Understanding LLM Responses in Programming Tasks: A Study on Prompt Quality, Personalization, and RAG Limitation

Meilyna Hutajulu[1], Ioka Purba[2], Mulyadi Siahaan[3], Samuel Situmeang[4*], Mario Simaremare[5]

[1,2,3,4,5] *Department of Information Systems, Institut Teknologi Del, Jl. Sisingamangaraja, Sitoluama Laguboti, Kab. Tobasa, Sumatra Utara, 22381, Indonesia*

Keywords

Adaptive Prompting; Generative AI; Personalized Learning; Retrieval-Augmented Generation.

Correspondence Email:
samuel.situmeang@del.ac.id

Abstract

Programming is a fundamental skill in the field of technology and education in the digital era. However, students' understanding of programming varies significantly between senior and junior learners. Senior students tend to have stronger comprehension of code structure and programming logic, while junior students often struggle to identify well-structured code. Current learning processes remain general and are not sufficiently tailored to individual student needs. Existing personalization approaches are still shallow and have not been fully integrated with students' abilities and learning contexts. The emergence of Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs) such as GPT, Claude, and Gemini, offers new opportunities to support more adaptive and personalized programming education. LLMs can act as interactive learning assistants capable of explaining concepts, generating code examples, and providing direct feedback. However, their use also presents challenges, including hallucination risks that may lead to conceptual misunderstandings and a strong dependency on the quality of user-crafted prompts. In addition, prior studies show that Retrieval-Augmented Generation (RAG) has not performed optimally in debugging contexts, as external documents are not always effectively utilized by the model. These issues highlight the need to explore more effective approaches for leveraging GenAI to deliver programming learning that is adaptive, contextual, and safe from misinformation.

1. Introduction

Programming has become one of the core competencies in modern technology and education. In today's digital era, the ability to write and understand code is not only essential for students in computer science related fields but has also become part of broader technological literacy. However, the level of programming comprehension among students varies significantly. Senior or advanced-level students generally possess stronger abilities in understanding code structure, program logic, and syntactic patterns compared to junior or beginner-level students (Nurollahian et al., 2024).

These differences indicate a clear gap in programming education. Traditional teaching methods that rely on generalized and uniform instructional approaches often fail to accommodate the individual needs of students. As a result, some students struggle to follow the material, while others do not receive sufficient challenge to further develop their skills. This highlights the importance of adopting more personalized and adaptive learning approaches tailored to each student's capabilities. Several studies show that personalization in programming education remains limited to shallow personalization, where learning is adapted only superficially without considering individual interests, capabilities, or specific learning difficulties (Prather, 2024). Such approaches are not sufficient to help students deeply understand core concepts or navigate through code structures effectively.

Recent advancements in Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs) such as GPT, Gemini, and Claude, open promising opportunities for more adaptive and personalized learning support. LLMs can understand user queries, interpret context, and produce relevant responses, making them potential interactive assistants for learning concepts, writing code, and performing debugging tasks. However, these models still face significant challenges (Annuš, 2025). One major limitation is hallucination, where the model generates plausible-sounding but incorrect information (Berberette et al., 2023). This poses risks for novice learners, who may adopt incorrect concepts or misunderstand code semantics.

Another challenge lies in students' ability to formulate effective prompts. Many student especially beginners tend to produce unclear or incomplete prompts, resulting in irrelevant or unhelpful responses (Simaremare et al., 2024). Previous studies also found that the integration of Retrieval-Augmented Generation (RAG) within programming education has not been fully effective, especially for debugging tasks, where external documents provided by the system are often not utilized optimally by the model (Simaremare et al., 2025).

These challenges collectively demonstrate that leveraging GenAI for programming education is not straightforward. Despite its potential, significant gaps remain in how LLMs interpret student intent, adapt explanations to different skill levels, and deliver contextually appropriate guidance while minimizing hallucination.

Therefore, this study focuses on identifying and analyzing the underlying issues related to the use of GenAI particularly LLMs in supporting programming education. The goal is to develop a deeper understanding of how these systems respond to student questions, how contextual alignment can be improved, and how personalized learning experiences can be enhanced for students across varying levels of programming proficiency.

2. Literature Review

2.1 Programming Education and Its Challenges

Learning programming is a complex process that requires not only logical reasoning skills but also the ability to understand structure, syntax, and computational thinking patterns. Early-stage students often struggle to connect theoretical concepts with their practical implementation in the form of source code. According to (Nurollahian et al., 2024), senior students demonstrate a higher ability to recognize well-organized code structures compared to junior students, indicating that programming proficiency develops through consistent exposure and experience.

However, many instructional approaches in programming courses remain uniform. Students of varying skill levels receive the same materials and teaching methods without differentiation based on individual proficiency. As a result, students who struggle in the early stages tend to fall behind, while more advanced students do not receive sufficient challenges to progress further (Prather, 2024).

2.2 Personalization in Learning

Personalized learning aims to adapt materials, methods, and learning pace to each student's needs and abilities. In digital education, personalization has become a key approach to increasing student engagement and learning effectiveness (Helen & Nada, 2024).

Yet, many personalization approaches in programming education still rely on shallow personalization, where learning content is adjusted only superficially (e.g., renaming variables or changing narrative themes) without considering students' abilities or learning styles. Consequently, such personalization does not significantly improve the understanding of more complex programming concepts (Helen & Nada, 2024).

Previous studies emphasize that effective deep personalization requires analyzing students' learning behaviors, common mistakes, and problem-solving strategies. Implementing this type of personalization is challenging because it demands systems capable of automatically recognizing user context and proficiency (Helen & Nada, 2024).

2.3. GenAI in Education

Advances in GenAI have brought significant changes to education. One prominent example is the rise of LLMs such as GPT, Claude, and Gemini. LLMs are designed to understand natural language contexts and generate responses relevant to user queries. In programming education, LLMs can act as interactive learning assistants capable of explaining concepts, generating sample code, and helping identify errors in programs (Solanki & Khublani, 2024).

However, GenAI systems also present limitations. A key issue is hallucination, where the model produces responses that sound convincing but are incorrect or irrelevant. In programming tasks, hallucinated explanations or incorrect code can lead to conceptual misunderstandings that students may struggle to detect (Berberette et al., 2023). Moreover, the quality of interaction between students and LLMs heavily depends on the quality of prompts provided. Ambiguous or underspecified prompts often result in responses that do not address the actual problem students face (Simaremare et al., 2024).

2.4. RAG

To mitigate hallucination and improve accuracy, some studies have applied RAG. RAG combines the generative capability of LLMs with an external retrieval mechanism that fetches relevant documents as grounding sources before the model generates its response (Gheorghiu, 2024).

Although RAG theoretically improves factual accuracy, its use in programming education especially for debugging tasks remains challenging. Prior research indicates that during debugging, RAG does not consistently contribute useful external context (Simaremare et al., 2025). Retrieved documents are often not effectively incorporated into the model's reasoning, resulting in minimal improvement in feedback quality. This suggests a need for further analysis of how LLMs interpret and utilize retrieved context when analyzing and correcting source code.

2.5. Challenges Students Face When Constructing Prompts

Beyond technological considerations, human factors play an important role in the effectiveness of GenAI-assisted learning. Many students struggle to craft effective prompts. They often provide instructions that are too general, non-contextual, or insufficiently specific to the problem at hand. Since prompt clarity directly influences the quality of responses, poorly constructed prompts frequently lead to irrelevant or suboptimal answers from the model (Simaremare et al., 2024).

This limitation in prompt formulation poses a key obstacle to the adoption of GenAI in programming education. Therefore, it is important to investigate how prompt characteristics influence the quality of responses students receive, and how guidance on prompt construction can improve learning outcomes.

3. Discussion

The review of existing literature reveals several critical issues underlying the use of LLM-based systems in programming education. LLMs offer strong potential as interactive learning tools, yet their effectiveness varies widely based on student proficiency, prompt clarity, and contextual understanding.

Novice learners frequently struggle to craft precise prompts because they lack foundational knowledge about programming concepts. Their interactions with LLMs often result in generic explanations that fail to address the core difficulty. In contrast, advanced students are better equipped to pose detailed and contextual questions, enabling LLMs to generate more relevant responses. This difference demonstrates that the value of GenAI tools depends not only on the model's capabilities but also on the student's ability to communicate the problem clearly.

Hallucination remains an important concern when GenAI is used for coding assistance. Inaccurate or misleading responses may go unnoticed by beginners, potentially reinforcing misconceptions. Without a verification mechanism or grounding strategy, students may extend these misunderstandings into subsequent learning activities.

The adoption of Retrieval-Augmented Generation aims to provide additional context by supplying external documents, yet its effectiveness in programming tasks is still limited. Even when relevant documents are retrieved, LLLMs do not consistently integrate them into their reasoning. This indicates that improvements in retrieval alignment and model-context integration are necessary for RAG to become more impactful in educational settings.

Prompting difficulties also emerge as one of the most frequently cited issues in GenAI-supported learning. Students' ability to articulate their understanding and learning needs affects how the model interprets the problem. Teaching adaptive prompting strategies or integrating prompt-refinement mechanisms may help address this challenge.

Taken together, these findings highlight the importance of developing more adaptive and personalized GenAI-based learning systems. Students possess diverse levels of prior knowledge, and a one-size-fits-all model cannot accommodate the full spectrum of learning needs. Effective systems must therefore incorporate personalization that reflects student proficiency, learning context, and the nature of the task.

4. Conclusions

This study outlines the major issues surrounding the adoption of GenAI, particularly LLMs, in programming education. Variations in student skill levels significantly influence how learners interact with LLMs and how effectively they interpret model responses. Novice learners often face difficulties in prompt construction and conceptual verification, whereas advanced learners benefit more readily from the models' capabilities.

The tendency of LLMs to hallucinate remains a persistent obstacle, amplifying the risk of conceptual misunderstandings during code interpretation or debugging. Verification mechanisms and clearer grounding strategies are necessary to reduce this risk.

The integration of Retrieval-Augmented Generation has not yet demonstrated consistent effectiveness in programming contexts, as retrieved documents are not always incorporated meaningfully into the model's reasoning. This suggests the need for improved alignment between retrieval systems and generative components.

Overall, the findings demonstrate that implementing GenAI in programming education must prioritize adaptability, contextual understanding, and safety. Effective instructional strategies should integrate prompt-refinement support, provide mechanisms for verification, and identify conditions under which RAG contributes meaningfully to the learning process.

5. Recommendations and Future Work

The findings from this study indicate several important directions for improving the use of GenAI in programming education. Future systems should provide more adaptive support that matches students' different levels of programming proficiency. Models need the ability to adjust explanations and guidance based on whether the learner is a beginner or an advanced student.

Improvements in prompt support are also necessary. Many learners struggle to formulate clear prompts, so future tools should help refine or clarify user queries before generating responses. This can ensure that students receive more accurate and relevant explanations.

Mitigating hallucination remains another essential focus. Systems should include mechanisms to verify generated code or explanations, reducing the risk of incorrect information that may mislead learners.

Enhancing Retrieval-Augmented Generation is equally important. Future work should explore better ways for models to incorporate retrieved documents so that external context contributes meaningfully during debugging or code analysis tasks.

Overall, further development should emphasize adaptive guidance, safer responses, and more effective use of contextual information. These improvements can support more reliable and personalized learning experiences for students who rely on GenAI in understanding programming.

6. References

Annuš, N. (2025). *Investigation of Generative AI Adoption in IT-Focused Vocational Secondary School Programming Education*.

Berberette, E., Hutchins, J., & Sadovnik, A. (2023). *Redefining "Hallucination" in LLMs: Towards a psychology-informed framework for mitigating misinformation*.

Gheorghiu, A. (2024). *Building Data-Driven Applications with LlamaIndex*.

Helen, F., & Nada, D. (2024). *DESIGNING LEARNING EXPERIENCES A FRAMEWORK FOR HIGHER EDUCATION AND Learning Experiences*. Routledge. https://doi.org/10.4324/9781003121008

Nurollahian, S., Rafferty, A. N., & Brown, N. (2024). *Growth in Knowledge of Programming Patterns : A Comparison Study of CS1 vs . CS2 Students*. 979–985. https://doi.org/10.1145/3626252.3630865

Prather, J. (2024). *Evaluating Contextually Personalized Programming Exercises Created with Generative AI*. 95–113. https://doi.org/10.1145/3632620.3671103

Simaremare, M., Pardede, C., Tampubolon, I., Simangunsong, D., & Manurung, P. (2024). Pair Programming in Programming Courses in the Era of Generative AI : Students ' Perspective. *2024 31st Asia-Pacific Software Engineering Conference (APSEC)*, 507–511. https://doi.org/10.1109/APSEC65559.2024.00069

Simaremare, M., Pardede, S., Purba, E., Pangaribuan, S., & Siagian, J. (2025). *THE IMPLEMENTATION OF GENERATIVE AI FOR PERSONALIZED LEARNING THROUGH THE RETRIEVAL AUGMENTED GENERATION (RAG) AND PERSONA APPROACHES*.

Solanki, S. R., & Khublani, D. K. (2024). *Generative Artificial Intelligence*.