Global Academic Frontiers, Vol. 3, Issue 2, 2025, pp. 99-105 Print ISSN 2995-5688 Online ISSN 2995-570X
DOI: https://doi.org/10.5281/zenodo.15582458

Construction of a Large Language Model-Driven Online Programming Experiment System and Research on Active Learning Paradigm Transformation

Shengying Yang [1*] Chen Lu [1] Fangfang Qiang [1]

[1] Zhejiang University of Science and Technology, Hangzhou 310023, China
*Corresponding author Email: syyang@zust.edu.cn
Received 14 April 2025; Accepted 29 May 2025; Published 3 June 2025

Abstract: Traditional online programming teaching platforms have significant deficiencies in supporting the development of students' abilities. The core issue is concentrated on the unidirectional nature of the evaluation system: it cannot effectively assess engineering elements such as code standardization, style uniformity, and runtime efficiency, nor can it provide students with in-depth suggestions for improvement. To break through this limitation, this paper constructs a new generation of online programming experiment platforms based on large language model technology. The platform can analyze students' code logic in real-time, generate targeted error correction suggestions, explanations of knowledge points, and optimization plans, and supports language interaction to help students quickly understand programming concepts. Experiments show that the platform significantly improves students' programming practice abilities, confirming its value in programming education. It provides an expandable technical solution for the innovation of programming education models and is of great significance in promoting the transformation of programming education from passive infusion to active exploration.

Keywords: Large Language Models, Programming Experiment Platform, Program Design, Artificial Intelligence

1 Introduction

1.1 Policy Background and Development of AI Technology

With the rapid development of artificial intelligence technology, large language models (LLMs) have become the core engine driving the digital transformation of education [1]. In recent years, national policies have clearly supported the application of AI technology in the field of education. For example, the "Education Power Construction Plan (2024–2035)" clearly proposes the development of "education-specific large models." By building a national education big data center and an intelligent evaluation system, it aims to transform teaching from "standardization" to "personalization." The Ministry of Education's "Action Plan for Empowering Education with Artificial Intelligence" further refines the requirements, emphasizing the systematic integration of AI education in primary and secondary schools by stages and the opening of resources from universities and enterprises to build smart education platforms. Local governments have also actively responded. Shanghai's "Several Measures for Promoting the Innovation and Development of Large Language Models (2023–2025)" focuses on "intelligent education and teaching" scenarios, encouraging the development of AI-assisted teaching tools and prioritizing the cultivation of AI talents in the education field. These policies collectively point to a common goal: to reshape the education ecosystem with AI technology, achieving large-scale personalized teaching and the cultivation of innovative abilities.

In the field of computer science education, large language models (LLMs) have demonstrated the ability to

clearly and systematically explain complex computer science knowledge and concepts [2]. These models can transform abstract algorithm logic and data structure principles into multi-level example explanations through natural language interaction, and dynamically demonstrate them with actual code snippets. This capability not only lowers the cognitive threshold for beginners but also provides differentiated knowledge decomposition and extension for students at different learning stages. The rise of AI language models has provided a new perspective and possibility for programming education [3].

At present, there are still few experiment platforms that can deeply integrate AI language models and systematically apply them to programming teaching. Although AI language models have made breakthrough progress in fields such as text generation and data analysis, their specialized applications in programming education remain relatively weak. Traditional programming teaching generally relies on the one-way infusion of knowledge by teachers, and students face problems such as low code debugging efficiency, delayed error feedback, and insufficient personalized guidance. Existing programming experiment platforms are mostly limited to basic functions (such as online compilation) and lack deep integration with large language models, making it difficult to achieve real-time logic analysis, natural language interaction, and adaptive learning path recommendations. This disconnection between technological application and policy orientation restricts the efficiency of students' computational thinking and practical ability development, and there is an urgent need to build an integrated AI large model programming experiment system to bridge the technological gap between policy goals and educational practice.

1.2 Deficiencies of Traditional Online Programming Platforms

Traditional online programming experiment platforms (such as ACM online judging systems) have played an important role in cultivating students' basic programming abilities as important tools in college programming education by providing standardized practice links and timely feedback mechanisms [4]. However, with the deepening of the digital transformation of education, their inherent limitations have gradually been exposed, restricting students' learning outcomes and potential development [5].

Firstly, the core function of traditional platforms is limited to the binary judgment of code execution results. The system can only provide conclusions of "correct" or "wrong", but cannot analyze the root causes of errors [6]. The platform cannot locate the erroneous code segments or generate correction suggestions, which is not conducive to students' understanding of the fundamental reasons for errors and forces them to consume energy in repeated trial and error.

Secondly, the singularity of the evaluation dimension restricts the cultivation of students' engineering abilities. Existing platforms overly focus on the functional realization of the code while neglecting key indicators such as code style standardization and maintainability. Students' code may pass all test cases but may have problems such as chaotic variable naming and high module coupling. Long-term training can easily form a "function-first" mindset. At the same time, the hidden danger of incomplete test case coverage further exacerbates the evaluation bias—if boundary conditions (such as empty input and extreme values) are not designed, students may misjudge the robustness of the code, leading to crash risks in actual applications.

A deeper contradiction is reflected in the lack of real-time interactive guidance. Many platforms cannot provide real-time guidance and code correction suggestions. When students encounter difficulties, they may be left in a helpless situation, which may increase their sense of frustration and hinder the in-depth development of students' computational thinking.

2 System Design

This platform focuses on the core chain of students' programming learning and adopts a lightweight, high-response architecture design by integrating large language models into the programming experiment system, significantly enhancing the interactivity and adaptability of the learning platform and helping students learn programming in a more effective and targeted way. As shown in Figure 1, the system architecture is divided into the

User Layer and the Core Processing Layer. The platform has built the following functional modules around the core educational goals:

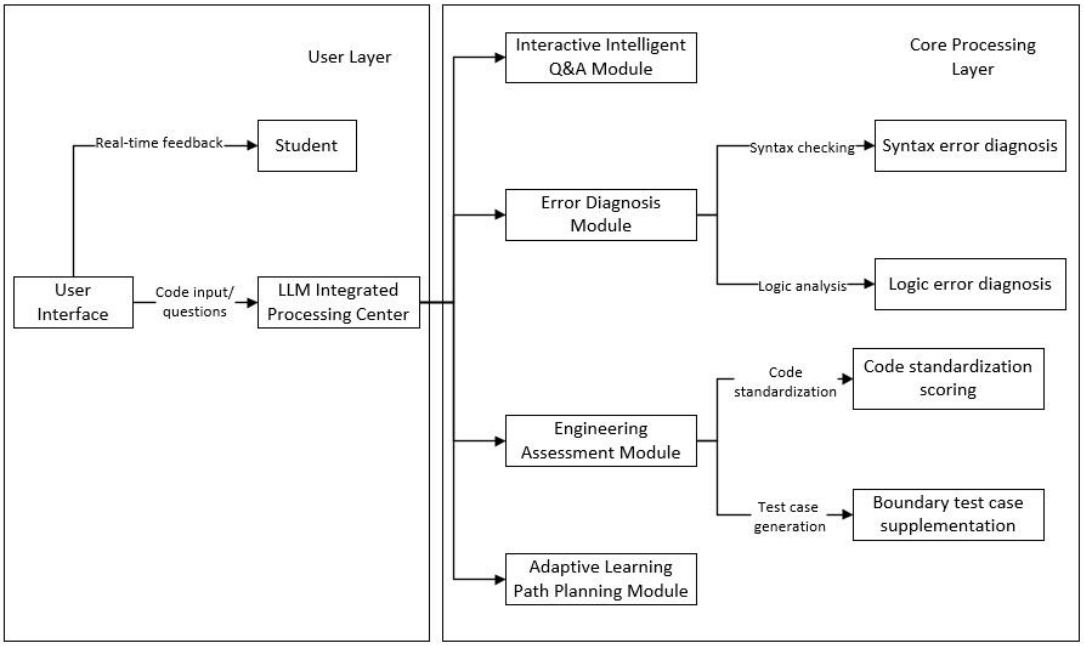

Figure 1. Large Language Model-Driven Online Programming Experiment System Architecture

2.1 Interactive Intelligent Q&A Module

This module integrates a code editor with a natural language interaction interface, supporting students to initiate questions in a composite manner of "code snippets + problem descriptions." The platform dynamically analyzes the core requirements of programming problems by combining code context semantic analysis and large model reasoning capabilities, generating multi-dimensional answers and knowledge associations. For ambiguous questions (e.g., "How to optimize this sorting code?"), the platform actively guides the refinement of the question scope to promote students' autonomous thinking and problem localization abilities.

2.2 Error Diagnosis Module

This module uses a hierarchical processing mechanism to deal with programming problems at different levels. For example, as shown in figure 2, the platform will conduct real-time checks on the code submitted by students. If basic syntax errors occur (e.g., indentation errors, undefined variables), it directly returns modification suggestions. For complex logic problems (e.g., multi-thread synchronization failure, algorithm boundary condition omission), it provides detailed error reports to help students understand the root causes of errors rather than mechanically copying answers. The system prioritizes returning code correction suggestions, followed by extended knowledge links and related practice questions to help students better grasp the knowledge points.

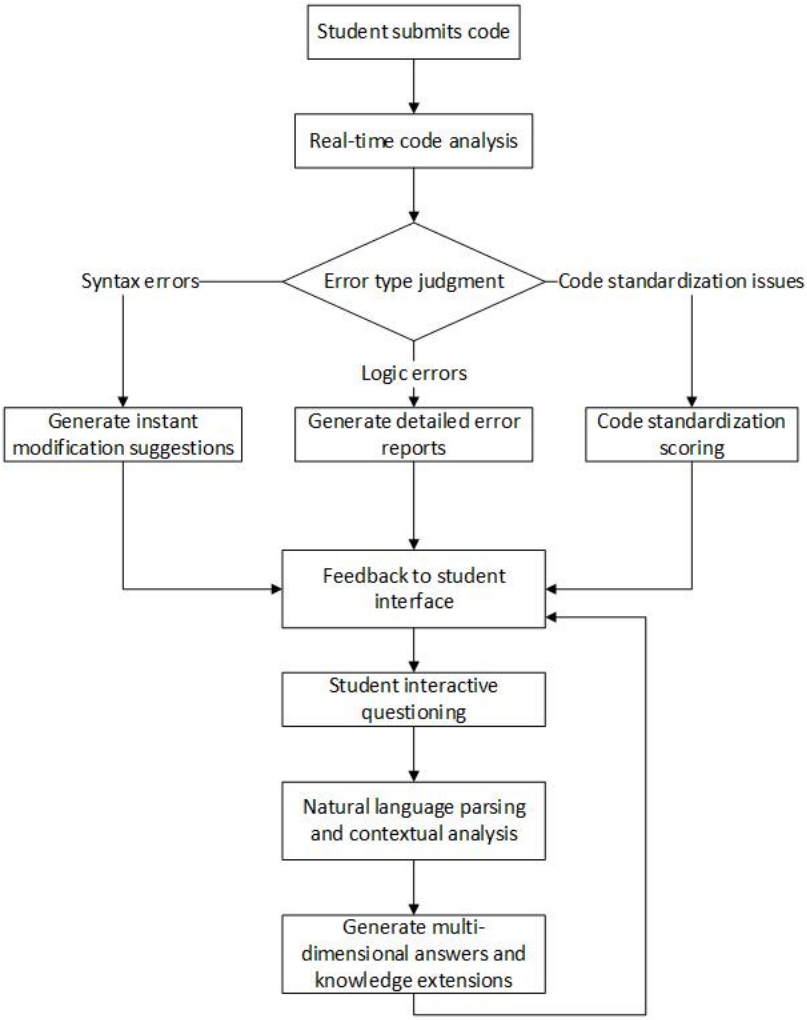

Figure 2. Real-Time Code Analysis Flow

2.3 Engineering Assessment Module

Considering the defects of traditional programming platform test cases and the neglect of code standardization, the platform's assessment dimensions have added scores for code standardization and maintainability. It also uses large models to automatically generate boundary test cases (e.g., empty input, extreme value data) to supplement the coverage blind spots of the pre-set cases in traditional platforms and fundamentally improve code robustness.

2.4 Adaptive Learning Path Planning Module

Traditional platforms' static resource recommendations lead to rigid learning paths. This platform builds a dynamic knowledge graph based on students' programming behavior data to achieve personalized learning support. It breaks the traditional platform's "mechanical repetition" training model and realizes a progressive training path from single knowledge points to systematic engineering capabilities [7].

3 Performance Evaluation and Practical Application

3.1 Verification Design of Platform Efficiency in Teaching Scenarios

Traditional programming experiment teaching generally adopts a linear model of "task assignment - independent practice - centralized Q&A": after the teacher assigns the experiment task and provides a brief explanation, students independently complete the code writing in the computer lab, and the teacher provides auxiliary Q&A through rounds of guidance. The experiment ends with students submitting their program design code.

This teaching method leads to insufficient student computer time, low program debugging and learning efficiency, and in a limited class time, teachers are often occupied by individual students' complex problems for a long time, making it difficult to provide targeted guidance to most students. At the same time, due to the lack of

real-time feedback and progressive hints, students generally face problems such as low debugging efficiency and slow knowledge internalization. Therefore, this teaching model hardly achieves "universal personalization" for students.

To verify the actual efficiency of the platform in programming education, this study is based on the teaching scenario of the Big Data Comprehensive Practice Course in our university. The post-class experiment questions covering the entire process of data crawling and data cleaning in the course are selected as evaluation samples. The course experiment tasks are shown in Table 1. By using the program code generated by the large language model and submitting it to the judging system for verification, the students' performance in the last four weeks under the platform-assisted teaching mode is compared with that in the first four weeks under the traditional experiment teaching mode. Under the condition of similar experiment difficulty coefficients, the students' average experiment scores have significantly improved, as shown in Table 2, which confirms the platform's promotion of students' engineering abilities.

Table 1 Partial Experiment Task List of the Course

No.	Experiment Item	Experiment Type
1	Complete the deployment of the Java development environment under Linux and the creation process of a Spring Boot project	Verification
2	Integrate the deployment of Linux environment, the collaborative tool chain of Python/Kettle, and data cleaning technology to complete the entire process of traffic data collection and preprocessing	Basic Comprehensive
3	Deploy Hadoop clusters under Linux and practice HDFS file transfer to master the basic architecture of distributed storage systems and the core operations of data management	Basic Comprehensive
4	Develop distributed computing programs based on the MapReduce framework	Basic Comprehensive
5	Complete the deployment of Sqoop tools and the construction of MySQL database table structures	Verification
6	Implement front-end and back-end data interaction and visualization presentation based on the Java Web technology stack	Comprehensive Design

Table 2 Analysis of Experiment Test Scores with Use of System

Experiment Section	Number of students	Score	Difficulty Coefficient	Percentage of students who felt system was helpful
1	30	95	1.0	70%
2	26	92	1.5	50%
3	30	95	1.4	50%
4	28	93	1.8	65%

5	29	92	1.8	45%
6	30	90	1.6	80%

In addition, this study conducted a questionnaire survey among 30 students who used the system. The survey results show that 60% of students believe that with the assistance of large language models, they can better understand the questions and obtain targeted programming guidance. Students generally believe that the system can effectively diagnose their shortcomings, and the feedback and suggestions provided can greatly improve their programming skills and problem-solving efficiency. Although some of the code assisted by the system still needs adjustment, they believe that the reference suggestions obtained are very valuable. These feedback from students further verify that the system plays a positive role in cultivating students' logical thinking, reasoning ability, and innovative consciousness.

3.2 Risks of Misuse of Programming Platforms

Although large language models can bring significant benefits to higher education, they also come with potential misuse risks[8,9]. Large language models significantly improve learning efficiency through instant knowledge supply, enabling students to quickly master new programming languages and build technical frameworks. When dealing with complex programming problems, these large model tools can not only provide innovative solutions but also help cultivate students' logical thinking abilities. However, their convenience may also lead to over-reliance among students, thereby neglecting the importance of independent thinking.

To address these challenges, the online programming experiment system integrated with AI large language models developed in this project has imposed restrictions on interactions. In the error diagnosis module, the platform will not directly provide complete code answers but will give specific improvement suggestions based on the submitted code. Only after the user's code passes the verification will the system provide the complete optimized code, allowing students to clearly understand the parts that need improvement. Through this design, students can fully enjoy the benefits of large model tools while maintaining and cultivating their ability to solve problems independently, avoiding becoming mere code generation tools. This system design and usage strategy effectively reduces potential misuse risks while achieving educational goals.

4 Conclusion

This paper designs and implements an intelligent programming experiment system based on large language models, targeting the deficiencies of traditional online programming teaching platforms in evaluation dimension singularity, feedback delay, and insufficient interactivity. By integrating core functions such as code standardization analysis, dynamic error diagnosis, and hierarchical guidance feedback, the platform effectively compensates for the shortcomings of existing systems and provides a smarter and more personalized programming learning environment. Experimental results show that with the assistance of the platform, students' programming practice abilities have significantly improved. Through interaction with AI models, students can understand complex programming concepts while learning to deal with various challenges encountered in actual programming. Therefore, this AI-integrated online programming experiment system not only has important educational significance but also demonstrates broad application potential. It heralds the future direction of programming education and provides a reference path for programming experiment teaching.

Acknowledgements

This work was supported by Zhejiang Higher Education Society Project "Artificial Intelligence Empowers Education and Teaching Application Research" Special Project (No. KT2024464), Zhejiang University of Science and Technology Teaching Research and Reform Project (No. 2023-jg16), and Zhejiang University of Science and Technology Graduate Course Construction Project (No. 2021yjskj04).

References

[1] Alqahtani, T., Badreldin, H. A., Alrashed, M., Alshaya, A. I., Alghamdi, S. S., Bin Saleh, K., ... & Albekairy, A. M. (2023). The emergent role of artificial intelligence, natural learning processing, and large language models in higher education and research. *Research in social and administrative pharmacy*, *19*(8), 1236-1242.

[2] Savelka, J., Agarwal, A., Bogart, C., Song, Y., & Sakr, M. (2023, June). Can generative pre-trained transformers (gpt) pass assessments in higher education programming courses?. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 117-123).

[3] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*, *103*, 102274.

[4] Yu, J. H., Chang, X. Z., Liu, W., & Huan, Z. (2023). An online integrated programming platform to acquire students' behavior data for immediate feedback teaching. *Computer Applications in Engineering Education*, *31*(3), 520-536.

[5] Yan, Y. M., Chen, C. Q., Hu, Y. B., & Ye, X. D. (2025). LLM-based collaborative programming: impact on students' computational thinking and self-efficacy. *Humanities and Social Sciences Communications*, *12*(1), 1-12.

[6] Roy, D., Zhang, X., Bhave, R., Bansal, C., Las-Casas, P., Fonseca, R., & Rajmohan, S. (2024, July). Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (pp. 208-219)..

[7] Strmečki, D., Magdalenić, I., & Radosević, D. (2018). A systematic literature review on the application of ontologies in automatic programming. *International journal of software engineering and knowledge engineering*, *28*(05), 559-591.

[8] Rahman, M. M., & Watanobe, Y. (2023). ChatGPT for education and research: Opportunities, threats, and strategies. *Applied sciences*, *13*(9), 5783.

[9] Wen, W., & Wen, H. (2024). Bridging Generative AI Technology and Teacher Education: Understanding Preservice Teachers' Processes of Unit Design with ChatGPT. *Contemporary Issues in Technology and Teacher Education (CITE Journal)*, *24*(4), n4.