Vibe Coding and AI-assisted Programming in Undergraduate Computer Education: A Systematic Review of Pedagogical Benefits and Foundational Skill Risks

Pongrapee Kaewsaiha[1], Chutima Klaysung[2] and Srisarin Norasedsophon[3]

[1,2,3] Suan Sunandha Rajabhat University, Thailand

Emails: pongrapee.ka@ssru.ac.th[1], chutima.kl@ssru.ac.th[2], srisarin.no@ssru.ac.th[3]

Abstract

This study aimed to identify and synthesize how vibe coding and AI-assisted programming are defined, conceptualized, and operationalized in undergraduate computer education programs; examine and categorize the reported pedagogical benefits of AI-assisted coding approaches; and identify the risks and challenges associated with them. A systematic review was conducted across major academic databases, including Scopus, IEEE Xplore, ACM Digital Library, and ERIC, covering the period from 2023 to 2025. A total of 19 studies met the predefined inclusion criteria for analysis. The synthesized findings reveal that GitHub Copilot and ChatGPT were often used as an AI pair programmer and an AI conversational assistant, respectively. The primary pedagogical benefits are the acceleration of rapid prototyping, the reduction of cognitive load associated with syntactic errors, and increased student motivation. Students often transition from low-level to higher-level coding, focusing on problem definition and critical code review. Conversely, significant foundational skill risks were identified, including a potential decline in core debugging skills, reduced deep understanding of algorithms and data structures, and an increased reliance on generated code, which may hinder mastery of fundamental programming constructs. The review also highlighted the emerging challenge of teaching proper prompt engineering and adequate code verification as essential new skills.

Keywords: AI-assisted Programming, Computer Education, Undergraduate, Vibe Coding

1. Introduction

The rapid integration of Large Language Models (LLMs) into the software development workflow has given rise to “Vibe Coding”—the practice of coding using natural language prompts. The term was introduced in February 2025 by Andrej Karpathy, an AI researcher, on his X post (Edwards, 2025). Vibe coding allows developers to quickly program without being bound by software development rules or formal procedures. The programmer specifies the desired software features and lets the AI prototype them. This practice has shifted away from traditional AI-assisted programming, where a specific tool, such as GitHub Copilot or Tabnine, suggests the following line of code based on what the programmer has already written (Glance, 2025).

The paradigm shift mandates a re-evaluation of pedagogical strategies in undergraduate curricula, as many university students have reportedly integrated artificial intelligence (AI) into their classroom assignments (Cavazos et al., 2024), ranging from simple data collection to completing writing tasks (Goulart et al., 2024; Thitivesa et al., 2025). However, allowing

students to use AI in classrooms might alter traditional teaching dynamics and raise ethical concerns (Salloum, 2024). In 2025, The Scottish Sun reported over 1,000 AI-related cheating incidents with students using chatbots like ChatGPT to complete coursework and assessments, a 700% increase from the past year (Rodger, 2025). The New York Post reported that 97% of Gen-Z students are using AI to write essays, do homework, and even get into college (Olander, 2025)

The Joint Council for Qualifications (2025) stated that copying/paraphrasing AI-generated content does not reflect the student's own thinking and should be considered AI misuse. In computer engineering education, where AI and coding are closely linked, allowing students to code with AI might hinder their coding skills, especially in introductory programming courses (Lau & Guo, 2023). This might raise the question: "If programmers use AI for coding, then who will code the AI?"

This systematic review examines the current state of research on integrating vibe coding practices into undergraduate curricula, focusing on reported pedagogical benefits and potential risks to foundational programming skills.

2. Research Objectives

This research study was aimed:

1. To synthesize existing definitions and conceptual frameworks of vibe coding and AI-assisted programming in undergraduate computer education.
2. To evaluate the pedagogical benefits and learning outcomes associated with vibe coding and AI-assisted programming practices.
3. To identify risks to foundational programming skills linked to these approaches.
4. To analyze instructional and curricular strategies proposed to balance between increasing productivity and lowering risk.

3. Methodology

To systematically examine empirical and conceptual literature on vibe coding / AI-assisted programming practices in undergraduate computer engineering (and closely related computing disciplines), this study followed the PRISMA-style systematic review plan (Page et al., 2021) described as follows:

Inclusion Criteria

Population: Undergraduate students in computer engineering, computer science, software engineering, or closely related fields

Intervention/Phenomenon: Explicit mention of vibe coding or AI-assisted programming (or similar terms), ChatGPT or Copilot-like tools

Outcomes: Programming performance, conceptual understanding of programming language, debugging ability, risks, and ethical concerns

Study types: Peer-reviewed sources that present empirical studies (experimental, quasi-experimental, survey, mixed-methods), systematic reviews, or meta-analyses

Publication years: 2023–present, as ChatGPT was publicly released as a free research preview at the end of 2022 (Walsh, 2025)

Language: English

Exclusion Criteria

Studies that focus solely on technical LLM performance benchmarks with no educational implications, studies that include K–12 education only, and non-peer-reviewed sources were excluded from the analyses (except for discussion).

Information Sources

This systematic review includes publications from the following sources (databases): Scopus, IEEE Xplore, ACM Digital Library, and ERIC.

Search Strategy

Three structured query blocks were run per database. Block-A looked up "vibe coding" as an emerging term. Block-B searched for AI-assisted programming in education, a core evidence preceding vibe coding. Block-C focused on specific tools, such as ChatGPT and GitHub Copilot.

Search Block-A:

("vibe coding" OR "vibe-coding")

AND

(education OR curriculum OR student* OR undergraduate*)

Search Block-B:

("AI-assisted programming" OR "AI assisted coding" OR "AI coding assistant"

OR "code generation" OR "generative AI")

AND

(student* OR undergraduate* OR "higher education")

AND

(programming OR software OR coding)

Search Block C:

("GitHub Copilot" OR "ChatGPT" OR "LLM" OR "large language model")

AND

(programming OR coding)

AND

(education OR student* OR learning)

Data Extraction Framework

A synthesis matrix was created with the following fields:

Bibliographic info: Author(s), year

Discipline: Computer Eng. (CE), Computer Science (CS), Software Eng. (SE), …

Subject: Intro. programming, web development (greenfield/brownfield), data science, …

AI tool: Copilot, ChatGPT, custom LLM

Study design: Experiment, survey, systematic review, mixed, …

Sample size: n

Reported benefits: Speed, motivation, confidence, design focus, …

Reported risks: Surface-level learning, debugging weakness, security, …

Foundational skills affected: Syntax, logic, debugging, code reading, …

Mitigation strategies: Scaffolding, AI restrictions, assessments, …

Key conclusions: Author claims

4. Result

According to the eligibility criteria, a total of 19 peer-reviewed sources were included in the analysis. Table 1 shows a synthesis of nine research papers that cover all aspects designed for the analysis.

Table 1 Synthesis Matrix

Bib. Info	Discipline	Subject	AI Tool	Study Design	Sample Size	Reported Benefits	Reported Risks	Foundational Skills Affected	Mitigation Strategies	Key Conclusions
Chen et al., 2024	Eng. Science	Intro. (Python)	ChatGPT	Quasi-exp.	49 students	Significant improvement in critical thinking and problem-solving skills.	Potential over-reliance hindering higher-order thinking skills (HOTS).	Logic reasoning, debugging, and problem-solving.	Personalized reflective reports based on logged coding errors.	Error-based reflective feedback effectively fosters cognitive engagement over simple generation.
Güner & Er, 2025	Statistics & CEIT (Mixed)	Intro. (Python)	ChatGPT	Mixed-method	158 students	Enhanced engagement, adaptive learning, and instant conceptual feedback.	Accuracy concerns, over-reliance, and plagiarism risks.	Syntax, logic, and problem-solving.	Hands-on prompt training and lab guides with sample prompts.	Training shifts students from "AI-Reliant" to "AI-Collaborative" profiles.

Bib. Info	Discipline	Subject	AI Tool	Study Design	Sample Size	Reported Benefits	Reported Risks	Foundational Skills Affected	Mitigation Strategies	Key Conclusions
Haruto et al., 2025	CS & CE	Intro. (Python, Java)	ChatGPT, Codex, Copilot	Sys. Review	25 studies	On-demand support, personalized tutoring, and reduced instructor workload.	Academic integrity, equity gaps, and "help-seeking loops".	Algorithmic thinking, syntax recall, and problem-solving.	Hybrid learning models and restricting AI in high-stakes assessments.	Pedagogical value is context-sensitive and depends on deliberate human-AI design.
He & Li, 2025	Digital Media Tech	Intro. (C)	Hierarchical Guided LLM	Quasi-exp.	80 students	Improved academic performance, self-efficacy, and learning interest.	Excessive reliance and weakening of independent exploration.	Algorithm understanding, syntax error correction.	Refined hierarchical guiding strategy (Q&A, debugging, interpretation).	AI assistance effectively replaces traditional "waiting mode" in human pairing.
Omeh et al., 2025	CS Education	Intro. (Java)	Google Gemini	Quasi-exp.	62 students	Significant improvement in programming skills with large effect size.	Over-dependence may reduce cognitive effort and critical thinking.	Algorithm development, structured programming, debugging.	Problem-Based Learning (PBL) integration and peer review.	High-ability students benefit more from AI integration than low-ability peers.
Roy et al., 2025	CS & SE	Capstone & Advanced SE	Copilot, GPT-4, Lucidchart AI	Case Study	84 students	Accelerated debugging (50%) and expanded project	Hallucinations in UML/Architecture and context-blind logic errors.	Architectural design, white-box testing, and prompt	"Right-to-Left" (R-L) approach: AI used only after core	AI increases confidence in tool adaptation but requires

Bib. Info	Discipline	Subject	AI Tool	Study Design	Sample Size	Reported Benefits	Reported Risks	Foundational Skills Affected	Mitigation Strategies	Key Conclusions
						scope (more features).		engineering.	proficiency is established.	foundational knowledge to verify output.
Schreiber & Tippe, 2025	SE	Security (Python, JS, TS)	ChatGPT, Copilot, Tabnine, Code Whisperer	Large-scale static analysis	7,703 files	Effective documentation generation and high productivity in routine code.	Language-specific vulnerabilities (SQLi, Code Injection) and hallucinations.	Secure coding practices and software maintainability.	Language-specific security controls and manual verification of AI outputs.	87.9% of code is safe, but Python code shows higher vulnerability rates across all tools.
Shihab et al., 2025	SE	Web Dev. (Brown field)	GitHub Copilot	Cont. & exp. groups	10 students	34.9% faster task completion and 50% more solution progress.	Diminished understanding of "how" or "why" code works; cognitive disengagement.	Manual code writing, web searching, and code integration.	Modeling and scaffolding critical reflection in GenAI activities.	Copilot restructures workflow into a "prompt → read response → implement" cycle.
Wang et al., 2025	Edu. Info. Tech.	ACM Comp et. (Stack / DSA)	Custom AI-Agent (LLM-based)	Quasi-exp.	45 students	Lower mental effort (cognitive load), higher motivation and self-efficacy.	Detailed responses may hinder active knowledge absorption if too comprehensive.	Data structure construction (stacks) and code readability.	Task difficulty adjustment and collaborative role rotation.	AI-Agents facilitate smoother co-regulation and higher quality collaborative outcomes.

The key findings from all reviewed sources can be divided into four main points, which can be summarized as follows:

The Landscape of AI Tools in Coding

AI tools are capable of suggesting solutions, generating code, explaining errors, and accelerating development workflows (Čerňanský et al., 2025). Some tools primarily act as AI-first coders that rely on natural language input, while others are often used as pair-programming agents in which programmers write code first. Key tools mentioned include:

GitHub Copilot is often described as an AI pair programmer (Čerňanský et al., 2025; Fan et al., 2025; Park et al., 2025) that provides inline code autocompletion and suggestions based on context (Čerňanský et al., 2025; Park et al., 2025). It leverages OpenAI Codex, a language model trained on public Git repository datasets (Haruto et al., 2025; Johanyák et al., 2023; Park et al., 2025). Its features extend to a conversational interface (Copilot Chat) for answering questions and suggesting unit tests. Copilot is highly integrated with Integrated Development Environments (IDEs) like Visual Studio Code (Čerňanský et al., 2025).

ChatGPT is a conversational agent based on LLMs. It excels at tasks like defining assignments, correcting errors, explaining code, and providing interactive help (Johanyák et al., 2023; Güner & Er, 2025). It is widely adopted in programming contexts, even dominating the attribution found in public GitHub repositories analyzed in one study (Schreiber & Tippe, 2025).

Cursor is an AI-first code editor that deeply embeds LLMs into the development experience, offering context-aware editing and high-level features such as code explanation and refactoring directly in the UI (Čerňanský et al., 2025).

The most widely adopted use case is code generation and completion, where LLMs predict the next lines or blocks of code, often accelerated by comments written in natural language (Čerňanský et al., 2025). Beyond code generation, these tools offer essential support for learning and productivity:

Code debugging and error correction: AI tools can identify and explain syntax or logical errors in code, suggesting fixes to reduce student frustration (Adhikari et al., 2024; Chen et al., 2024; Güner & Er, 2025; He & Li, 2025; Johanyák et al., 2023). For students, this translates to real-time assistance, helping quickly locate and correct errors (Cao, 2025; Wang et al., 2025).

Code explanation: LLMs excel at transforming complex code into accessible natural language explanations or summaries, aiding comprehension and debugging (Adhikari et al., 2024; Čerňanský et al., 2025; Güner & Er, 2025; He & Li, 2025; Johanyák et al., 2023). This capacity is valued, particularly for helping students understand code they did not write themselves (Adhikari et al., 2024; Bakharia & Abdi, 2024).

Testing and quality assurance: LLMs can generate unit and integration tests, reducing the manual effort required for verification (Blasquez, 2025; Čerňanský et al., 2025; Roy et al., 2025).

Impact on Programming Performance and Efficiency

The integration of these assistive tools significantly enhances efficiency and productivity, suggesting that AI can expedite coding workflows by acting as a supplementary knowledge source.

Productivity gains: Shihab et al. (2025) found that professionals using Copilot reported productivity increases (24% average, 44% for junior developers). Students using Copilot completed coding tasks significantly faster and made more progress in implementing solutions than when working without AI.

Reduced cognitive load: AI assistance reduces the extraneous cognitive load associated with non-essential tasks such as syntax recall and boilerplate code generation (Cao, 2025; Park et al., 2025; Shihab et al., 2025). This freeing up of memory allows students to focus more on the inherent complexity of the task (Shihab et al., 2025).

Enhanced learning outcomes: AI assistance can effectively improve students' programming performance, self-efficacy, and programming interest (He & Li, 2025). Research comparing AI-assisted pair programming with human-human pairing and individual approaches found that AI-assisted groups demonstrated superior performance and higher intrinsic motivation than individual programming (Fan et al., 2025).

Challenges: Dependency, Quality, and Ethical Concerns

Despite the benefits, relying on generative AI tools presents critical pedagogical and technical challenges that require careful management (Haruto et al., 2025; Park et al., 2025).

Over-reliance and skill atrophy: Excessive dependence on AI tools may hinder the development of independent problem-solving skills and deeper conceptual understanding (Chen et al., 2024; Güner & Er, 2025; Haruto et al., 2025; Park et al., 2025; Shihab et al., 2025). Students often adopt a passive learning approach, seeking quick answers rather than engaging in deeper reasoning, potentially leading to long-term skill deficits (Fan et al., 2025; Haruto et al., 2025).

Code quality and inaccuracy (hallucinations): AI-generated code is not guaranteed to be correct (Park et al., 2025; Schreiber & Tippe, 2025). LLMs are prone to "hallucinations"—generating believable but incorrect or irrelevant content, including non-existent libraries or subtle logic flaws (Čerňanský et al., 2025; Bakharia & Abdi, 2024; Blasquez, 2025; Roy et al., 2025). This necessitates continuous critical human oversight (Čerňanský et al., 2025; Karnalim et al., 2024; Schreiber & Tippe, 2025).

Security vulnerabilities: A large-scale analysis (Schreiber & Tippe, 2025) revealed that although 87.9% of AI-generated code lacks identifiable Common Weakness Enumeration (CWE)- mapped vulnerabilities, AI systems still generate code containing widely known and severe security weaknesses (e.g., SQL Injection, OS Command Injection). Python code, in particular, consistently exhibited higher vulnerability rates across tools compared to JavaScript and TypeScript.

Effective Integration and Pedagogical Strategies

To leverage AI tools effectively, educators must adopt strategies that shift the focus from manual coding to critical thinking and refinement (Bakharia & Abdi, 2024).

Fostering critical evaluation: Students must be trained to critically evaluate and verify AI-generated outputs, treating them as first drafts rather than final solutions (Blasquez , 2025; Čerňanský et al., 2025; Roy et al., 2025). Industry practitioners emphasize the need for students to assess the correctness, applicability, and quality of AI-generated code (Ocay & Rodrigo, 2025).

Teaching prompt engineering: A critical new skill is "prompt engineering," or the ability to ask precise, contextually relevant questions to optimize AI output (Güner & Er, 2025; Ocay & Rodrigo, 2025). This strategy is essential for iteratively refining AI-generated code (Roy et al., 2025).

Scaffolding and guidance: The integration should be structured, perhaps using a hierarchical guidance strategy that provides support (Q&A, error correction) without immediately yielding complete solutions (He & Li, 2025).

Reflection and feedback: AI tools like ChatGPT can generate personalized reflective reports based on a student's coding errors, promoting reflection and enhancing higher-order thinking skills (HOTS), such as critical thinking and problem-solving (Chen et al., 2024).

Balancing AI use: Educators should balance AI assistance with traditional learning activities and may use AI-free assessments to ensure students develop independent problem-solving skills and retain core concepts without becoming dependent on AI (Johanyák et al., 2023; Park et al., 2025).

5. Conclusion

This study examined the growing integration of AI into undergraduate computer education through a systematic review. Two different coding approaches can be concluded from 19 peer-reviewed sources. With vibe coding, or AI-first coding, students identify prompts in natural language; the AI agent then generates the code, and students debug or modify it. AI-assisted programming lets students code first, and the AI assistant suggests the next line.

Studies indicate that AI-assisted problem-based learning and pair programming can significantly enhance student performance, increase intrinsic motivation, and reduce learning anxiety. However, these tools also pose challenges, such as a potential decline in fundamental syntax mastery and an increased risk of academic plagiarism through code obfuscation.

Beyond the classroom, the emergence of coding agents and assistants like GitHub Copilot and Cursor is shifting the development workflow toward "vibe coding," where creators focus on high-level intent rather than manual implementation. While these autonomous tools enable non-experts to build complex applications, they still require careful evaluation to mitigate model hallucinations and navigate intricate technical infrastructures. Together, the texts highlight a shift toward collaborative human-AI workflows that prioritize problem-solving and logic over traditional manual coding skills.

References

Adhikari, B., Dhakal, S., & Jha, A. (2024). Maximizing Student Engagement in Coding Education with Explanatory AI. In *2024 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). IEEE. https://doi.org/10.1109/FIE61694.2024.10893108

Bakharia, A., & Abdi, S. (2024). Shaping Programming and Data Science Education: Insights from GenAI Technical Book Trends. In *2024 IEEE International Conference on Advanced Learning Technologies (ICALT)* (pp. 116-120). IEEE. https://doi.org/10.1109/ICALT61570.2024.00040

Blasquez, I. (2025). Developing critical thinking with AI coding assistants: An educational experience focusing on testing and legacy code. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2025)* (pp. 1–7). Association for Computing Machinery. https://doi.org/10.1145/3724363.3729050

Cao, H. (2025). Experiences and insights gained from AI-assisted programming instruction in higher education. In *Proceedings of the 2025 International Conference on AI-enabled Education (AIEE 2025)* (pp. 1–8). Association for Computing Machinery. https://doi.org/10.1145/3768421.3768449

Cavazos, J. T., Hauck, K. A., Baskin, H. M., & Bain, C. M. (2024). ChatGPT Goes to College: Exploring Student Perspectives on Artificial Intelligence in the Classroom. *Teaching of Psychology, 52*(3), 357–368. https://doi.org/10.1177/00986283241268829

Čerňanský, M., Hafner, P., & Dirgová Luptáková, I. (2025). LLM tools for programming. In *2025 International Conference on Emerging eLearning Technologies and Applications (ICETA)* (pp. 139–44). IEEE. https://doi.org/10.1109/ICETA67772.2025.11280153

Chen, P.-H., Huang, Y.-M., Wu, T.-T., & Lee, H.-Y. (2024). Coding error-based reflective learning with ChatGPT: A study on higher-order thinking skills in programming education. In *Proceedings of the 6th International Conference on Modern Educational Technology (ICMET 2024)* (pp. 1–7). Association for Computing Machinery. https://doi.org/10.1145/3729434.3729435

Edwards, B. (2025). Will the future of software development run on vibes? *Ars Technica*. https://arstechnica.com/ai/2025/03/is-vibe-coding-with-ai-gnarly-or-reckless-maybe-some-of-both/

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*(16). https://doi.org/10.1186/s40594-025-00537-3

Glance. (2025). What's the Difference Between Vibe Coding and AI-Assisted Programming? *Expert Guide Series*. https://thisisglance.com/learning-centre/whats-the-difference-between-vibe-coding-and-traditional-ai-assisted-programming

Goulart, L., Matte, M. L., Mendoza, A., Alvarado, L., & Veloso, I. (2024). AI or student writing? Analyzing the situational and linguistic characteristics of undergraduate student writing and AI-generated assignments. *Journal of Second Language Writing, 66*, 101160. https://doi.org/10.1016/j.jslw.2024.101160

Güner, H., & Er, E. (2025). AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies, 30*(9), 12681–12707. https://doi.org/10.1007/s10639-025-13337-7

Haruto, S., Nnadi, L. C., & Yutaka, W. (2025). Systematic review of large language model applications in programming education. In H. Fujita et al. (Eds.), *New Trends in Intelligent Software Methodologies, Tools and Techniques* (pp. 83–95). IOS Press. https://doi.org/10.3233/FAIA250512

He, H., & Li, X. (2025). A Comparative Study on the Effectiveness of Generative AI Tools and Pair Programming in College Students' Programming Learning. In *2025 International Conference on Distance Education and Learning (ICDEL)* (pp. 224-228). IEEE. https://doi.org/10.1109/ICDEL65868.2025.11193454

Johanyák, Z. C., Cserkó, J., & Pásztor, A. (2023). AI-assisted university programming education in practice. In *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)* (pp. 185–186). IEEE. https://doi.org/10.1109/CSEET58097.2023.00039

Joint Council for Qualifications. (2025). *AI use in assessments: Your role in protecting the integrity of qualifications*. https://www.jcq.org.uk/exams-office/malpractice/artificial-intelligence/

Karnalim, O., Toba, H., Johan, M. C., Handoyo, E. D., Setiawan, Y. D., & Luwia, J. A. (2023). Plagiarism and AI assistance misuse in web programming: Unfair benefits and characteristics. In *2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)* (pp. 1-5). IEEE. https://doi.org/10.1109/TALE56641.2023.10398397

Lau, S., & Guo, P. (2023). From" Ban it till we understand it" to" Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1* (pp. 106-121). https://doi.org/10.1145/3568813.3600138

Ocay, A. B., & Rodrigo, M. M. T. (2025). Gen AI in computing education: A gap analysis of pedagogical practices and industry expectations. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2025)*. Association for Computing Machinery. https://doi.org/10.1145/3724389.3730793

Olander, S. (2025). Startling 97% of Gen Z students are using AI to write essays, do homework — and even get into college. New York Post. https://nypost.com/2025/07/05/lifestyle/gen-z-turns-to-ai-for-homework-essays-and-college-apps

Omeh, C. B., Ayanwale, M. A., Mnguni, L. E., & Olelewe, C. J. (2025). Fostering programming skill and critical thinking through AI-assisted PBL integration. *Journal of New Approaches in Educational Research, 14*(22). https://doi.org/10.1007/s44322-025-00041-0

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372. https://doi.org/10.1136/bmj.n71

Park, S. M., Ho, M., Lin, M. P. C., & Ryoo, J. (2025). Evaluating the Impact of Assistive AI Tools on Learning Outcomes and Ethical Considerations in Programming Education. In *2025 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1-10). IEEE. https://doi.org/10.1109/EDUCON62633.2025.11016517

Rodger, P. (2025). Scots university students caught using AI to cheat over 1,000 times – and it's 'only the tip of the iceberg'. *The Scottish Sun*. https://www.thescottishsun.co.uk/news/14422897/scotland-university-students-artificial-intelligence-tip-iceberg/

Roy, N., Horielko, O., & Omojokun, O. (2025, October). Integrating AI Tools in Advanced Computer Science Curricula: A Case Study of Course Redesign. In *Proceedings of the ACM Global on Computing Education Conference 2025 Vol 1* (pp. 92-98). Association for Computing Machinery. https://doi.org/10.1145/3736181.3747130

Salloum, S. A. (2024). AI perils in Education: Exploring ethical concerns. In *Studies in big data* (pp. 669–675). https://doi.org/10.1007/978-3-031-52280-2_43

Schreiber, M., & Tippe, P. (2025). Security vulnerabilities in AI-Generated Code: A Large-Scale Analysis of Public GitHub repositories. In *Lecture notes in computer science* (pp. 153–172). https://doi.org/10.1007/978-981-95-3537-8_9

Shihab, M. I. H., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y., & Mulanda, B. W. (2025). The effects of GitHub Copilot on computing students' programming effectiveness, efficiency, and processes in brownfield coding tasks. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1 (ICER 2025 Vol. 1)*. Association for Computing Machinery. https://doi.org/10.1145/3702652.3744219

Thitivesa, D., Phongam, P., & Siraphatada, Y. (2025). AI in a Language Classroom: Pre-Service Teachers' Perceptions of Automated Writing Assistance Tools. In *International Conference on Sustainable Education and Teaching (ICSET 2025)* (pp. 30-35).

Walsh, S. (2025). *Timeline Of ChatGPT Updates & Key Events*. Search Engine Journal. https://www.searchenginejournal.com/history-of-chatgpt-timeline/488370/

Wang, H., Wang, C., Chen, Z., Liu, F., Bao, C., & Xu, X. (2025). Impact of AI-agent-supported collaborative learning on the learning outcomes of University programming courses. *Education and Information Technologies, 30*, 17717–17749. https://doi.org/10.1007/s10639-025-13487-8