Available on: http://gjetr.ep-journals.org/ E-ISSN: 3051-3782, P-ISSN: 3051-3774

Global Journal of Engineering and Technology Review

Volume: 02 Issue: 07 July 2026

Generative AI in Programming Education: A Review of Learning Support, Feedback, and Assessment Challenges

Dang Thi Kim Giao

Ho Chi Minh City University of Foreign Languages – Information Technology, Vietnam

Published Date: July 11, 2026 **Article DOI : 10.65150/EP-gjetr/V2E7/2026-03**

ABSTRACT: Generative AI is increasingly used in programming education. Tools such as ChatGPT, GPT-4, and GitHub Copilot can generate code, explain concepts, support debugging, and provide feedback. This review examines recent studies on generative AI in programming education, focusing on learning support, feedback generation, and assessment challenges. The findings show that generative AI can help students understand concepts, fix errors, and receive quick guidance. It can also help teachers prepare materials and reduce grading workload. However, it may also lead to overreliance, weak critical thinking, shallow understanding, and problems in homework, grading, plagiarism detection, and assessment validity. The review concludes that generative AI should be used as a guided support tool, not as a replacement for programming practice or teacher guidance.

KEYWORDS: Generative AI; programming education; ChatGPT; feedback; assessment; academic integrity

INTRODUCTION

Generative AI is becoming an important topic in programming education. Tools such as ChatGPT, GPT-4, and GitHub Copilot can generate code, explain programming concepts, support debugging, and provide feedback. These tools create new opportunities for teaching and learning, but they also raise concerns about learning quality, feedback, assessment, and academic integrity (Baig et al., 2026; Garcia, 2025; Manorat et al., 2025).

Programming education is not only about producing correct code. Students also need to understand logic, trace errors, test solutions, and explain how their programs work. These skills are especially important for novice programmers. However, beginners often struggle with syntax, algorithmic thinking, debugging, and problem-solving. In this context, generative AI may provide useful support. It can give examples, explain errors, and guide students when they cannot continue their work (Groothuijsen et al., 2024; Zviel-Girshin, 2024).

At the same time, this support creates a new problem. Generative AI can help students learn, but it can also do too much of the work for them. Students may copy AI-generated code without understanding the solution. They may also accept AI suggestions without checking their correctness. This is risky in introductory programming, where students still need to build basic programming logic and independent problem-solving skills (Prather et al., 2024; Xue et al., 2024; Yang et al., 2025).

Generative AI also changes the role of feedback and assessment. AI tools can provide quick feedback and may reduce teachers' grading workload. However, AI-generated feedback may be incomplete, inaccurate, or difficult for students to evaluate. AI-generated code also makes traditional homework and programming assignments less reliable as evidence of student ability (Azaiz et al., 2024; Bernik et al., 2025; Tang et al., 2023).

Several recent reviews have discussed ChatGPT, generative AI, or AI-assisted tools in programming education. These reviews provide broad views of research trends, applications, benefits, and challenges (Baig et al., 2026; Garcia, 2025; Manorat et al., 2025; Nathaniel et al., 2025b). However, many existing reviews discuss the topic broadly. Less attention has been given to the connection between learning support, feedback generation, and assessment validity in programming courses. Therefore, the key question is not only whether generative AI should be used, but how it should be guided, checked, and assessed in programming education.

This review examines recent studies on generative AI in programming education. It focuses on three areas: learning support, feedback generation, and assessment challenges. It aims to clarify how generative AI supports students and teachers, what risks it creates, and what conditions are needed for responsible use in programming courses.

The review is guided by three research questions:

RQ1. How does generative AI support students in learning programming?

RQ2. How is generative AI used to provide feedback in programming education?

RQ3. What assessment challenges are created by generative AI in programming courses?

2. LITERATURE REVIEW AND METHODS

2.1. Scope of the review

This paper uses a focused literature review approach. It does not aim to provide a full systematic review of all studies on artificial intelligence in programming education. Instead, it reviews recent studies that are directly related to generative AI, programming education, feedback, and assessment.

Several recent reviews have examined this area from broader perspectives. Garcia (2025) reviewed the use of ChatGPT in teaching and learning computer programming. Baig et al. (2026) provided a scoping review of ChatGPT applications in computer programming education. Manorat et al. (2025) reviewed artificial intelligence and machine learning in programming education across course design, classroom implementation, assessment and feedback, and performance monitoring. Nathaniel et al. (2025b) reviewed the integration of generative AI in programming education. These studies provide a useful foundation for the present review.

However, the present review has a narrower focus. It does not cover all uses of AI in programming education. It focuses on three practical areas that are central to current programming courses: learning support, feedback generation, and assessment challenges. This focus was chosen because these areas directly affect how students learn, how teachers support learning, and how programming ability is assessed.

2.2. Selection and Analysis of Studies

The reviewed studies were selected based on three criteria. First, they had to focus on programming education, computer science education, or introductory programming. Second, they had to discuss generative AI, ChatGPT, large language models, or AI coding tools. Third, they had to address at least one of the three themes of this review: learning support, feedback generation, or assessment challenges.

The studies were identified through the author's literature search and major academic databases, including Scopus, Web of Science, IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, and Taylor & Francis. Most selected studies were published between 2023 and 2026, reflecting the recent development of generative AI tools in programming education.

Recent journal articles were prioritized. Conference papers were also included when they were highly relevant to novice programmers, AI-generated feedback, or academic integrity. Studies that focused mainly on professional software engineering, AI agents, technical code generation, or general AI in education were not used as core sources unless they were directly relevant to programming education.

The selected studies were read and grouped into three themes. The first theme examines how generative AI supports students in learning programming. The second theme examines how generative AI provides feedback and supports teachers. The third theme examines assessment problems caused by AI-generated code. This thematic structure is used in the Findings and Discussion section.

Review existing research to build context for the study, identify what is known, and point out knowledge gaps that your study intends to fill.

3. MATERIALS AND METHODS

This section discusses the reviewed studies through three themes: learning support, feedback generation, and assessment challenges. These themes show how generative AI affects both learning and teaching in programming education.

3.1. Learning Support in Programming Education

Generative AI can support students during programming tasks in several ways. It can explain programming concepts, generate code examples, identify errors, suggest corrections, and support debugging. These functions are useful for novice programmers because they often struggle with syntax, logic, and problem-solving.

Several studies show that students use generative AI as a learning assistant. Groothuijsen et al. (2024) found that students used ChatGPT for error checking, debugging, code explanation, conceptual understanding, code generation, and code optimization. Zviel-Girshin (2024) also reported that novice programmers used AI tools for creating comments, correcting bugs, and seeking information. These findings suggest that AI tools can help students continue working when they are stuck.

AI tools may also support learning by giving quick explanations. Students can ask for examples, request simpler explanations, or check why their code does not work. This kind of support can make programming tasks less frustrating. Park (2025) showed that code suggestions and explanations can support programming learning. Scholl and Kiesler (2024) also showed that novice programmers used ChatGPT in different ways while solving programming exercises.

However, the value of generative AI depends on how students use it. Some students use AI suggestions critically. They compare answers, revise code, and try to understand the logic. Others may use AI mainly to get complete solutions. Güner and Er (2025) showed that students had different interaction profiles with ChatGPT, ranging from AI-reliant code generators to more independent or collaborative users. This suggests that the same tool can support or weaken learning depending on student behavior.

The main risk is overreliance. Prather et al. (2024) argued that generative AI may widen the gap between stronger and weaker novice programmers. Stronger students may use AI to improve their work, while weaker students may accept AI output without enough understanding. Xue et al. (2024) found limited evidence that ChatGPT improved learning performance in introductory programming. Yang et al. (2025) also reported that ChatGPT support did not always lead to better learning outcomes. These findings show that access to AI support does not automatically lead to deeper learning.

Overall, generative AI can support programming learning, but it should not replace students' own thinking. It is most useful when students use it to understand errors, test ideas, and revise their code. It is less useful when students use it only to receive ready-made answers. Therefore, programming courses need clear guidance on how students should use AI tools during learning tasks.

3.2. Feedback Generation and Teacher Support

Generative AI can also support feedback in programming education. Feedback is important because students need to know what is wrong in their code and how they can improve it. In programming courses, this process can take much time, especially in large classes. AI tools may help by giving quick comments on errors, code structure, and possible corrections.

Several studies show that large language models can generate feedback for programming tasks. Azaiz et al. (2024) examined the use of GPT-4 for feedback generation in programming exercises. Their findings suggest that AI-generated feedback can be useful, but it is not always fully reliable. Bernik et al. (2025) also showed that large language models can support grading and feedback for student assignments. These studies suggest that AI can assist feedback work, but teacher supervision remains necessary.

AI tools can provide different kinds of feedback. They can explain syntax errors, suggest debugging steps, comment on code readability, and offer alternative solutions. Phung et al. (2023) showed that ChatGPT and GPT-4 can support several programming education tasks, including program repair, hint generation, grading feedback, and explanation. These functions can help students receive support before teachers are available.

Generative AI may also reduce teacher workload. Teachers can use it to prepare examples, generate practice tasks, draft feedback, and check common student errors. Dolinsky (2025) discussed the use of generative AI in introductory programming, including content, assignments, and assessment. Manorat et al. (2025) also showed that AI and machine learning have been widely used in assessment and feedback in programming education. This suggests that AI support is not limited to students. It can also help teachers manage teaching and assessment tasks.

However, AI-generated feedback has clear limits. It may be too general, too corrective, or incorrect. Sometimes, it may give the answer instead of guiding students to think. This can reduce the value of feedback as a learning process. Bringula (2024) also noted that ChatGPT can support programming courses, but its limitations need to be considered carefully.

For feedback to support learning, it should not only fix code. It should help students understand why an error happens and how to solve similar problems later. Therefore, AI feedback should be used as an early layer of support, not as the final judgment. Teachers still need to check important feedback, design clear feedback rules, and help students evaluate AI suggestions.

Overall, generative AI can make feedback faster and more available. It can support both students and teachers. However, it should be combined with human checking and clear learning goals. In programming education, useful feedback should guide students toward understanding, not simply give them corrected code.

3.3. Assessment Challenges and Academic Integrity

Generative AI creates serious challenges for assessment in programming courses. Many traditional assignments ask students to write code outside class. These tasks are useful when the submitted code reflects the student's own thinking. However, AI tools can now generate complete programming solutions. This makes it harder to judge whether the submitted work shows the student's actual ability.

One major problem is homework validity. If students use AI to complete most of a task, the final program may not show their real programming knowledge. Tang et al. (2023) showed that AI-generated programming solutions can affect academic integrity and make detection difficult. Kiesler and Schiffner (2023) also showed that large language models can solve many introductory programming tasks, which creates new problems for assessment design.

Another issue is that correct code does not always show understanding. Students may submit code that runs well but cannot explain how it works. Kohen-Vacs et al. (2025) found that students may struggle to correct AI-generated code. This suggests that assessment should not only check whether a program works. It should also check whether students can evaluate, explain, and revise code.

Plagiarism detection is also more difficult. AI-generated code may not match existing online sources, so traditional plagiarism tools may fail to identify it. This problem makes academic integrity policies more important. Evangelista (2025) argued that assessment design and ethical AI policies need to be reconsidered in the age of ChatGPT. In programming education, this means that AI use must be made visible in the assessment process.

Assessment should therefore move beyond checking final code only. Students can be asked to explain their code, annotate AI-assisted parts, keep revision logs, or complete short oral checks. Baig et al. (2026) also suggested strategies such as reflective journals, oral exams, and clearer documentation of AI use. These methods can help teachers see both the final product and the learning process.

AI-generated code also raises quality concerns. Yetiştiren et al. (2023) showed that AI-assisted code generation tools may produce code with different levels of correctness and quality. This matters for assessment because students may accept code that looks correct but contains hidden errors, poor design, or weak maintainability. Silva et al. (2024) also noted concerns about overdependence and limited understanding when students use ChatGPT in programming.

Overall, generative AI does not make assessment impossible, but it makes traditional assessment less reliable. Programming courses need assessment tasks that check understanding, reasoning, debugging, and explanation. Fair assessment should not only ask whether code works. It should also examine whether students understand the reasoning behind the code.

4. IMPLICATIONS

The reviewed studies suggest that generative AI should be integrated carefully into programming courses. The main issue is not whether students use AI tools, because many students already use them. The more important issue is how teachers guide, monitor, and assess this use.

First, students need clear instructions. They should know when AI use is allowed, what kinds of support are acceptable, and what must be completed independently. For example, using AI to explain an error may support learning, but submitting AI-generated code without understanding it should not be accepted. Clear rules can reduce confusion and support fair learning.

Second, students need prompt training. They should learn how to ask useful questions, check AI responses, and revise their code based on feedback. Güner and Er (2025) showed that students interact with ChatGPT in different ways. Nathaniel et al. (2025a) also emphasized the role of guided and scaffolded AI use in supporting programming learning. Therefore, teachers should not assume that students can use AI tools effectively without support.

Third, programming tasks should require explanation and reflection. Students should not only submit final code. They should also explain their logic, describe errors they fixed, and justify important changes. Kohen-Vacs et al. (2025) showed that students may find it difficult to correct AI-

generated code. Explanation and correction tasks can help teachers check whether students understand the program, not only whether the program runs.

Fourth, AI feedback should be combined with teacher feedback. AI tools can provide quick support, but they should not become the final source of judgment. Teachers still need to check important feedback, especially when students are learning basic concepts. This is necessary because AI feedback may be incomplete, inaccurate, or too direct.

Finally, assessment design should be updated. Programming assessment should include tasks that make student thinking visible, such as in-class coding, oral explanation, code annotation, reflection logs, and AI-use disclosure. Baig et al. (2026) and Evangelista (2025) suggest that assessment policies need to respond to generative AI. These strategies can help teachers assess both the final product and the learning process.

Overall, responsible use of generative AI requires guidance, transparency, and human oversight. AI tools are useful when they help students ask questions, test ideas, debug code, and revise their work. They become risky when students use them without reflection, verification, or teacher guidance.

5. LIMITATIONS AND FUTURE RESEARCH:

This review has some limitations. First, it is a focused literature review, not a full systematic review. It does not cover all studies on artificial intelligence or machine learning in programming education. Instead, it focuses on recent studies related to generative AI, learning support, feedback, and assessment.

Second, the field is still new. Many studies on ChatGPT and generative AI in programming education were published only recently (Baig et al., 2026; Garcia, 2025). Some studies also use small samples, short interventions, or specific course contexts. Therefore, their findings may not apply to all programming courses.

Third, the evidence is still mixed. Some studies report positive effects on engagement, feedback, and learning support. Other studies show limited learning gains, overreliance, or weak understanding among novice programmers (Xue et al., 2024; Yang et al., 2025). This means that more empirical research is needed.

Future research should examine the long-term effects of generative AI on programming learning. More studies are needed to understand whether students actually improve their programming logic, debugging skills, and problem-solving ability over time. Research should also examine how different types of guidance, prompt training, and assessment design affect student learning.

Future studies should also focus more on teachers. Generative AI may reduce some workload, but it may also create new tasks, such as checking AI feedback, redesigning assignments, and monitoring AI use. More research is needed on how teachers manage these tasks in real programming classes.

Finally, more work is needed on fair and valid assessment. Programming courses need assessment models that can measure both final code and student understanding. Future research should test whether methods such as oral explanation, code annotation, reflection logs, in-class coding, and AI-use disclosure can improve assessment validity in AI-supported programming courses.

6. CONCLUSION

Generative AI has become an important issue in programming education. Tools such as ChatGPT, GPT-4, and GitHub Copilot can support students by explaining concepts, helping with debugging, and providing feedback. They can also support teachers by helping with materials, feedback, and some grading tasks.

However, the benefits are not automatic. Students may rely on AI-generated code without developing enough understanding. AI tools also make homework, grading, plagiarism detection, and assessment validity more difficult. These risks are especially important in introductory programming, where students need to build basic logic, debugging skills, and problem-solving ability.

This review shows that generative AI should be used as a guided support tool. It should not replace programming practice, teacher guidance, or students' own reasoning. Students need clear instructions, prompt training, and opportunities to explain and revise their code. Teachers need assessment methods that check both the final product and the learning process.

Overall, the value of generative AI depends on how it is guided, checked, and assessed. Its main role should be to help students think, debug, and solve problems, not simply produce correct code.

ACKNOWLEDGMENT

ChatGPT was used only for language editing. The author developed the review scope, analysis, interpretation, and manuscript content, and takes full responsibility for the final work.

REFERENCES

1) Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-generation for programming exercises with GPT-4. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1 (pp. 31-37).
2) Baig, M. I., Yadegaridehkordi, E., & Bokani, A. (2026). ChatGPT in computer programming education: A review of current literature and applications. Australasian Journal of Educational Technology.
3) Bernik, A., Radošević, D., & Čep, A. (2025). A Comparative Study of Large Language Models in Programming Education: Accuracy, Efficiency, and Feedback in Student Assignment Grading. Applied Sciences, 15(18), 10055.
4) Bringula, R. (2024, February). ChatGPT in a programming course: benefits and limitations. In Frontiers in Education (Vol. 9, p. 1248705). Frontiers Media SA.

5) Dolinsky, M. (2025). Generative artificial intelligence and introductory programming education at universities: Content, assignments, assessment. WSEAS Transactions on Advances in Engineering Education, 22, 41-50.
6) Evangelista, E. D. L. (2025). Ensuring academic integrity in the age of ChatGPT: Rethinking exam design, assessment strategies, and ethical AI policies in higher education. Contemporary Educational Technology, 17(1), ep559.
7) Garcia, M. B. (2025). Teaching and learning computer programming using ChatGPT: A rapid review of literature amid the rise of generative AI technologies. Education and information technologies, 30(12), 16721-16745.
8) Groothuijsen, S., Van den Beemt, A., Remmers, J. C., & van Meeuwen, L. W. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. Computers and Education: Artificial Intelligence, 7, 100290.
9) 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.
10) Kiesler, N., & Schiffner, D. (2023). Large language models in introductory programming education: ChatGPT's performance and implications for assessments. arXiv preprint arXiv:2308.08572.
11) Kohen-Vacs, D., Usher, M., & Jansen, M. (2025). Integrating generative AI into programming education: Student perceptions and the challenge of correcting AI errors. International Journal of Artificial Intelligence in Education, 1-19.
12) Manorat, P., Tuarob, S., & Pongpaichet, S. (2025). Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence, 8, 100403.
13) Nathaniel, J., Oyelere, S. S., Suhonen, J., & Tedre, M. (2025a). Investigating the impact of Generative AI integration on the Sustenance of Higher-Order Thinking Skills and understanding of programming logic in Programming Education. Computers and Education: Artificial Intelligence, 100460.
14) Nathaniel, J., Oyelere, S. S., Suhonen, J., & Tedre, M. (2025b). Literature Review on the Integration of Generative AI in Programming Education. International Journal of Artificial Intelligence in Education, 1-32.
15) Park, A., & Kim, T. (2025). Code suggestions and explanations in programming learning: Use of ChatGPT and performance. The International Journal of Management Education, 23(2), 101119.
16) Phung, T., Pădurean, V. A., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., ... & Soares, G. (2023, August). Generative AI for programming education: Benchmarking ChatGPT, GPT-4, and human tutors. In Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2 (pp. 41-42).
17) Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., ... & Briggs, B. (2024, August). The widening gap: The benefits and harms of generative ai for novice programmers. In Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1 (pp. 469-486).
18) Scholl, A., & Kiesler, N. (2024, October). How novice programmers use and experience ChatGPT when solving programming exercises in an introductory course. In 2024 ieee frontiers in education conference (fie) (pp. 1-9). IEEE.
19) Silva, C. A. G. D., Ramos, F. N., De Moraes, R. V., & Santos, E. L. D. (2024). ChatGPT: Challenges and benefits in software programming for higher education. Sustainability, 16(3), 1245.
20) Tang, C. M., Ng, V. S., Leung, H. M., & Yuen, J. C. (2023). AI-generated programming solutions: impacts on academic integrity and good practices.
21) Xue, Y., Chen, H., Bai, G. R., Tairas, R., & Huang, Y. (2024, April). Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in CS1. In Proceedings of the 46th International conference on software engineering: software engineering education and training (pp. 331-341).
22) Yang, T. C., Hsu, Y. C., & Wu, J. Y. (2025). The effectiveness of ChatGPT in assisting high school students in programming learning: Evidence from a quasi-experimental research. Interactive Learning Environments, 33(6), 3726-3743.
23) Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. arXiv preprint arXiv:2304.10778.
24) Zviel-Girshin, R. (2024). The good and bad of AI tools in novice programming education. Education Sciences, 14(10), 1089.

Corresponding Author: Dang Thi Kim Giao