World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

INTERACTIVE GENERATIVE AI CODING ASSISTANT FOR STUDENTS

Kavitha Soppari, Bhavitha Vupperpally *, Rupa Sri Desai Bobbilipati and Rohan Devu

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning) of ACE Engineering College, Hyderabad, Telangana India.
* Corresponding Author

ORCID Details
Kavitha Soppari: https://orcid.org/0000-0002-5343-2320
Bhavitha Vupperpally : https://orcid.org/0009-0006-5835-8525
Rupa Sri Desai Bobbilipati: https://orcid.org/0009-0009-3608-9143
Rohan Devu : https://orcid.org/0009-0002-9865-5701

World Journal of Advanced Research and Reviews, 2026, 32(02), 1487–1492

Publication history: Received on 09 October 2026; revised on 18 November 2026; accepted on 20 November 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.32.2.2956

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

The Interactive Generative AI Coding Assistant for Students aims to make programming education more interactive, personalized, and effective. It addresses common challenges faced by beginners, such as understanding programming concepts, identifying coding errors, and developing problem-solving skills. The system provides coding exercises, automated error analysis, and progressive AI-generated hints to help students understand and correct mistakes without directly revealing complete solutions. It also tracks learning progress, identifies individual strengths and weaknesses, and recommends suitable exercises based on student performance. Additionally, teacher dashboards provide insights into student progress and learning difficulties. Overall, the system encourages independent thinking, improves programming skills, and supports personalized learning.

Keywords: Generative AI, Artificial Intelligence, Programming Assistant, Adaptive Learning, Error Analysis, Personalized Learning, Coding Practice, Problem-Solving Skills.

1 INTRODUCTION

Programming education plays an important role in developing logical thinking, computational skills, and problem-solving abilities among students. However, beginners often face difficulties in understanding programming concepts, identifying coding errors, and debugging their programs. Traditional programming platforms mainly provide test results and error messages, which may not offer sufficient guidance to overcome these challenges.

Recent advancements in Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs) have created new opportunities for interactive and personalized programming assistance. These technologies can provide code explanations, identify errors, generate hints, and support students throughout the coding process.

This review examines existing research on AI-based programming assistants and intelligent tutoring systems, focusing on automated error analysis, progressive hint generation, and adaptive learning approaches. It highlights how these techniques can improve programming education, strengthen students' problem-solving abilities, and encourage independent learning.

2 LITERATURE SURVEY

2.1 Yang et al. (2024) – "Enhancing Python Learning with PyTutor: Efficacy of a ChatGPT-Based Intelligent Tutoring System in Programming Education."

2.1.1 Methodologies and Algorithms

The system integrates the ChatGPT model with an online programming learning environment to generate context-aware assistance. It provides four levels of support: pseudocode hints, cloze code hints, basic code solutions, and advanced code solutions. Its effectiveness was evaluated through a comparative study involving 35 students using PyTutor and 36 students in a comparison group.

2.2 Lin et al. (2026) – "Design of a Scalable AI-Powered Tutoring System with Adaptive, Error-Aware Prompting Built on Automated Programming Assessment."

Lin, and Yang (2026) proposed an AI-powered tutoring system that combines automated programming assessment with Large Language Models (LLMs). The system generates personalized hints based on students' coding errors and problem-solving behaviour rather than directly providing complete solutions. This approach aims to improve programming assistance while maintaining student independence and learning autonomy.

2.2.1 Methodologies and Algorithms

The proposed methodology uses a microservices architecture integrating a DOMjudge-based automated assessment service with an LLM-driven tutoring module. It analyses programming submissions, assessment results, and student interaction histories to generate adaptive, error-aware prompts. The system also maintains learning-state representations from submission histories and AI interactions to provide contextual guidance while preserving standardized code evaluation.

2.3 Daungcharone et al. (2026) – "Coding with AI: Exploring Technology Acceptance and Learning Motivation in Computer Programming with AI Assistance."

Daungcharone et al. (2026) investigated the acceptance of AI-assisted learning tools and their influence on students' motivation in programming education. The study explored students' perceptions of usefulness, ease of use, confidence, and willingness to use AI tools. The findings indicated positive perceptions and high motivation scores, suggesting that AI-assisted learning can encourage student engagement, including among learners with less interest in programming.

2.3.1 Methodologies and Algorithms

The study implemented an AI-assisted learning environment for web application development and evaluated students' technology acceptance and learning motivation. Students were categorized into three groups based on their programming enthusiasm: dislike, neutral, and like. A **one-way Multivariate Analysis of Variance (MANOVA)** was used to examine differences in perceived ease of use, perceived usefulness, attitude, intention to use, and learning motivation.

2.4 Kimmel et al. (2024) – "Enhancing Programming Error Messages in Real Time with Generative AI."

Kimmel et al. (2024) investigated the use of Generative AI to improve programming error messages and make them easier for students to understand. The study integrated ChatGPT-generated feedback into an automated programming assessment tool to explain compiler, runtime, and logical errors. The findings indicated that AI-generated feedback does not automatically improve learning, highlighting the importance of feedback quality and interface design.

2.4.1 Methodologies and Algorithms

The researchers integrated GPT-4-generated feedback into the Athene automated programming assessment tool. Whenever students submitted programs, the system analyzed compiler errors, runtime errors, and logical errors and generated corresponding explanations using ChatGPT. The researchers evaluated the usefulness and usability of the AI-generated feedback, examining how effectively it supported students in understanding programming errors.

2.5 Liffiton et al. (2023) – "CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes."

Liffiton et al. (2023) introduced CodeHelp, an LLM-powered programming assistant designed to provide on-demand support to students without directly revealing complete solutions. The system uses guardrails to generate appropriate guidance and help learners understand and resolve coding errors. Its evaluation with 52 students over 12 weeks showed that students valued its availability and programming assistance, while instructors found it useful as a complement to classroom support.

2.5.1 Methodologies and Algorithms

The methodology uses Large Language Models with a structured prompting pipeline and guardrails to control the responses generated for student queries. Students can submit programming questions along with their code and error messages to receive contextual assistance without being given direct solutions. The system was evaluated through a 12-week classroom deployment, analyzing student interactions, usage patterns, and feedback from 52 students

2.6 He et al. (2025) – "A Comparative Study on the Effectiveness of Generative AI Tools and Pair Programming in College Students' Programming Learning."

He and Li (2025) examined the effectiveness of AI-assisted pair programming compared with traditional human–human pair programming among college students learning C programming. The study investigated students' programming performance, self-efficacy, and learning interest. The results showed that AI-assisted programming positively improved students' programming performance, confidence, and interest in learning.

2.6.1 Methodologies and Algorithms

The researchers employed a quasi-experimental approach involving 80 undergraduate students divided into an AI-assisted group of 38 students and a human–human pair programming group of 42 students. Over eight weeks, the AI-assisted group used generative AI tools for understanding concepts, writing code, and debugging, while the control group worked with human partners. Pre-tests and post-tests, along with paired-sample and independent-sample t-tests, were used to evaluate changes in programming performance, self-efficacy, and learning interest.

2.7 Syed et al. (2026) – "AI-Powered Smart Study Assistant Using Generative AI."

Syed and Daimi (2026) proposed an AI-powered study assistant that uses Generative AI to provide personalized academic support to students. The system offers features such as answering questions, summarizing study materials, generating notes, and resolving doubts in real time. The study highlights how AI-based assistance can make learning more interactive, accessible, and effective.

2.7.1 Methodologies and Algorithms

The system uses Natural Language Processing (NLP) techniques and transformer-based Large Language Models (LLMs) to understand student queries and generate relevant responses. It is developed using Python, the Flask framework, NLP libraries, and Generative AI APIs, supported by an educational knowledge base collected from textbooks, lecture notes, and online learning resources. Its performance is evaluated based on response accuracy, relevance, response time, and user satisfaction.

2.8 Mahdaoui et al. (2022) – "Comparative Study Between Automatic Hint Generation Approaches in Intelligent Programming Tutors.

Mahdaoui et al. (2022) presented a comparative study of automatic hint-generation approaches used in Intelligent Programming Tutors. The study examined how data-driven techniques and program repair methods can generate hints to help students correct errors and progress through programming exercises. It highlighted the strengths and limitations of different approaches in providing effective, step-by-step programming assistance.

2.8.1 Methodologies and Algorithms

The study compared automatic hint-generation techniques based on data-driven approaches and program repair methods. Data-driven techniques use previously collected student solutions and coding patterns to generate suggestions, while program repair techniques identify and recommend code modifications such as inserting or deleting code segments. The comparison focused on the capabilities, advantages, and limitations of these approaches in generating useful hints without requiring extensive expert-defined knowledge.

2.9 Kouam (2024) – "The Effectiveness of Intelligent Tutoring Systems in Supporting Students with Varying Levels of Programming Experience."

Kouam (2024) investigated the effectiveness of Intelligent Tutoring Systems (ITS) in supporting students with different levels of programming experience. The study examined their impact on programming skills, feedback usefulness, and student satisfaction. The findings highlighted that ITS effectiveness varies according to learners' experience, emphasizing the importance of personalized programming support.

2.9.1 Methodologies and Algorithms

The study adopted a mixed-methods research approach involving 160 computer science students classified into beginner, intermediate, and advanced programming levels. All participants used the Blackbox Intelligent Tutoring System and provided feedback through questionnaires containing closed-ended and open-ended questions. Quantitative data were analyzed using descriptive statistics, Chi-square tests, and ANOVA, while qualitative responses were examined through thematic analysis using NVivo software.

2.10 Hafeez et al. (2023) – "Evaluating Solo vs Pair Programming in an Online Setting for Introductory Programming Students."

Hafeez et al. (2023) investigated the effectiveness of solo and pair programming among students learning programming in an online environment. The study examined how collaborative coding influences programming exercise completion and solution correctness. The results showed that pair programming achieved higher success rates than solo programming, highlighting the benefits of collaborative learning for beginners.

2.10.1 Methodologies and Algorithms

The researchers developed an experimental methodology in which students performed programming exercises individually or in pairs using a shared online Integrated Development Environment (IDE). The study involved 116 students enrolled in an introductory programming course, with pair programming involving driver and navigator roles through video communication. The researchers compared test-case completion and solution correctness between the two groups using submission results and statistical analysis.

2.11 Comparative Analysis

Table 1: Comparative Analysis of Existing Research Papers on Programming Tutors.

S.No.	Name of the paper	Algorithms Used	Results	Limitations
1.	Enhancing Python Learning with PyTutor: Efficacy of a ChatGPT-Based Intelligent Tutoring System in Programming Education	ChatGPT, structured hint generation (pseudocode, cloze, basic and advanced hints)	Improved engagement, completion and success rates, particularly among beginners. No numerical accuracy reported.	Students may become dependent on AI hints, affecting independent problem-solving.
2.	Design of a Scalable AI-Powered Tutoring System with Adaptive, Error-Aware Prompting Built on Automated Programming Assessment	DOMjudge, Large Language Models (LLMs), adaptive error-aware prompting	Demonstrates integration of automated assessment and AI guidance. No numerical accuracy reported.	Requires real-world evaluation and further development of learner modelling.
3.	Coding with AI: Exploring Technology Acceptance and Learning Motivation in Computer Programming with AI Assistance	AI-assisted programming, questionnaires, one-way MANOVA	Evaluates student acceptance and learning motivation. No programming accuracy reported.	Positive perceptions do not necessarily prove improved programming skills or long-term learning.
4.	Enhancing Programming Error Messages in Real Time with Generative AI	GPT-4, prompt engineering, automated programming assessment	AI-generated feedback provided additional programming support, but usefulness varied. No numerical accuracy reported.	AI feedback may be vague or incorrect and may not always improve learning.

5.	CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes	LLMs, prompt engineering, guardrails	Evaluated with 52 students over 12 weeks; students appreciated immediate programming support.	AI responses may be inaccurate, and students may over-rely on assistance.
6.	A Comparative Study on the Effectiveness of Generative AI Tools and Pair Programming in College Students' Programming Learning	Generative AI-assisted programming and human pair programming	Reports improvements in programming performance, self-efficacy and learning interest.	Limited sample size and focus on C programming may affect generalizability.
7.	AI-Powered Smart Study Assistant Using Generative AI	Natural Language Processing (NLP), Large Language Models, Generative AI	Reports improved learning efficiency and student engagement.	May generate incomplete or inaccurate responses to complex queries.
8.	Comparative Study Between Automatic Hint Generation Approaches in Intelligent Programming Tutors	Data-driven approaches and program-repair-based approaches	Compares different automatic hint-generation approaches. No numerical accuracy reported.	Hints may lack detailed explanations and depend on reference solutions.
9.	The Effectiveness of Intelligent Tutoring Systems in Supporting Students with Varying Levels of Programming Experience	Intelligent Tutoring Systems and learner-support evaluation	Examines effectiveness acrossdifferent programming experience levels.	Effectiveness may vary according to students' programming experience and learning needs.
10.	Evaluating Solo vs Pair Programming in an Online Setting for Introductory Programming Students	Online pair programming, driver–navigator model, automated test-case evaluation	Pair programming achieved 63.5% success across all test cases, compared with 45% for solo programming.	Requires student coordination and does not provide AI-based personalized guidance

2.11.1 Research Gaps

The review of the existing research papers reveals several gaps in the development of AI-based programming learning systems. Although current studies explore Generative AI, Large Language Models (LLMs), and Intelligent Tutoring Systems to support programming education, most of them concentrate on individual functionalities such as code explanation, automated error detection, debugging assistance, or hint generation rather than integrating these features into a single learning platform. Existing approaches also have limitations in providing comprehensive error classification and identifying the specific programming concepts in which students face difficulties. While some systems offer automated hints, there is still a need for progressive, step-by-step guidance that helps students understand their mistakes and correct their code without directly providing complete solutions. Furthermore, personalized learning support requires improvement, as students have different programming abilities, learning speeds, and areas of weakness. Another important concern is the possibility of students becoming overly dependent on AI-generated answers, which may affect their independent thinking and problem-solving skills. Existing approaches also provide limited integration of continuous performance tracking, including coding attempts, recurring errors, hint usage, and concept-level improvement. In addition, teacher monitoring features need further attention to help instructors understand individual student progress and identify learning difficulties. Adaptive exercise recommendations based on students' previous performance and skill levels are also not sufficiently integrated into many existing systems. Therefore, there is a need for an integrated and interactive Generative AI-based programming assistant that combines automated code execution, error classification, progressive AI-generated hints, personalized exercise recommendations, student performance tracking, and teacher monitoring while encouraging independent learning and improving students' programming and problem-solving abilities.

Abbreviations

- AI – Artificial Intelligence
- LLM – Large Language Model

3 CONCLUSION

The review of existing research highlights the potential of Generative Artificial Intelligence and Intelligent Tutoring Systems in making programming education more effective and interactive. Existing studies demonstrate that AI-based tools can support students through code explanations, automated error detection, hint generation, and personalized learning assistance. However, challenges such as inaccurate feedback, limited personalization, overdependence on AI, and insufficient student progress monitoring remain important concerns. The proposed Interactive Generative AI Coding Assistant for Students aims to address these limitations by integrating code execution, automated error classification, progressive AI-generated hints, adaptive exercise recommendations, and student performance tracking into a single platform. It also provides teachers with insights into students' progress and learning difficulties. By guiding students instead of directly providing complete solutions, the system aims to encourage independent thinking, improve programming skills, and strengthen problem-solving abilities. Overall, this approach can contribute to a more personalized, supportive, and effective programming learning environment.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

The authors sincerely express their gratitude to Dr. Kavitha Soppari, professor and Head of the Department, CSE (Artificial Intelligence and Machine Learning), ACE Engineering College, for her valuable guidance, encouragement, and continuous support throughout this research work.

Disclosure of conflict of interest

The authors declare no conflict of interest regarding this manuscript.

REFERENCES

[1] L.-W. Lin, M.-R. Lin, and H.-C. Yang, “Design of a Scalable AI-Powered Tutoring System with Adaptive, Error-Aware Prompting Built on Automated Programming Assessment,” 2026.

[2] K. Daungcharone, K. Thongkoo, K. Tumphasuwan, and A. Saosaovaphak, “Coding with AI: Exploring Technology Acceptance and Learning Motivation in Computer Programming with AI Assistance,” 2026.

[3] A. Syed and I. R. Daimi, “AI-Powered Smart Study Assistant Using Generative AI,” *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. XV, issue XIII, May 2026, doi: 10.51583/IJLTEMAS.2026.1501300003.

[4] H. He and X. Li, “A Comparative Study on the Effectiveness of Generative AI Tools and Pair Programming in College Students’ Programming Learning,” in *Proceedings of IEEE ICDEL*, 2025.

[5] A. W. F. Kouam, “The Effectiveness of Intelligent Tutoring Systems in Supporting Students with Varying Levels of Programming Experience,” *Discover Education*, vol. 3, article 278, 2024, doi: 10.1007/s44217-024-00385-3.

[6] A. C. M. Yang, J.-Y. Lin, C.-Y. Lin, and H. Ogata, “Enhancing Python Learning with PyTutor: Efficacy of a ChatGPT-Based Intelligent Tutoring System in Programming Education,” *Computers and Education: Artificial Intelligence*, vol. 7, article 100309, 2024, doi: 10.1016/j.caeai.2024.100309.

[7] B. Kimmel, A. L. Geisert, L. Yaro, B. Gipson, R. T. Hotchkiss, S. K. Osae-Asante, H. Vaught, G. Wininger, and C. Yamaguchi, “Enhancing Programming Error Messages in Real Time with Generative AI,” 2024.

[8] M. Liffiton, B. Sheese, J. Savelka, and P. Denny, “CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes,” in *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, 2023.

[9] M. Hafeez, A. Karki, Y. Radwan, A. Saha, A. Zavaleta Bernuy, and B. Harrington, “Evaluating Solo vs Pair Programming in an Online Setting for Introductory Programming Students,” 2023.

[10] M. Mahdaoui, S. Nouh, M. S. Elkasmi Alaoui, and M. Sadiq, “Comparative Study Between Automatic Hint Generation Approaches in Intelligent Programming Tutors,” *Procedia Computer Science*, vol. 198, pp. 391–396, 2022.