CodeCatcher: Enhancing Novice Debugging Performance with AI-Assisted Hinting

Piyush Girish Deshpande

Purdue University

Abstract

The process of debugging stands as a major mental challenge for new programmers which produces frustration together with delayed progress and student disengagement. This research presents CodeCatcher as a Python-based debugging environment that uses Gemini large language model (LLM) to assess real-time structured AI-generated hints in educational settings. A controlled experiment involved 15 undergraduate programming students who were divided into two groups: one received conceptual hints from Gemini while the other group worked independently without any assistance. The AI-assisted group outperformed the independent group in task completion rates (87.5% vs. 50%) and task duration (302s vs. 392s) while showing lower cognitive load according to the simplified NASA-TLX scale. The analysis of hint interactions shows that specific and well-worded queries help students solve problems more effectively. The research evidence supports the use of LLM-based hinting systems as reflective scaffolding tools which promote metacognition and decrease mental effort while improving debugging performance in time-constrained learning environments. The research demonstrates how AI-augmented education can revolutionize programming instruction by providing appropriate guidance while preserving student independence.

The recent advancements in artificial intelligence (AI) through large language models (LLMs) have brought about substantial changes in programming education methods. AI serves as an educational cognitive partner through its implementation in conversational agents within Integrated Development Environments(IDEs) and intelligent tutoring systems(ITS) that provide adaptive scaffolding. The recent advancements present an opportunity to transform debugging support systems because this activity remains one of the most mentally challenging tasks for new programmers.

The current methods of teaching debugging lack essential features of personalized instruction and immediate feedback and scaffolding which help beginners learn hypothesis-based problem-solving. The research community now actively investigates how artificial intelligence systems can decrease mental workload and boost student motivation while improving educational results through tutoring and partnership and co-programming capabilities. The implementation of AI systems in educational settings creates essential questions about whether AI-generated hints decrease cognitive burden or simply relocate it. Students have different reactions to AI feedback compared to instructor feedback. The most successful AI interaction models for time-limited and affective situations remain unclear.

The review compiles recent research from multiple subdomains which includes LLM-based tutoring and cognitive load in AI-supported learning as well as student feedback perception and human-AI collaboration to establish the foundation for our study. The following sections use peer-reviewed research to analyze AI-assisted debugging systems through pedagogical, behavioral and technical lenses before

presenting the case for evaluating structured hinting through LLMs in time-constrained learning environments.

Review of Literature

The Pedagogical Value of Debugging Practice with LLMs

The adoption of large language models (LLMs) as educational tools brings new possibilities to construct structured learning experiences about debugging. The HypoCompass system proposed by Ma et al. (2024) introduces a "learning-by-teaching" approach which trains students to help simulated LLM-agents detect bugs through hypothesis development despite this being an essential yet poorly instructed debugging skill. The evaluation demonstrates a 12% better debugging performance and a 14% shorter task duration thus showing LLMs as effective scaffolds over direct answer providers.

The method suggests debugging extends past syntax fixing because it requires a step-by-step mental procedure to develop and validate hypotheses.

Real-Time AI-Augmented Debugging Systems

Levin et al. (2024) proposed ChatDBG which represents an emerging class of conversational AI debuggers that maintain direct interaction with code execution state. It differs from typical LLMs by linking with runtime environments including LLDB and IPython which enable exact feedback that stems from actual program state analysis. Levin et al. prove that ChatDBG boosts error diagnosis by 180% while their research shows how students' clarification steps enhance their correction performance rates.

Thus, emphasis should be put on structured hinting through LLMs supported by evaluations based on learner interactions with hints and solution efficiency measurement.

Cognitive Load in AI-Supported Programming

The integration of AI into educational environments faces an essential challenge related to cognitive load. The research conducted by Haque et al. (2025) investigates how artificial intelligence assistance impacts cognitive load by determining whether it decreases mental work or just repositions it. The researchers used EEG and self-reporting methods to discover that developers including beginners experience heightened mental workload when they use AI to generate suggestions for code even though they report increased productivity.

More examinations are required in hint-query volume together with task completion within time limits to measure cognitive load indirectly.

Instructor vs. AI-Generated Feedback

AI offers scalable feedback but students demonstrate complex reactions to its acceptance. Er et al. (2024) analyzed how students reacted to instructor comments versus computer-generated feedback. The students valued the AI feedback because it provided consistent responses quickly yet they chose instructors to explain abstract concepts and ambiguous errors.

The research supports the development of AI tools which foster reflective inquiry or hints instead of direct instruction or solution.

ITS Systems for Programming Education

The survey conducted by Fodouop Kouam (2024) about Intelligent Tutoring Systems (ITS) explains how these tools guide students through adaptive feedback as well as semantic modeling and personalized remediation. The research establishes that properly designed ITSs lead to substantial learning gains in introductory courses because they focus on cognitive engagement.

Fan et al. (2025) presented a quasi-experimental study examines 234 undergraduate students through AI-assisted pair programming compared to regular pair and individual programming approaches. The research results show AI-assisted pair programming boosts students' intrinsic motivation while decreasing their programming anxiety to levels equal to human–human pair programming.

Sakshm AI presented by Gupta et al. (2025) is an intelligent tutoring system through its implementation of Socratic tutoring and thorough feedback to boost coding education. The research evaluates the system's effects on problem-solving abilities and student involvement among engineering students as it provides information about large-scale intelligent tutoring systems**.**

The current researches do not target error localization and hypothesis development to get improved results.

Learner Behavior and Affective Feedback Cycles

Lepp and Kaimre (2025) present qualitative research findings from CS students which demonstrate how AI tools excel at error detection and contextual understanding and motivational support. The students felt safer asking stupid questions to AI systems and enjoyed the freedom to try again without criticism which stood as advantages compared to conventional classroom instruction.

AI-provided social-emotional support seems to lower the mental burden of affective cognitive load which is typically worsened by traditional time-limited assessments.

Chatbots as Scaffolding Partners

Game-based AI chatbot environments were tested by Xu et al. (2024) for their effectiveness in improving motivation accuracy and higher-order thinking in IT learners. Importantly, low achievers benefited the most a group that often struggles with debugging independently. The design choice to examine learner behavior in both AI hint presence and absence under time pressure receives validation from these findings.

Visual Explanations and Metacognitive Structuring

Lepp and Kaimre stress that LLMs provide more than just code fixes, they help structure thinking, especially through visual explanations and step-by-step guidance. Numerous students characterized the AI system as a “private tutor” which supports exploratory learning by allowing students to conduct multiple cycles of trial and reflection.

MindScratch is a multimodal generative AI-powered visual programming tool designed by Chen et al. (2025) to support classroom learning. This tool combines controlled activities with free-form programming while developing students’ computational abilities and creative thinking.

AI Integration in IDEs and Human-AI Experience (HAX)

The systematic literature review by Sergeyuk et al. (2025) discusses the growth of AI-enhanced Integrated Development Environments (IDEs) and introduces the Human-AI Experience (HAX) concept. The authors’ evaluation of 89 studies showed

that two main paradigms of AI usage in IDEs exist which include autocompletion (e.g., GitHub Copilot) and conversational agents with hybrid models gaining popularity.

The review points out that AI enhances both productivity and flow but introduces verification overhead and automation bias and over-reliance particularly among novices. The study findings match the primary objective of this research's debugging assistant to provide structured hints for problem-solving instead of generating automatic solutions.

The review by Sergeyuk et al. (2024) examines 36 studies by organizing them into three main research branches which include Design, Impact, and Quality of Interaction. This shows the dynamic software development field with AI integration in IDEs and proposes further research topics including task-specific interface design, trust-building mechanisms, and readability improvement. The review notes that educational context studies and personalization strategies remain underrepresented in the current research which this study addresses by studying novice performance in a learning environment while measuring time-constrained cognitive outcomes.

Collectively, the current studies demonstrate that LLM-powered systems which are properly integrated enhance debugging effectiveness and reduce learner frustration while improving metacognitive skills. Hint structure and feedback framing together with learner agency determine the effectiveness of this approach. The evaluation of AI-guided hinting which takes place under controlled timing and hint-query constraints directly tackles the cognitive behavioral and affective dimensions in this research.

Based on the insights drawn from the literature, this study investigates the following research question.

RQ: Does AI-assisted hinting reduce the cognitive load and error rate in debugging tasks for novice programmers under time-constrained conditions?

The question guides the experimental design, which compares students receiving real-time, structured AI hints through Gemini with those solving the same debugging task without assistance.

Method

Fifteen undergraduate students enrolled in an introductory programming course participated in the study. All participants had less than one year of formal programming experience. The students were randomly assigned to one of two groups: an AI-assisted experimental group (n = 8) and a non-AI control group (n = 7).

Materials

The primary tool used in the study was **CodeCatcher**, a Python-based debugging environment developed specifically for this experiment. The system presented students with buggy C++ code and allowed them to either submit corrected code or, in the AI-assisted group, interact with Gemini AI to request natural language hints. Hints were designed to guide students conceptually without revealing the exact solution. Each session featured one medium-difficulty debugging task and a fixed time limit of eight minutes.

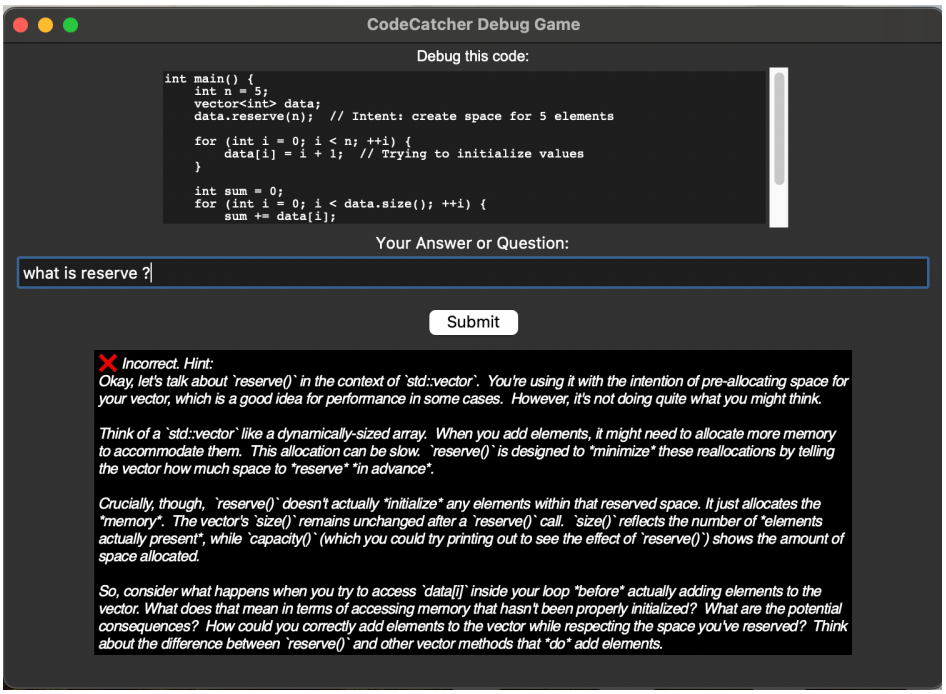

Figure 1 The CodeCatcher GUI: A debugging environment showing the code prompt, user input field, and real-time AI-generated hint.

As shown in **Figure 1**, the CodeCatcher interface provided an intuitive platform for students to view buggy code, submit responses and receive conceptual hints in real time.

Design

A **between-subjects experimental design** was implemented to compare the performance of students with and without AI assistance. The two conditions were as follows

Experimental Group (AI-assisted): Students used CodeCatcher with Gemini-powered hints enabled.

Control Group (Non-AI): Students used CodeCatcher without any AI assistance or hint system.

Procedure

All participants first completed a short pre-task survey to self-report their confidence in debugging. Students were then given a brief introduction to the CodeCatcher interface. Each participant attempted the debugging task individually within the allocated eight-minute time window.

Students in the AI-assisted group were permitted to ask up to five natural language questions to the Gemini model. These interactions were tracked, along with the number of attempts and time taken to arrive at the correct solution. Following task completion, all participants completed a post-task questionnaire to evaluate their perceived cognitive load, engagement, and confidence using a simplified NASA-TLX scale.

Measures

Completion Success: Whether the student corrected the bug successfully within the time limit (Yes/No)

Number of Hint Queries: Total number of questions asked to the Gemini system (experimental group only)

Time to Solve: Time taken (in seconds) to submit the correct fix or time expired

Cognitive Load: Measured using a simplified post-task NASA-TLX scale

Results

The research findings evaluated students' debugging performance and their cognitive load and efficiency when using AI-assisted hinting. A total of 16 students participated in the study, equally divided between the experimental group (AI-assisted) and the control group (non-AI). Three key metrics were examined: completion success, time to solve, and self-reported cognitive load. The researchers monitored how many

hint queries students in the AI group needed. Completion Success The AI-assisted group achieved successful code corrections in 87.5% of cases within the time limit. The control group achieved successful debugging task completion by only 50% of its students. Students who received Gemini-generated hints demonstrated higher success rates when working under time constraints. Time to Solve The average time to completion in the AI-assisted group was 302 seconds (SD = 48). The control group spent an average of 392 seconds (SD = 55) on the debugging task before reaching the time limit. The AI group achieved superior results while requiring less time to complete their tasks on average. Hint Interaction (AI Group Only) Students who received AI assistance needed an average of 3.25 hint queries (SD = 1.2) to finish the task. The analysis revealed a moderate negative relationship between hint queries and task completion speed (r = -0.46) which means students who used focused questions solved faster. Cognitive Load Self-reported NASA-TLX cognitive load scores averaged: AI-assisted group: 3.1 / 5 (Lower perceived effort) Control group: 4.0 / 5 (Higher frustration and effort reported) The findings confirm that AI-based structured assistance helps decrease both mental workload and emotional stress during debugging tasks.

Discussion

The results from this study support the hypothesis that hinting can enhance novice programmers' debugging performance while reducing their cognitive load. Students in the AI-assisted group performed better on the task, achieved the task faster, and experienced lower cognitive strain than students in the control group.

Debugging Effectiveness

The 37.5% improvement in task completion rate between the AI-assisted and control groups aligns with prior findings from tools like HypoCompass and ChatDBG, which emphasize the importance of conceptual scaffolding in debugging environments. The Gemini-powered hints appeared to offer just enough structure to guide reasoning without overwhelming learners or giving away the solution. This supports literature emphasizing the pedagogical value of LLMs as reflective tutors rather than answer providers.

Time and Query Efficiency

The reduction in average time-to-solution and the observed correlation between hint usage and task speed reinforce the potential of real-time AI feedback to facilitate focused problem solving. Students who engaged meaningfully with the hint system appeared to iterate faster, possibly due to reduced uncertainty and more targeted debugging strategies.

Cognitive Load Reduction

The difference in NASA-TLX scores suggests that AI-assisted students perceived the debugging task as less mentally taxing. This complements prior work (Haque et al.), which warns that while AI tools can boost productivity, they may also create additional verification strain. In this study, however, the structure of the hints (limited, natural language-based, and scaffolded) may have helped balance guidance with autonomy, reducing both mental effort and affective load.

Implications for Educational Practice

The findings demonstrate the value of embedding LLMs into educational debugging tools in a way that prioritizes learning, not just performance. CodeCatcher's

design constrained hint interactions and no direct fixes promotes metacognition and minimizes over-reliance. Educators could adopt similar approaches to provide on-demand support at scale while preserving the integrity of formative assessment.

Conclusion

This study explored the impact of AI-assisted hinting on novice programmers' debugging performance, efficiency, and cognitive load within a time-constrained environment. Using a custom-built Gemini-powered educational tool, **CodeCatcher**, we compared students who received structured, non-revealing AI-generated hints with those who completed the same task independently.

The findings indicate that AI-assisted hinting can significantly improve both task completion rates and time-to-solution while also reducing perceived cognitive effort. Students in the experimental group not only performed better but also reported lower mental demand and frustration levels, reinforcing the potential of LLM-driven scaffolding in early programming education. By structuring the AI interaction around reflective inquiry rather than direct solution delivery, this study contributes to a growing body of work advocating for intelligent tutoring systems that promote metacognition, confidence, and independent problem-solving. While promising, these results should be validated with larger, more diverse samples and extended to longitudinal studies to assess long-term learning outcomes. Ultimately, this work supports the integration of LLM-powered hinting systems into introductory programming instruction not as a replacement for human feedback, but as a scalable and pedagogically sound augmentation to traditional teaching methods.

This study is exploratory and based on a small sample size. While effect sizes are promising, broader validation is needed across diverse student populations and programming languages. Additionally, the current Gemini model's ability to understand nuanced student questions may vary, and future systems should adaptively model student knowledge to further personalize feedback.

Follow-up studies could explore the long-term impact of AI hinting on independent debugging ability. Logging deeper metadata like time between hints, hint content type, or emotional sentiment could also help fine-tune LLM-assisted systems for education. Expanding this study into a longitudinal format would help assess retention, transferability, and scaffold fading effects.

References

Ma, Q., Shen, H., Koedinger, K., & Wu, S. T. (2024, July). How to teach programming in the ai era? using llms as a teachable agent for debugging. In *International Conference on Artificial Intelligence in Education* (pp. 265-279). Cham: Springer Nature Switzerland.

Levin, K., van Kempen, N., Berger, E. D., & Freund, S. N. (2024). Chatdbg: An ai-powered debugging assistant. *arXiv preprint arXiv:2403.16354*.

Haque, E. A., Brown, C., LaToza, T. D., & Johnson, B. (2025). Towards Decoding Developer Cognition in the Age of AI Assistants. *arXiv preprint arXiv:2501.02684*.

Er, E., Akçapınar, G., Bayazıt, A., Noroozi, O., & Banihashem, S. K. (2024). Assessing student perceptions and use of instructor versus AI-generated feedback. *British Journal of Educational Technology*.

Fodouop Kouam, A. W. (2024). The effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience. *Discover Education*, *3*(1), 1-13.

Lepp, M., & Kaimre, J. (2025). Does generative AI help in learning programming: Students' perceptions, reported use and relation to performance. *Computers in Human Behavior Reports*, *18*, 100642.

Xu, Y., Zhu, J., Wang, M., Qian, F., Yang, Y., & Zhang, J. (2024). The impact of a digital game-based AI chatbot on students' academic performance, higher-order thinking, and behavioral patterns in an information technology curriculum. *Applied Sciences*, *14*(15), 6418.

Sergeyuk, A., Zakharov, I., Koshchenko, E., & Izadi, M. (2025). Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. *arXiv preprint arXiv:2503.06195*.

Sergeyuk, A., Titov, S., & Izadi, M. (2024, April). In-ide human-ai experience in the era of large language models; a literature review. In *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments* (pp. 95-100).

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*(1), 16.

Gupta, R., Goyal, H., Kumar, D., Mehra, A., Sharma, S., Mittal, K., & Challa, J. S. (2025). Sakshm AI: Advancing AI-Assisted Coding Education for Engineering Students in India Through Socratic Tutoring and Comprehensive Feedback. *arXiv preprint arXiv:2503.12479*.

Chen, Y., Xiao, S., Song, Y., Li, Z., Sun, L., & Chen, L. (2025). MindScratch: A Visual Programming Support Tool for Classroom Learning Based on Multimodal Generative AI. *International Journal of Human–Computer Interaction*, 1-19.