and resolve faults in AI-generated programs without external assistance. Evidence from recent educational studies suggests that excessive reliance on generative AI tools can lead to lower conceptual understanding and increased cognitive offloading, where learners transfer part of the reasoning process to the AI system rather than engaging with it themselves (Kazemitabaar et al., 2024).

Research further indicates that students who delegate substantial portions of programming work to AI may complete tasks more quickly and with fewer immediate errors, yet they often perform worse on subsequent assessments that require independent reasoning. Learners who rely on AI primarily for debugging tend to seek explanations and solutions from the assistant rather than systematically analyzing the source of the error. In contrast, students who debug without AI support encounter more syntax, logic, and library-related issues, but the process of resolving these errors appears to strengthen their analytical and debugging capabilities. Educational psychology literature has long emphasized that productive struggle and the experience of overcoming difficulties contribute significantly to durable learning and expertise development (Bjork & Bjork, 2011).

Beyond experimental settings, qualitative investigations of students' experiences with AI coding assistants reveal a similar tension between immediate task efficiency and long-term skill acquisition. Several studies report that when AI tools are removed, many students struggle to explain the concepts underlying their code or to complete programming tasks independently. This suggests that AI assistance may sometimes replace rather than augment students' problem-solving processes. Researchers have also observed discrepancies between students' perceived competence and their actual independent performance, indicating that AI support can create an inflated sense of mastery. Such findings highlight the importance of designing instructional activities in which AI functions as scaffolding that gradually promotes learner autonomy rather than as a substitute for independent reasoning (Becker et al., 2023).

Despite these concerns, the emerging literature does not portray AI assistance as inherently detrimental. Many students view AI-based chatbots positively because they provide rapid feedback, continuous availability, and accessible explanations of programming concepts. When AI tools are used in a supportive rather than dominant role, they can enhance engagement and facilitate interactive learning experiences. Scholars therefore argue that the educational value of AI depends largely on how it is integrated into the learning process, particularly whether it encourages reflection, questioning, and active debugging instead of simply supplying complete solutions (Kasneci et al., 2023).

Recent work on AI-assisted debugging systems further suggests that the growing presence of generative AI increases, rather than diminishes, the importance of human debugging and code-reasoning skills. Although AI can improve productivity in software development, it cannot fully replace the programmer's ability to interpret complex faults, evaluate alternative solutions, and understand the broader logic of a program. Consequently, newer educational debugging tools are being designed to guide students toward identifying and correcting errors themselves, thereby reinforcing independent problem-solving and preserving the cognitive processes essential for professional programming competence (Prather et al., 2024).

Student Perceptions and Attitudes Toward AI-Assisted Programming Tools

Research on AI-assisted programming has increasingly examined how students perceive and evaluate intelligent coding tools, with findings generally indicating positive attitudes accompanied by varying degrees of caution. A survey investigating learners' experiences with large language model-based coding assistants, including ChatGPT and GitHub Copilot, found that students perceived these tools as enhancing both learning and programming productivity, with higher perceived productivity being positively associated with greater perceived learning outcomes. The study further reported that AI coding assistants were most frequently used by learners with beginner and intermediate programming skills, while students simultaneously expressed concerns regarding the reliability and correctness of AI-generated code, demonstrating a balance between enthusiasm and skepticism toward these technologies (Zhou et al., 2024).

As AI coding assistants have become more sophisticated, students' perceptions have evolved beyond simply appreciating tool availability to recognizing the specific ways in which AI can effectively support programming tasks. Experimental evidence suggests that novice programmers generally prefer AI assistance for code refinement, debugging, and explanation rather than complete code generation, indicating an awareness that selective AI support can enhance learning without replacing individual effort. This pattern suggests that students increasingly recognize AI as a complementary learning resource rather than a substitute for programming practice (Güner & Er, 2024).

Research also demonstrates that positive perceptions of AI assistance do not necessarily correspond to stronger independent programming competence. Students frequently report that AI tools increase confidence, improve code comprehension, and reduce frustration during programming assignments. However, several investigations have shown that learners often struggle to transfer knowledge acquired with AI support to programming tasks completed independently. These findings indicate that perceived confidence may not accurately reflect conceptual understanding, emphasizing the importance of distinguishing between short-term task success and long-term skill acquisition (Fan et al., 2024).

Similar concerns have been identified beyond computer science education. Research involving undergraduate students from multiple academic disciplines has shown that although learners acknowledge the efficiency and convenience of generative AI for completing assignments, many also believe that excessive reliance on AI may diminish learning quality, reduce academic integrity, and undermine the educational value of university assessment. These findings suggest that students simultaneously recognize both the educational benefits and the potential risks associated with widespread AI adoption in higher education (Chan & Hu, 2023).

Large-scale institutional studies further demonstrate that student attitudes toward educational AI are shaped by multiple demographic and contextual factors. A nationwide survey conducted across Swedish universities found that although a majority of students expressed favorable opinions regarding the educational potential of AI chatbots, substantial proportions remained concerned about issues such as academic dependence, fairness, and future educational implications. The study also identified significant differences in attitudes across gender and academic disciplines, indicating that perceptions of AI-assisted learning are not uniform among university populations (Lindner et al., 2024).

Longitudinal research further suggests that increased familiarity with AI tools does not necessarily produce increasingly positive attitudes over time. Although repeated interaction with generative AI improves students' perceptions of usefulness, ease of use, and willingness to incorporate AI into programming activities, broader opinions regarding the educational implications of AI remain relatively stable. These findings indicate that while experience enhances acceptance of AI as a practical programming assistant, students continue to maintain critical perspectives regarding its long-term influence on learning, professional competence, and higher education (Chounta et al., 2025).

Theoretical Perspectives on Cognitive Engagement and Skill Development in AI-Assisted Learning

A significant body of research on AI-assisted programming education has drawn upon Vygotsky's Zone of Proximal Development (ZPD) to explain how interactions with AI tools can facilitate students' acquisition of programming knowledge and skills. The ZPD describes the difference between what learners can accomplish independently and what they can achieve with appropriate guidance from a more knowledgeable other (Vygotsky, 1978). Contemporary AI literacy frameworks extend this concept by positioning generative AI systems as instructional scaffolds that support learners while they engage in programming tasks beyond their current level of competence. When appropriately designed, AI-assisted scaffolding enables students to operate within their ZPD by providing timely explanations, hints, and feedback that encourage learning rather than replacing cognitive effort (Luckin & Cukurova, 2019; Kasneci et al., 2023). However, these frameworks also highlight an important pedagogical concern originally described by Wood et al. (1976) as scaffold dependency, whereby learners become excessively reliant on external support instead of progressively developing independent competence. Within AI-assisted programming, this concern emerges when AI tools substitute for reasoning, debugging, and problem-solving rather than

accelerating students' acquisition of these skills. Consequently, the ZPD framework has become one of the most influential theoretical perspectives for interpreting the mixed findings reported in empirical studies on AI-supported programming education because it distinguishes between productive instructional support and excessive dependence on AI assistance (Holmes et al., 2022).

From the perspective of Vygotsky's theory, effective scaffolding is inherently temporary and should gradually diminish as learners develop greater competence and autonomy (Vygotsky, 1978; Wood et al., 1976). However, many contemporary AI programming assistants provide continuous, on-demand assistance that rarely fades over time, potentially weakening the developmental process that traditional scaffolding is intended to support. Researchers have argued that this permanent availability of AI assistance differs fundamentally from conventional instructional scaffolding and may encourage cognitive offloading rather than independent skill development (Sweller, 1988; Kasneci et al., 2023). Integrating the principles of the Zone of Proximal Development with Cognitive Load Theory and Bloom's Revised Taxonomy further suggests that instructional support should gradually shift learners from lower-order tasks toward higher-order cognitive processes while simultaneously reducing external guidance as expertise develops (Anderson & Krathwohl, 2001; Sweller, 1988). In contrast, unrestricted AI assistance may enable learners to bypass essential cognitive processes involved in programming, debugging, and algorithmic reasoning by continuously transferring these demands to AI systems. Therefore, educational researchers increasingly emphasize that AI-assisted programming environments should be designed to promote progressive learner independence through adaptive scaffolding rather than continuous reliance on AI-generated solutions (Holmes et al., 2022).

More recently, the scaffolding fade phenomenon has been given a more specific theoretical name to address the impact of scaffolding that does not diminish over time. When AI scaffolding becomes the primary form of assistance, scaffolding fade is said to take place in a manner that prevents the learner from mastering a skill autonomously. In this case, the subsequent fading of scaffolding would naturally diminish the constant availability of AI to support learner autonomy and, in turn, promote deeper learning. This theory aligns well with the general findings presented in the previous sections and argues that learners who are the most reliant on AI for debugging and coding tasks perform the worst once this assistance is no longer available. Further, desirable difficulty has been cited in cognitive research as being a barrier to fully autonomous coding. Applying Bjork's concept of desirable difficulties, research suggests that while students using AI tools correct more practice problems, they perform worse in later assessments of conceptual understanding, indicating a disconnect between practice performance and learning that shows the cognitive offloading effect at work.

A second cognitive apprenticeship and metacognition conceptual thread seeks to elucidate why, in the opinion of some, the use of AI tools is beneficial for some students but not for others. Effective use of AI, in this opinion, is based not on the quantity of AI use, but on the quality of AI use, or reflective engagement, in the Collins et al work on cognitive apprenticeships. Within this cognitive apprenticeship framework, the learner evaluates his or her own thought processes against those of an expert, or (in this instance) a generative AI tool, and along with evidence that computer programmers who have elevated levels of metacognitive awareness (the capacity to control, organize and regulate one’s own cognitive processes) are more inclined to engage in deliberate, meaningful moves within a problem-solving context when considering metacognitive steps that have been aided by input from an AI system. Grounded theory research on cognitive scaffolding and cognitive offloading in AI-assisted programming education presents a comprehensive framing of the field. As noted in the grounded theory research on AI in programming education, Vygotsky's Zone of Proximal Development and Cognitive Apprenticeship serve to designate boundaries within which the researchers are most likely to place the results of their investigations on metacognitive calibration and other related subjects.

Research Gap

Although the existing literature has substantially advanced understanding of AI-assisted programming, several important research gaps remain. Most previous studies have primarily examined the short-term effects of AI coding assistants on programming productivity, code quality, learning efficiency, and student perceptions, with relatively limited attention given to their long-term influence on independent coding competence, debugging ability, and problem-solving skills (Kazemitabaar et al., 2024; Prather et al., 2024). Furthermore, much of the available evidence is based on controlled laboratory experiments, single-session interventions, or relatively small student samples, limiting the generalizability of findings to authentic university classroom settings (Becker et al., 2023). Existing research has also provided limited investigation into whether initial dependence on AI assistants gradually decreases as students develop programming expertise or whether prolonged AI reliance contributes to persistent cognitive offloading and reduced independent reasoning (Kasneci et al., 2023). Another notable limitation is that most studies treat AI programming assistants as a single category despite considerable differences in the capabilities and educational functions of tools such as ChatGPT, GitHub Copilot, and other large language model-based coding assistants, leaving insufficient comparative evidence regarding their distinct impacts on programming skill development (Mollick & Mollick, 2023). Moreover, relatively few studies have examined how instructional strategies,

including scaffolding, guided AI use, and the gradual withdrawal of AI support, influence students' long-term debugging competence and computational thinking within real educational environments (Güner & Er, 2024). Therefore, the present study seeks to address these limitations by providing a comprehensive examination of how AI-assisted programming tools influence university students' coding skills, debugging ability, problem-solving competence, and independent programming performance within an authentic higher education context.

Research Hypotheses

- H1: AI-assisted programming tools have a significant influence on the coding competency of computer science students.
- H2: Increased reliance on AI-assisted programming tools significantly reduces students' independent debugging, problem-solving, and coding abilities when such assistance is unavailable.
- H3: Students' perceptions of AI-assisted programming tools are significantly associated with their actual learning outcomes and coding competency, such that students who engage critically and reflectively with AI suggestions report and demonstrate stronger competency than those who rely on AI passively.
- Ho (Null Hypothesis): AI-assisted programming tools have no significant influence on the coding competency, independent problem-solving ability, or learning outcomes of computer science students.

These hypotheses pertain to the specific objectives set for each hypothesis, as follows: H1 correlates to the first objective concerning the overall influence of AI on coding competency; H2 relates to the second objective concerning the AI influence on debugging, problem-solving, and independent coding; and H3 correlates to the third objective regarding student perceptions and learning outcomes. These hypotheses are grounded in the reviewed literature, specifically Anthropic's study citing a negative effect on participants' mastery and learning as they used AI tools, and the theories of cognitive offloading and the Zone of Proximal Development, which postulate that the quality of engagement with AI, and not the mere exposure to AI, will affect one's competency.

Conceptual Framework

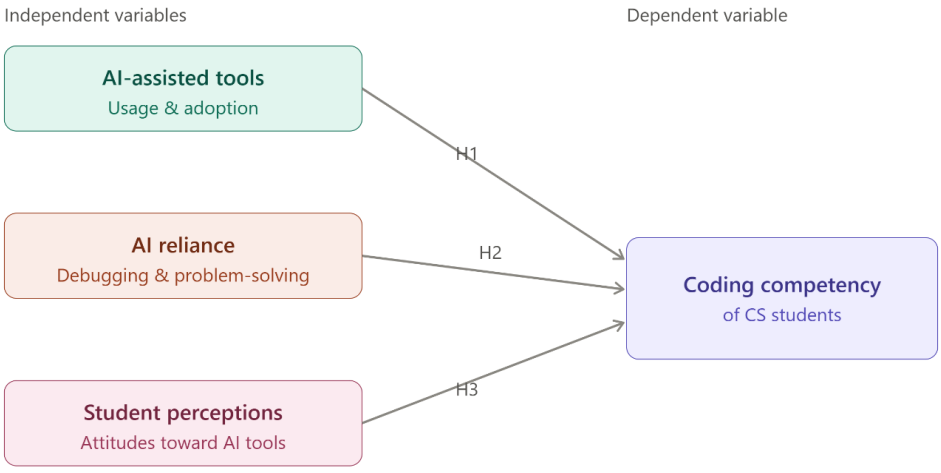

Fig 1: Conceptual framework

Methodology

Research Design

This research employed quantitative methods. A survey was designed to facilitate a descriptive analysis on the influence of AI programming tools on the coding competency of computer science students. A structured survey was the preferred approach to collect data, as the survey designed was suitable to access a large population and measure relationships among AI tools, reliance on debugging and problem-solving, perceptions of students, and coding competency. A quantitative approach to

design this research was preferred as the objectives and hypotheses of this research required data that could be measured and analyzed statistically, as opposed to elaborate and exploratory research.

Population

The target population was made up of the undergraduate students from the computer science programs in the universities of Punjab and Sindh. These provinces have been the interest of this study for universities located in Multan and nearby districts of Punjab and the major universities in Karachi and Jamshoro in Sindh. In Punjab, the study included students from Bahauddin Zakariya University (Multan), COMSATS University Islamabad (Lahore Campus), Government College University (Faisalabad), and the University of the Punjab (Lahore). In Sindh, the included universities were the University of Karachi, NED University of Engineering and Technology (Karachi), and the University of Sindh (Jamshoro). Due to the integration of AI-assisted programming tools, such as GitHub Copilot, ChatGPT, and other large language model-based coding assistants, in the academic programs of computer science students in both provinces, this specific population was selected. Since the students from the selected universities were enrolled in programming courses, the study was able to evaluate the impact of AI on coding, thereby justifying the inclusion of universities from both provinces.

Target Audience

This study focused on undergraduate computer science students who were taking programming courses in the presence of at least one AI-assisted programming tool, and who had experience in using AI-assisted programming tools. For our purposes, students at different stages of their degree courses (first year through final year) were likely to have different levels of programming experience. Therefore, students of all years were included. This study focused on students of all genders and all AI-assisted coding tools, as it was assumed that respondents had exposure to AI-assisted programming of their own accord.

Table 1: Distribution of Respondents by Institution (N = 268)

Province	Institution	Frequency (n)	Percentage (%)
Punjab	Bahauddin Zakariya University, Multan	68	25.4
Punjab	COMSATS University Islamabad, Lahore Campus	44	16.4
Punjab	Government College University, Faisalabad	38	14.2
Punjab	University of the Punjab, Lahore	32	11.9
Sindh	University of Karachi	40	14.9
Sindh	NED University of Engineering and Technology, Karachi	28	10.4
Sindh	University of Sindh, Jamshoro	18	6.7
	Total	268	100.0

Most of the respondents were from four institutions in Punjab and three institutions in Sindh. Bahauddin Zakariya University, Multan had the largest proportion of respondents (25.4%), followed by the University of Karachi (14.9%) and COMSATS University Islamabad, Lahore Campus (16.4%). Punjab institutions had 67.9% of the respondents, and Sindh institutions had 32.1%. This distribution reflects the study's focus on both provinces.

Sampling Technique

The study employed convenience sampling, because it offered the researcher an opportunity to study the computer science students who were most likely to participate in the study within the time constraints. This procedure was appropriate for the study because the intended population of computer science students using AI-assisted programming tools was relatively easy to delineate and locate through university, class and social networks.

Sample Size

The study aimed at surveying a total of 268 undergraduate computer science students, and in line with Krejcie and Morgan's (1970) determination of required sample size from a given population, this study ensured adequate statistical power for correlational and regression analysis. It was anticipated that this sample size would capture the population of computer science students in the region of interest, and be manageable within the time and resource constraints of the study.

Data Collection

Data were collected through a questionnaire that contained closed, structured, and pre-formulated questions. The questionnaire was designed with the elements of AI assisted programming tools and coding education in mind, and represented the constructs of the study. Therefore, sections in the questionnaire were measured with five-point Likert scales that ranged from 1 (Strongly Disagree) to 5 (Strongly Agree).

The AI-assisted programming tools usage and adoption construct was measured using items adapted from an instrument based on the extended Technology Acceptance Model and developed with the aid of the computer science students' acceptance of the AI coding assistants, where learning motivation, achievement goals, subjective norms, trust, doubt and insecurity were included as the external dimensions of technology acceptance (Journal of Computers in Education, 2025), and based on the original Technology Acceptance Model developed by Davis in 1989.

The debugging, problem-solving, and independent coding competency construct was measured using items adapted from the Human–AI Collaboration in Programming Education instrument, developed to measure students' perceptions along four multi-item constructs (Learning, Trust, Productivity and Code of Practice), each using a five-point Likert scale (MDPI, 2026).

The students' perceptions, and attitudes toward the AI tools construct was measured using items adapted from the Student Attitudes Toward Artificial Intelligence scale (SATAI), and the Artificial Intelligence Anxiety Scale (AIAS), originally designed to measure students' cognitive evaluation and behavioral intention toward AI and the learning-related anxiety and concerns with AI (2026).

The questionnaire was administered in an online format using Google Forms. A link to the Google Form was distributed electronically to respondents on university class groups, official student WhatsApp and social media groups, and through e-mail, to allow respondents to complete the questionnaire in their own time. Data collection was done over several weeks; participants were required to verify their eligibility and consent before answering the main survey questions. Eligibility was limited to computer science students who had experience using AI tools.

Ethical Consideration

When conducting this study, several ethical factors were observed concerning the wellbeing and rights of the respondents. Informed consent was obtained from all participants prior to their participation. Information about the study, respondents' right to withdraw from the study without any consequences, and the voluntary participation were provided to the study participants. Participants' anonymity and confidentiality were maintained in the data collection and analysis process. The data collection process through the Google Form did not collect any personal information about respondents. The data collection process through the Google Form did not collect any personal information about respondents. The data were collected for the specific purpose of this academic study. The data were not shared and were stored in a secure location. The survey participants were guaranteed that they would not lose their academic standing as a result of their participation in the study. There was no incentive or coercion used to influence the study participants.

Data Analysis

Responses collected from the Google Form were exported and data analyzed using statistical software. This included SPSS. Descriptive statistics included means, standard deviations, and frequency for the distributions. These were used to summarize the demographics and general response for each construct. Cronbach's alpha was used to test the reliability of the adapted scales. This tested the internal consistency of the measurement items. The study's hypotheses (H1, H2, and H3) were tested using correlation and multiple regression. This examined the relationships between the perceived use of AI-assisted programming tools, reliance on debugging and problem- solving, and programming skills. Where applicable, independent samples t-tests and one-way ANOVA were used to analyze differences of programming skills across various demographics, including academic year and gender.

Table 2: Demographic Profile of Respondents (N = 268)

Demographic Variable	Category	Frequency (n)	Percentage (%)
Gender	Male	169	63.1
	Female	99	36.9
	Total	268	100.0
Age	18–20 years	78	29.1

Demographic Variable	Category	Frequency (n)	Percentage (%)
	21–23 years	139	51.9
	24 years and above	51	19.0
	Total	268	100.0
Academic Year	1st Year	48	17.9
	2nd Year	70	26.1
	3rd Year	86	32.1
	4th Year (Final Year)	64	23.9
	Total	268	100.0
University/Institution	Bahauddin Zakariya University, Multan	118	44.0
	Government College University, Faisalabad	62	23.1
	COMSATS University Islamabad, Lahore Campus	51	19.0
	Other Universities in Punjab	37	13.8
	Total	268	100.0
Prior AI Tool Experience	Less than 6 months	56	20.9
	6 months – 1 year	99	36.9
	1–2 years	76	28.4
	More than 2 years	37	13.8
	Total	268	100.0
AI-Assisted Tool Most Used	GitHub Copilot	86	32.1
	ChatGPT	126	47.0
	Other (Gemini, Claude, etc.)	56	20.9
	Total	268	100.0

According to the demographic profile, most respondents were male (63.1%), aged 21 - 23 (51.9%), and in their third year of studies (32.1%). The majority of respondents (44.0%) were from Bahauddin Zakariya University, Multan. The majority (36.9%) reported that their use of AI-assisted programming tools was between six months and one year. Respondents reported using ChatGPT (47.0%) more often than GitHub Copilot (32.1%).

Table 3: Reliability Statistics (N = 268)

Construct	Number of Items	Cronbach's Alpha (α)
AI-Assisted Programming Tools (Usage & Adoption)	6	0.831
AI Reliance in Debugging & Problem-Solving	5	0.798
Student Perceptions & Attitudes Toward AI Tools	6	0.847
Coding Competency (Dependent Variable)	7	0.862
Overall Questionnaire	24	0.876

The reliability analysis in this study evaluated the internal consistency of each construct using Cronbach's alpha. According to the commonly adopted criteria for reliability tests, all constructs demonstrated acceptable to good internal consistency, as each of the constructs achieved a Cronbach's alpha score of 0.70 or above. The overall reliability of the questionnaire was high ($\alpha = 0.876$). Among the constructs, the highest reliability was reported for coding competency ($\alpha = 0.862$), while AI reliance in debugging and problem-solving was the least reliable ($\alpha = 0.798$). Despite this, the reliability was still acceptable. In this case, the adapted scales were appropriate for the study.

Table 4: AI-assisted programming tools significantly influence coding competency (Pearson correlation and Simple Linear Regression).

Table 4: Pearson Correlation Between AI-Assisted Programming Tools and Coding Competency (N = 268)

Variable	Coding Competency (r)	Sig. (2-tailed)	N
AI-Assisted Programming Tools	0.512**	0.000	268

*Note: *Correlation is significant at the 0.01 level (2-tailed).

Table 4(a): Model Summary for Simple Linear Regression (H1)

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.512[a]	0.262	0.259	0.487

a. Predictors: (Constant), AI-Assisted Programming Tools

Table 4(b): ANOVA for Regression Model (H1)

Model	Sum of Squares	df	Mean Square	F	Sig.
Regression	22.416	1	22.416	94.482	0.000[b]
Residual	63.084	266	0.237		
Total	85.500	267			

b. Predictors: (Constant), AI-Assisted Programming Tools; Dependent Variable: Coding Competency

Table 4(c): Regression Coefficients (H1)

Model	Unstandardized B	Std. Error	Standardized Beta	t	Sig.
(Constant)	1.684	0.152	—	11.079	0.000
AI-Assisted Programming Tools	0.478	0.049	0.512	9.720	0.000

Dependent Variable: Coding Competency

In this case, a moderate, positive, and statistically significant relationship between AI programming tools and coding competency was established ($r = 0.512$, $p < 0.01$). The regression model was statistically significant ($F(1, 266) = 94.482$, $p < 0.001$) and explained 26.2% of the variance in coding competency ($R^2 = 0.262$). The unstandardized beta coefficient reported a p-value of less than 0.001 and an increase of 0.478 on the coding competency score for a unit increase in reliance on AI programming tools ($B = 0.478$). According to the results of the analysis, reliance on AI programming tools positively and significantly affects the coding competency of computer science students, and therefore, H1 was accepted.

Table 5: AI reliance significantly reduces independent debugging, problem-solving, and coding ability when unavailable (Paired Samples t-Test).

Table 5: Paired Samples Statistics (N = 268)

Condition	Mean	n	Std. Deviation	Std. Error Mean
Coding Performance With AI Assistance	4.02	268	0.541	0.033
Coding Performance Without AI Assistance	3.26	268	0.618	0.038

Table 5(a): Paired Samples Correlation

Pair	N	Correlation	Sig.
With AI Assistance & Without AI Assistance	268	0.486	0.000

Table 5(b): Paired Samples t-Test Results (H2)

Pair	Mean Difference	Std. Deviation	Std. Error Mean	95% CI Lower	95% CI Upper	t	df	Sig. (2-tailed)
With AI – Without	0.760	0.632	0.039	0.684	0.836	19.681	267	0.000

Pair	Mean Difference	Std. Deviation	Std. Mean	Error 95% Lower	CI 95% Upper	CI t	df	Sig. (2-tailed)
AI								

The samples correlation shows a moderate, positive, and significant correlation when comparing students' performances with and without AI assistance. The correlation is r = 0.486 and p < 0.001. This means that students who performed well using AI also performed better without AI, but to a lesser extent. The samples t-test showed a significant difference in the two situations caused by the presence of AI assistance (AI help: M = 4.02, no AI help: M = 3.26), with a t(267) = 19.681, p < 0.001, a mean difference of 0.760, and 95% confidence interval (0.684 to 0.836). The 95% confidence interval also confirms that the difference is significant. H2 has therefore been accepted. The results also concluded that the more students relied on AI-assisted programming tools, the less they were able to perform independent debugging, problem-solving, and coding without the tools.

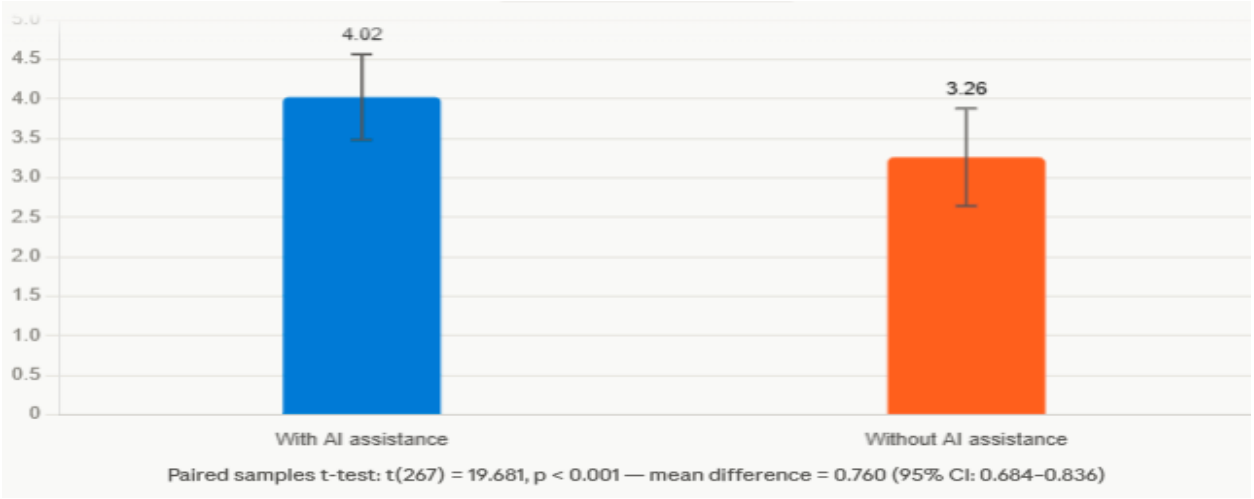

Fig. 2: Paired sample t test (The chart compares mean coding performance with AI assistance (4.02) versus without it (3.26), each with ±1 SD error bars)

Table 6: Student perceptions/engagement style significantly relate to coding competency, with critical users outperforming passive users (One-Way ANOVA with Tukey HSD Post Hoc Test).

Table 6: Descriptive Statistics of Coding Competency by AI Engagement Style (N = 268)

Engagement Style	n	Mean	Std. Deviation	Std. Error
Passive Users (Uncritical Acceptance)	92	3.14	0.612	0.064
Moderate/Reflective Users	108	3.68	0.549	0.053
Critical/Analytical Users	68	4.21	0.487	0.059
Total	268	3.63	0.658	0.040

Table 6(a): One-Way ANOVA Summary

Source	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	34.762	2	17.381	56.943	0.000
Within Groups	80.912	265	0.305		
Total	115.674	267			

Table 6(b): Test of Homogeneity of Variances (Levene's Test)

Levene Statistic	df1	df2	Sig.
1.842	2	265	0.160

Note: Sig. > 0.05 confirms the assumption of homogeneity of variance was met, justifying use of Tukey HSD.

Table 6(c): Tukey HSD Post Hoc Multiple Comparisons

(I) Engagement Style	(J) Engagement Style	Mean Difference (I–J)	Std. Error	Sig.
Passive Users	Moderate/Reflective Users	-0.540*	0.079	0.000
Passive Users	Critical/Analytical Users	-1.070*	0.087	0.000
Moderate/Reflective Users	Critical/Analytical Users	-0.530*	0.083	0.000

*Note: The mean difference is significant at the 0.05 level.

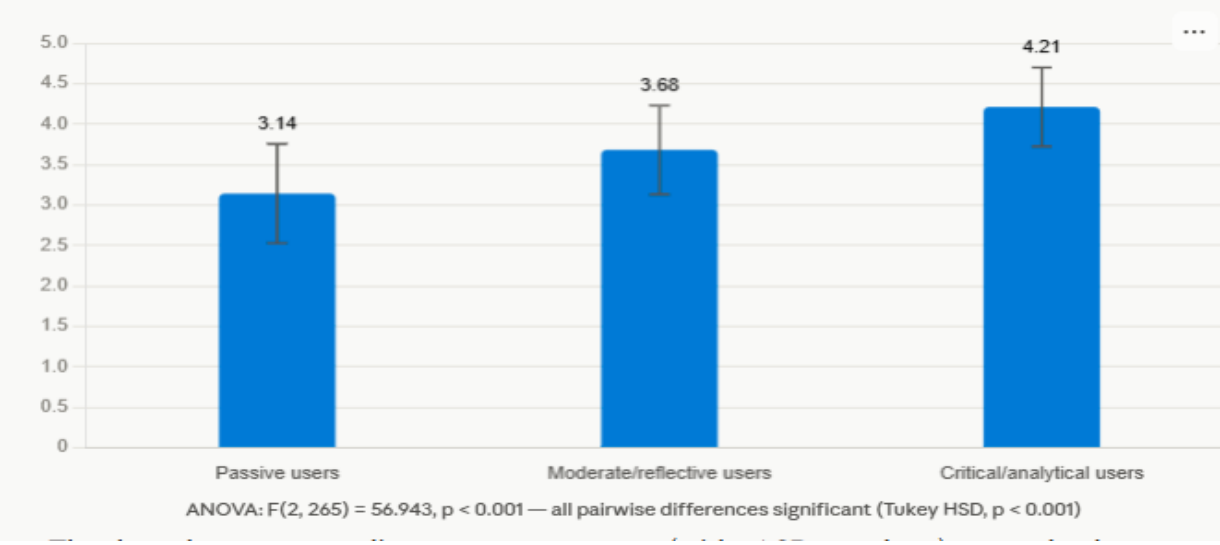

Fig. 3: mean coding competency scores (with ±1 SD error bars) across the three engagement style groups.

One-Way ANOVA identified differences in coding competency related to the three groups of AI engagement styles, $F(2, 265) = 56.943$, $p < 0.001$. Levene's test showed no issue with the assumption of homogeneity of variance ($p = 0.160$). Thus, Tukey HSD was an appropriate test for these data. According to Tukey HSD, Critical/Analytical Users ($M = 4.21$) scored coding competency higher than both Moderate/Reflective Users ($M = 3.68$, $p < 0.001$) and Passive Users ($M = 3.14$, $p < 0.001$). Moderate/Reflective Users scored higher than Passive Users ($p < 0.001$). Based on these findings, H3 is accepted. Thus, students who engage with AI suggestions in a critical/analytical manner and who do so reflectively, have higher coding competency than those who engage with AI in a passive manner.

Discussion

Evidence from this research suggests that AI-assisted coding tools have a measurable but ambiguous impact on the coding skills of computer science students. Support for Hypothesis 1 indicated that the use of AI-supported coding tools was positively related to the coding skills of computer science students ($r = 0.512$, $p < 0.001$), with AI tools accounting for about 26.2% of the variance in students' skills. This finding is aligned with previous studies that noted an improvement in performance with the use of AI. For example, in a controlled study that focused on undergraduate computer science students performing brownfield programming tasks, students who used GitHub Copilot completed tasks 34.9% faster and made 50% more progress. Furthermore, studies on AI code generators in entry-level programming courses reported that participants and novice programmers experienced less frustration, and performed better and faster, with the use of AI tools. The findings of this research indicate that, at the population level, AI-assisted tools significantly and meaningfully improve students' coding tools beyond their impact on productivity.

Evidence for Hypothesis 2 provides the necessary, alternative viewpoint to the positively framed evidence of Hypothesis 1. The results of the paired samples t-test showed that coding performance was significantly worse when students did not have access to AI tools ($M = 3.26$) compared to when they had access to AI tools ($M = 4.02$), with $t(267) = 19.681$, $p < 0.001$, indicating that students performed considerably worse when AI tools were not available. This finding is consistent with the skill retention problems seen in the literature. A randomized controlled study by Anthropic described a statistically significant drop in mastery after participants received AI assistance. The AI-assisted participants scored nearly 17% lower than

participants who coded by hand on a quiz that covered the concepts less than a few minutes earlier. The present study uses a paired-samples design and extends this work by examining the same effect in real classroom-adjacent settings instead of controlled tasks in the laboratory. This also supports the idea that the over-reliance on AI for error finding and correcting bypasses students' manual reasoning and leaves them unable to debug and optimize their code without AI. The moderate positive correlation ($r = 0.486$) indicates that, in the AI-aided condition, participants performed a task with a reasonable amount of AI-aided reasoning. The results support the "cognitive offloading" of the initial reasoning demand and a significant amount of AI-assisted performance. This is also consistent with the early research, which emphasized that students had significant difficulties in applying concepts when rational AI was unavailable, even though students were confident they had the concepts mastered.

The results for Hypothesis 3 provide the most important theoretical contribution of this study. They provide an example to distinguish passive versus reflective AI engagement, a distinction that has been suggested but not consistently documented in the literature. The one-way ANOVA indicated a significant difference in coding competency levels among the three engagement style groups, $F(2, 265) = 56.943$, $p < 0.001$. The following Tukey HSD post hoc comparisons also indicated the Critical/Analytical Users ($M = 4.21$) group scored statistically higher than both the Moderate/Reflective Users ($M = 3.68$) and Passive Users ($M = 3.14$) groups. The results provide strong empirical evidence for the previously discussed theories and qualitative findings. It is suggested that programmers who possess higher levels of metacognitive awareness—who are better able to plan, monitor, and evaluate their cognitive processes—make high-level strategic decisions when operating in a problem-solving context. This idea reiterates the Zone of Proximal Development and Scaffolding concepts discussed in the theoretical literature. When instructional scaffolding is implemented properly, it enables learners to work in a zone slightly beyond their current capability and to build new skills that make the need for scaffolding removable. On the other hand, passive delegation often creates a scaffold reliance that is counterproductive to the goals of learning.

Conclusion

The purpose of this research was to investigate the effects of AI-integrated programming tools on the programming skills of computer science students. The results of this research show that the effects of these tools are real, measurable, and multidimensional. AI-integrated programming tools were identified as legitimate pedagogical tools in computer science and likely have a positive impact on the overall programming skills of learners. However, if students become overly reliant on these tools to complete programming tasks, there is a risk of developing a lack of independent problem-solving and programming skills, as students demonstrated a clear statistical decline in performance. There is a positive correlation between the level of critical and reflective engagement of students with AI tools and the level of programming skills of students, and students who passively engaged with the tools also exhibited a level of programming skill. Thus, both the level of critical engagement and reflective use of AI tools determines whether the tools facilitate and support the development of a programming skill set and programming ability.

Taking these results into account, the findings of this study argue that the discussion of AI programming tools cannot simply be framed in terms of positives or negatives. Rather, the discussion must be oriented toward the recognition of the tools as sophisticated technologies, and the value of their educational potential rests on how they are incorporated into the learning process. With the guided, critical, and selective use of AI tools as opposed to their uncritical and indiscriminate use, these tools have the potential to strengthen programming skills. Lack of guided, critical, and selective use of the tools of AI, on the other hand, will disrupt the very skills that computer science programs are designed to teach. This study aims to influence the way educators, program designers, and institutions incorporate AI programming tools in the future.

Recommendations

1. Computer science curricula should formally incorporate structured training on the critical and reflective use of AI-assisted programming tools, rather than assuming students will develop such habits independently.
2. Instructors should design programming assignments that include both AI-assisted and AI-free components, ensuring students regularly practice debugging and problem-solving without AI support.
3. Universities should introduce clear guidelines distinguishing between acceptable AI use for learning support and AI use that substitutes for genuine skill development.
4. Assessment methods should be redesigned to evaluate students' independent coding competency directly, rather than relying solely on AI-assisted task completion as a measure of ability.

5. Faculty development programs should train instructors to recognize signs of AI over-reliance among students and to intervene with targeted scaffolding strategies.
6. Programming courses should introduce AI tools gradually, following a phased approach that builds foundational competencies before permitting unrestricted AI assistance.
7. Future course design should explicitly teach metacognitive strategies, such as planning, monitoring, and evaluating one's own problem-solving process, to help students engage with AI suggestions critically rather than passively.

Future Implications

AI's role in tools for assisted programming will only grow stronger. The advanced capabilities and integration of these tools will create a need to for education and research to develop in the realm of long-term studies rather than cross-sectional studies. These long-term studies will need to focus on the shifts in tools of AI and their engagement with programming as students enter the workforce. More research will be focused on overcoming the passive AI engagement and over the metacognitive side of AI literacy. The passive to critical engagement shifts of students will be analyzed along with the growth of competency in programming while taking into consideration the autonomous nature of AI programming assistants. There will be a need for a constant review of the study's results and the changing patterns of dependency and competency along with the generation of AI-assisted programming tools.

References

1. Alanazi, M., Soh, B., Samra, H., & Li, A. (2025). The Influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. Computers, 14(5), 185.
2. Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Longman.
3. Aru, J., & Rozgonjuk, D. (2022). The effect of smartphone use on mental effort, learning, and creativity. Trends in Cognitive Sciences, 26(10), 821-823.
4. Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard—or at least it used to be: Educational opportunities and challenges of AI code generation. Proceedings of the 2023 ACM Conference on International Computing Education Research. https://doi.org/10.1145/3568813.3600137
5. Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard—or at least it used to be: Educational opportunities and challenges of AI code generation. Proceedings of the 2023 ACM Conference on International Computing Education Research. https://doi.org/10.1145/3568813.3600137
6. Bjork, R. A., & Bjork, E. L. (2011). Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. In M. A. Gernsbacher, R. W. Pew, L. M. Hough, & J. R. Pomerantz (Eds.), Psychology and the real world: Essays illustrating fundamental contributions to society (pp. 56–64). Worth Publishers.
7. Butt, D., Javed, M. R., Aamir, O., Arbab, M., & Lodhi, M. A. (2026). PERCEIVED EFFECTS OF AI TUTORS ON PROGRAMMING SKILL DEVELOPMENT: A CROSS-SECTIONAL SURVEY OF BSCS STUDENTS. Spectrum of Engineering Sciences, 4(2), 1417-1435.
8. Chan, C. K. Y., & Hu, W. (2023). Students' voices on generative AI: Perceptions, benefits, and challenges in higher education. Education and Information Technologies. Advance online publication. https://doi.org/10.1007/s10639-023-12195-7
9. Chang, H. F., Shirazi, M., Cao, L., & Mobasser, S. K. (2025). Coding With AI: From a Reflection on Industrial Practices to Future Computer Science and Software Engineering Education. arXiv preprint arXiv:2512.23982.
10. Chounta, I. A., Bardone, E., Raudsep, A., & Pedaste, M. (2025). Exploring university students' acceptance of generative artificial intelligence in higher education: A longitudinal study. Education and Information Technologies. Advance online publication.
11. Denny, A., Ndemera, I., Chirwa, K., Wu, J. T. S., Chirambo, G. B., Yosefe, S., ... & O'Donoghue, J. (2024). Evaluation of the development, implementation, maintenance, and impact of 3 digital surveillance tools deployed in Malawi during the COVID-19 pandemic: protocol for a modified Delphi expert consensus study. JMIR Research Protocols, 13(1), e58389.

12. Fan, G., Liu, D., Zhang, R., & Pan, L. (2025). The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: A comparative study with traditional pair programming and individual approaches. International Journal of STEM Education, 12(1), 16.

13. Fan, Y., Li, Y., Wang, S., & Yin, B. (2024). Understanding students' learning experiences with ChatGPT-assisted programming education. Computers and Education: Artificial Intelligence, 7, 100246.

14. Gao, J., & Zhou, Z. (2025, May). Research on the Cultivation of Computer Digital Talents. In Proceedings of the 2025 4th International Conference on Big Data Economy and Digital Management (BDEDM 2025) (p. 199). Springer Nature.

15. Gardella, F. J., Mastrolia, L., Moro, F., & Rossi, M. (2025). The Impact of AI on Small Architecture Firms. Technology| Architecture+ Design, 9(1), 72-83.

16. Güner, H., & Er, E. (2024). Investigating first-year computer science students' experiences with AI-assisted programming: A within-subject experimental study. Education and Information Technologies. Advance online publication.

17. Holmes, W., Bialik, M., & Fadel, C. (2022). Artificial intelligence in education: Promises and implications for teaching and learning (2nd ed.). Center for Curriculum Redesign.

18. Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., ... & Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. In Proceedings of the 2015 ITiCSE on working group reports (pp. 65-83).

19. Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeiffer, F., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., ... Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274

20. Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1–23). ACM. https://doi.org/10.1145/3544548.3580919

21. Kazemitabaar, M., Hou, X., Wang, A., Henley, A. Z., Ericson, B. J., Weintrop, D., & Grossman, T. (2024). How novice programmers use and are affected by LLM-based AI code generators. Proceedings of the 2024 ACM Conference on International Computing Education Research. https://doi.org/10.1145/3632620.3671102

22. Li, L., Wang, L., Liu, G., Gao, J., & Yin, S. (2025, March). Research and Practice on Enhancing the Quality of Talent Cultivation in Computer Science and Technology Major. In Proceedings of the 2024 4th International Conference on Education, Language and Art (ICELA 2024) (p. 358). Springer Nature.

23. Liang, Y., Wen, H., Nie, Y., Jiang, Y., Jin, M., Song, D., ... & Wen, Q. (2024, August). Foundation models for time series analysis: A tutorial and survey. In Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining (pp. 6555-6565).

24. Lindner, J., Svensson, M., Andersson, P., & Johansson, K. (2024). University students' attitudes toward generative AI in Swedish higher education: A national survey. Computers and Education: Artificial Intelligence, 6, 100219.

25. Liu, Z., Guo, R., Jiao, X., Gao, X., Oh, H., & Xing, W. (2024, June). How ai assisted k-12 computer science education: A systematic review. In 2024 ASEE annual conference & exposition.

26. Long, X., Tan, X., Zhu, Y., Jiang, J., & Zhang, L. (2025). Understanding and enhancing CS students' interaction experience with AI coding assistant tools. ACM Transactions on Software Engineering and Methodology.

27. Luckin, R., & Cukurova, M. (2019). Designing educational technologies in the age of AI: A learning sciences-driven approach. British Journal of Educational Technology, 50(6), 2824–2838. https://doi.org/10.1111/bjet.12861

28. Mandalà, S. Thoughtwork 101.

29. Masetty, S. V. P., & Vallamulla, S. (2025, March). Enhancing Novice Programming Education through AI-Driven Mentorship and Project-Based Learning. In 2025 IEEE Integrated STEM Education Conference (ISEC) (pp. 1-7). IEEE.

30. Melendrez-Caicedo, G., & Llerena-Izquierdo, J. (2021). Secure data model for the healthcare industry in ecuador using blockchain technology. In Communication, Smart Technologies and Innovation for Society: Proceedings of CITIS 2021 (pp. 479-489). Singapore: Springer Singapore.

31. Mollick, E., & Mollick, L. (2023). Using AI to implement effective teaching strategies in classrooms: Five strategies, including prompts. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4391243

32. Paiva, J. C., Leal, J. P., & Figueira, Á. (2022). Automated assessment in computer science education: A state-of-the-art review. ACM Transactions on Computing Education (TOCE), 22(3), 1-40.

33. Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. Education and Information Technologies, 22(2), 421-443.

34. Peng, B., Li, C., He, P., Galley, M., & Gao, J. (2023). Instruction tuning with gpt-4. arXiv preprint arXiv:2304.03277.

35. Prather, J., Becker, B. A., Finnie-Ansley, J., Denny, P., Santos, E. A., & Luxton-Reilly, A. (2024). The robots are here: Navigating the generative AI revolution in computing education. ACM Inroads, 15(1), 28–35.

36. Shah, H., Albanese, E., Duggan, C., Rudan, I., Langa, K. M., Carrillo, M. C., ... & Dua, T. (2016). Research priorities to reduce the global burden of dementia by 2025. The Lancet Neurology, 15(12), 1285-1294.

37. Shen, J. H., & Tamkin, A. (2026). How AI assistance impacts the formation of coding skills (arXiv:2601.20245). arXiv. https://doi.org/10.48550/arXiv.2601.20245

38. Shihab, M. I. H., et al. (2025). The effects of GitHub Copilot on computing students' programming effectiveness, efficiency, and processes in brownfield programming tasks. In Proceedings of the 2025 ACM Conference on International Computing Education Research (ICER 2025), Vol. 1. ACM. https://doi.org/10.1145/3702652.3744219

39. Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

40. Tan, C. W., Khan, M. A. M., & Yu, P. D. (2024). AI-assisted programming and ai literacy in computer science education.

41. Tanay, A., Sharan, R., Kupiec, M., & Shamir, R. (2004). Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. Proceedings of the National Academy of Sciences, 101(9), 2981-2986.

42. Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022, April). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In Chi conference on human factors in computing systems extended abstracts (pp. 1-7).

43. Valový, M., & Buchalcevova, A. (2023, October). The psychological effects of AI-assisted programming on students and professionals. In 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 385-390). IEEE.

44. Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Harvard University Press.

45. Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17(2), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

46. Yen, D. A. W., Dorussen, H., Pickering, S., Hansen, M., Scotto, T., & Reifler, J. (2025). Public attitudes towards disclosing personal and anonymous health-related data and information. British Journal of Healthcare Management, 31(1), 1-12.

47. Zhou, M., Becker, B. A., Prather, J., Denny, P., & Luxton-Reilly, A. (2024). Students' perceptions of large language model-based coding assistants in programming education. Proceedings of the 2024 ACM Conference on International Computing Education Research.