INTERNATIONAL RESEARCH JOURNAL OF ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET)

eISSN: 3087-4068

Volume. 02, Issue. 10, pp. 91-101, October 2025"

EVALUATING CONVERSATIONAL AND PLATFORM-INTEGRATED GENERATIVE AI FOR AUTOMATED, TIMELY FEEDBACK IN PROGRAMMING EDUCATION: A QUASI-EXPERIMENTAL STUDY UTILIZING GPT-4O-MINI

Prof. Kenji A. Takada

Faculty of Computer Science and Engineering, Meiji University of Technology, Tokyo, Japan

Article received: 14/08/2025, Article Revised: 21/09/2025, Article Published: 31/10/2025

ABSTRACT

Context: Effective feedback is critical for novice programmers, but providing it in a timely and scalable manner poses a significant challenge in higher education [13], [14], [37]. Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs) trained on code [9], [36], offers a promising avenue to automate this process [1], [22].

Objectives: This quasi-experimental study aimed to evaluate the usability, student perceptions, and academic impact of two distinct GenAI-assisted feedback tools, both powered by GPT-4o-mini: a conversational assistant (tutorB@t) and a platform-embedded tool integrated with a virtual code evaluator (tutorBot+).

Methods: The study involved 91 undergraduate computer science students, with 37 assigned to the experimental AI-assisted group. We measured student programming performance, passing rates, and user perception using the System Usability Scale (SUS) [6] to assess the perceived utility and ease of use of the developed tools.

Results: Students highly valued the immediacy and accessibility of the AI feedback. Perception scores were positive, with tutorB@t achieving a SUS score of 70.6 and tutorBot+ scoring 65.2, and a high intent to reuse (81% and 79%, respectively). Crucially, despite positive perceptions, the study found no statistically significant difference in objective programming performance or passing rates between the groups. This outcome is attributed primarily to factors such as a lack of group homogeneity, external academic pressures, and occasional student misunderstanding of the GenAI-provided feedback.

Conclusion: Timely, automated feedback from GenAI is highly valued by students for its accessibility. Yet, the current study suggests that design limitations (usability, student misunderstandings, external factors) may mask the direct academic impact, highlighting a need for refined integration and future research incorporating affective measures [15], [38] to fully understand and unlock the pedagogical potential of LLM-based feedback [33].

KEYWORDS

Programming Education, Generative AI, GPT-4o-mini, Automated Feedback, Timely Feedback, Quasi-Experimental Study, Usability.

INTRODUCTION

1.1. Background: The Critical Role of Feedback in Programming Education

Learning to code is fundamentally a recursive process that relies heavily on iterative practice, failure, and correction. For novice programmers, the journey from a conceptual understanding of an algorithm to a successful, working implementation is fraught with common pitfalls, logical errors, and syntax struggles. In this context, feedback is not merely a mechanism for grading; it is the essential catalyst for learning, development, and eventual mastery [13].

Effective feedback must possess three crucial qualities: it needs to be timely, arriving when the student is still focused on the problem; it must be targeted, addressing the specific misconception or error; and it needs to be

actionable, providing guidance on how to improve rather than just what is wrong [13], [25]. When feedback is delayed, students often move on to other tasks, and the cognitive link between the error and the correction is weakened, severely diminishing its pedagogical power [2]. Conversely, when feedback is prompt and high-quality, it is associated with significantly enhanced student learning outcomes and self-efficacy [37], [14].

However, the reality of programming education in modern higher education settings—characterized by large class sizes and resource constraints—means that providing this level of personalized, high-quality, and timely feedback consistently is a massive logistical challenge [14]. Human instructors struggle to keep up with the volume of code submissions, leading to significant delays in grading and personalized guidance. This systemic delay creates a critical educational bottleneck, prompting the search for automated solutions.

1.2. The Emergence of Generative AI (GenAI) in Education

The advent of sophisticated Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs), has offered a compelling and potentially transformative solution to this scaling problem [40], [18]. LLMs like those developed by OpenAI have demonstrated a remarkable ability to understand, generate, and, critically for this domain, reason about code [9], [36]. Evaluations suggest these models can successfully pass technical interviews, generate novel code snippets, and even fix complex bugs [36].

This capacity represents a significant step beyond earlier automated educational tools, such as basic error checkers or rule-based Intelligent Tutoring Systems (ITS) [16], [30]. Whereas prior systems relied on pre-defined knowledge bases and structured rules, GenAI offers the flexibility to synthesize human-like, natural language explanations and corrections tailored to a student's unique code submission and context [5]. This allows LLMs to function not just as evaluators but as genuine virtual tutors [4], capable of engaging in dialogue and offering nuanced, context-sensitive support.

1.3. LLMs as Automated Feedback Tools

The application of LLMs to generate feedback for programming assignments has recently become a prolific area of research [1]. Studies have explored their efficacy in automating the grading process [22], creating high-quality hints [24], and serving as conversational assistants [10], [11].

These approaches generally fall into two categories, which inform the design of the tools investigated here:

1. Conversational Chatbots: Tools like tutorB@t operate as standalone assistants, allowing students to paste code or ask general questions. They prioritize interactive dialogue and context-based guidance [4], [39].

2. Platform-Integrated Systems: Tools like tutorBot+ embed the GenAI directly into the learning management or programming environment. They typically ingest compiler errors or test results from an evaluator (like Judge0) and use this specific data to generate highly targeted feedback [27], [28].

Recent advancements have made LLMs more accessible. The introduction of models like GPT-4o-mini [21] provides a high-performing, yet cost-efficient, option. This model boasts significant reasoning capabilities compared to earlier generations while drastically reducing the computational cost per interaction. This makes the implementation of high-volume, real-time feedback systems pedagogically and economically feasible for large university programs.

1.4. Identifying the Research Gap

While the promise of GenAI in Computer Science education is clear, two critical gaps persist in the current literature:

First, there is a distinct lack of comparative studies that simultaneously evaluate the efficacy of different LLM deployment mechanisms—specifically, a conversational chatbot versus a platform-embedded, non-conversational tool—within a single controlled educational context. Understanding how the interface design is associated with student interaction, perception, and learning outcomes is crucial for practitioners and tool developers [19].

Second, despite overwhelming evidence that students enjoy using GenAI tools and find them helpful [32], there remains a persistent uncertainty regarding whether this high satisfaction consistently predicts measurable, objective gains in academic performance. Many studies focus on subjective usability or qualitative engagement, leaving the core question of academic impact unanswered.

This study directly addresses these gaps by implementing and testing two distinct feedback tools—tutorB@t and tutorBot+—both powered by the cost-effective yet powerful GPT-4o-mini [21], within a quasi-experimental setting.

1.5. Research Questions and Study Overview

This research sought to answer the following questions, using language appropriate for a correlational study:

1. Is the perceived usability of both conversational

and platform-integrated GenAI feedback tools (tutorB@t and tutorBot+) acceptable to students, as measured by standardized usability metrics?

2. Is the provision of automated, timely GenAI feedback, regardless of deployment method, associated with a statistically significant difference in student programming performance (e.g., passing rates or test success) compared to traditional methods?

The remainder of this article details the pedagogical foundations underpinning effective feedback, the architectural design of the two tools, the methodology of the quasi-experimental study, the quantitative and qualitative results, and a robust discussion of the implications of the findings.

II. Related Work

2.1. Pedagogical Foundations of Effective Programming Feedback

The philosophical foundation of our tool design rests on established principles of educational psychology and feedback theory. Hattie and Timperley [13] emphasized that effective feedback must answer four key questions: Where am I going? (goals), How am I going? (progress), Where to next? (directions), and critically, How can I close the gap? (actionable steps). GenAI is uniquely positioned to help students close the gap by providing highly specific, what-to-do-next advice directly related to their current code [33].

Furthermore, deep learning requires students to develop metacognition—the ability to monitor and regulate one's own thinking and learning processes [29], [26]. In programming, this involves debugging skills, self-checking assumptions, and planning solutions. AI-driven feedback, unlike simple pass/fail tests, can foster metacognition by asking guiding questions or providing hints that nudge the student toward the solution without simply giving it away.

Finally, we consider motivational design [15]. John Keller's ARCS Model (Attention, Relevance, Confidence, Satisfaction) suggests that technology must maintain a student's engagement to be effective [15]. Timely, personalized, and constructive feedback delivered by the AI addresses the "Confidence" and "Satisfaction" components by reducing the time spent blocked on trivial errors and providing a sense of personalized support, which can be highly motivating [38].

2.1.3. Feedback and the Cultivation of Metacognition: The Productive Struggle

The process of debugging and problem-solving in programming is fundamentally a metacognitive exercise. Metacognition—often defined as 'thinking about thinking'—encompasses a set of cognitive skills vital for self-regulated learning and error recovery [29]. Schraw and Dennison defined metacognitive awareness along two dimensions: knowledge of cognition (what a learner knows about their own cognitive resources and strategies) and regulation of cognition (how a learner monitors and controls their learning processes) [29].

Effective feedback in programming must therefore target not just the solution, but the student's process of reaching that solution, promoting regulation of cognition through four distinct phases:

1. Planning: Setting goals and selecting appropriate strategies (e.g., choosing a loop structure).

2. Monitoring: Tracking progress, comparing current state to goals, and identifying obstacles (e.g., recognizing that a program is stuck in an infinite loop).

3. Evaluation: Assessing the outcome of a strategy and determining if the final solution is correct and efficient (e.g., running the program and analyzing the output).

4. Revision: Adjusting the plan and strategies based on the evaluation (i.e., debugging).

Failure to achieve a solution often results in an impasse, which VanLehn argued is the critical precursor to learning [34]. An impasse-driven learning model suggests that the most profound learning occurs when a student recognizes that their current knowledge or strategy is insufficient, forcing them to engage in self-repair and knowledge revision.

The critical challenge for automated feedback systems, including GenAI, is to interrupt this impasse in a productive way. This is known as managing the Productive Struggle—providing just enough information to keep the student motivated (preventing abandonment) while withholding the complete solution to ensure they perform the necessary cognitive work (preventing over-reliance) [33].

Traditional ITS systems managed this productive struggle via carefully curated hint sequences, often relying on structured rule sets or cognitive models [34]. In contrast, GenAI systems generate hints and explanations dynamically using natural language. This flexibility presents both a strength and a weakness:

- Strength: GenAI can generate a wider variety of responses and adapt the tone, aligning with the "human tutor-style" feedback advocated by some researchers [24], [33].
- Weakness: The LLM's output is not inherently

optimized for metacognitive scaffolding. Without stringent prompt engineering, the feedback may become too verbose, increasing the student's extraneous cognitive load (information processing unrelated to learning the core concept) [33]. Alternatively, the LLM may inadvertently provide information that bypasses the productive struggle entirely, potentially leading to a quick fix but no deep learning [7].

Therefore, the efficacy of our two tools—the conversational tutorB@t and the integrated tutorBot+—must be analyzed through the lens of which phase of metacognitive regulation they most effectively support and whether their interface design inherently manages the productive struggle. The conversational approach (tutorB@t) is often associated with supporting the Planning and Monitoring phases by allowing students to articulate their thought process and receive general conceptual guidance. Conversely, the integrated approach (tutorBot+) is primarily driven by compiler errors and test case failure, making it highly suitable for supporting the Evaluation and Revision phases, as its feedback is explicitly grounded in the execution outcome.

The tension between these two models—flexibility and conceptual guidance (tutorB@t) versus rigid, execution-grounded diagnosis (tutorBot+)—is central to understanding why high usability did not translate to performance gains in this study. If the AI is used to avoid the necessary productive struggle, then improved performance remains elusive, regardless of how friendly or accessible the tool is.

2.2. Automated Feedback Systems and Intelligent Tutoring Systems (ITS)

The concept of using technology to automate instruction and feedback has a long history, dating back to early Intelligent Tutoring Systems (ITS) [30]. These systems, built on cognitive models and production rules, aimed to provide individualized instruction and detailed error diagnosis [34]. Classic examples include systems that tracked student attempts and provided targeted hints based on pre-defined knowledge rules [16].

However, traditional ITS often suffered from a knowledge acquisition bottleneck—building and maintaining the domain knowledge base necessary to cover all possible student errors was incredibly labor-intensive and challenging to scale across new problems or programming languages.

GenAI fundamentally alters this landscape. Rather than relying on explicit programming of error models, LLMs leverage vast training data to infer common errors and generate novel, human-like diagnostic explanations [5]. This flexibility positions LLMs as "domain-general" tutors that can adapt instantly to new problems, thereby solving the scale and maintenance issues that plagued previous automated systems [33]. Studies suggest LLMs can successfully automate feedback and even generate hints styled after human tutors [24].

2.3. GenAI Implementation in Computer Science Education

Current research highlights a bifurcation in how LLMs are applied to programming instruction:

Conversational Interfaces (tutorB@t Model): Tools implemented as chatbots (e.g., ChatGPT, or course-specific bots [10], [11]) are highly praised for their low barrier to entry and their ability to sustain a human-like dialogue [39]. They excel at providing immediate, flexible clarification on concepts, interpreting error messages, and suggesting general debugging strategies [4]. The student controls the conversation, which is often associated with a high sense of autonomy, potentially leading to increased satisfaction.

Platform-Integrated Systems (tutorBot+ Model): Integrating GenAI directly into the development or learning environment provides key advantages in terms of context grounding. By piping the student's exact code, the compiler output, and the test case results directly into the LLM's prompt, the system ensures the AI's response is highly relevant and avoids generic advice [27], [28]. Systems that debug code by verifying runtime execution step-by-step [41] demonstrate the power of deep platform integration. Our tutorBot+ tool follows this model by embedding GenAI into the existing flow of code submission and evaluation, prioritizing the provision of non-conversational, focused feedback.

2.4. LLM Selection and Benchmarking

The selection of the LLM for this study, GPT-4o-mini [21], was a strategic decision balancing performance, cost, and accessibility. While early GenAI research often focused on flagship models, the long-term sustainability of automated feedback for large student populations requires models that are both capable and economically viable.

GPT-4o-mini is advertised as a model offering improved speed and intelligence at a fraction of the cost of its predecessors [21]. For code tasks, the primary requirement is strong logical reasoning and adherence to instructions, which is essential for accurate error diagnosis. General benchmarks for code models (e.g., [9], [36], [42]) demonstrate rapid improvements in the field. The use of a cost-efficient LLM allows this study to test the feasibility of deploying automated feedback at scale, which is crucial for real-world academic adoption. Furthermore, studies like LiveBench [35] emphasize the need for ongoing evaluation, justifying our effort to test a recent model iteration.

2.5. Ethical and Design Considerations for AI Feedback

The introduction of powerful GenAI tools into education is not without its challenges. Chief among these are concerns related to plagiarism (students using the AI to solve problems rather than learn) and over-reliance [7], [19]. To mitigate these risks, the design of effective AI feedback must adhere to principles of responsible development [17], specifically by focusing on providing hints and explanations rather than complete solutions. The goal must be to promote learning, not to facilitate cheating.

Furthermore, any deployed educational technology must be accessible. Future development must align with standards such as the W3C Authoring Tool Accessibility Guidelines (ATAG) to ensure the tools benefit all learners, avoiding the creation of new accessibility barriers [17]. Assessing the usability, as measured by the SUS [6], forms the initial step in ensuring a positive and accessible student experience.

III. Methods

3.1. Study Design and Participants

This research utilized a quasi-experimental design conducted over a single academic semester within an introductory programming course at a university. A true randomized controlled trial was deemed impractical due to the ethical constraint of withholding a potentially beneficial learning resource from an entire group and the need to integrate the tools seamlessly into the existing curriculum structure.

The study included 91 undergraduate computer science students enrolled in the required first-year programming course. The experimental group, which was granted access to the GenAI-powered feedback tools (tutorB@t and tutorBot+), consisted of 37 students. The control group (54 students) continued to receive feedback via the standard method: static unit tests from the code evaluation platform and delayed, instructor-provided feedback.

3.2. AI Tool Development and Architecture

Both tools utilized the same underlying GenAI model, GPT-4o-mini [21], accessed via a university API key. This minimized the variable of LLM capability, focusing the comparison on the interface and integration model.

tutorB@t (Conversational Assistant)

This tool was designed as a separate web-based chatbot assistant, simulating an interactive tutor.

- Interaction: Students could paste their code or error messages and ask open-opened questions (e.g., "Why is my loop not stopping?").

- Prompt Engineering: The core prompt was engineered [39] to instruct the LLM to adopt the persona of a helpful but non-solution-giving programming tutor. It emphasized: 1) identifying the error, 2) explaining the concept behind the error, and 3) providing a structured hint or a guiding question instead of the corrected code [24]. This enforced a focus on metacognition.

tutorBot+ (Platform-Integrated Tool)

This tool was integrated directly into the university's custom programming evaluation platform, operating non-conversationally.

- Interaction: When a student submitted code, the platform first ran it through a Judge0 instance (a virtual code evaluator). If the code failed any test case, the platform automatically sent a structured prompt to the LLM.

- Prompt Grounding: The LLM's prompt was strictly "grounded" by providing the problem description, the student's code, the failed test case input/output, and the exact error message from the compiler (if available).

- Output: The LLM returned a concise, targeted feedback block that was displayed on the submission results page. This feedback was formatted as a single, actionable suggestion, prioritizing efficiency and speed over dialogue.

3.3. Research Procedure

1. Onboarding: All students in the experimental group received a mandatory session detailing the capabilities and appropriate ethical use of the AI tools. They were explicitly instructed that the tools were for learning assistance and debugging, not for problem-solving.

2. Intervention: For ten sequential weekly programming lab assignments, the experimental group had unrestricted access to both tutorB@t and tutorBot+ in addition to all traditional resources. The control group only had access to traditional resources. The assignments were standardized problem-based learning tasks [12] covering core introductory topics (e.g., loops, conditionals, functions).

3. Data Collection: Usage logs (timestamp, number of interactions) were collected for the experimental group. Post-intervention, all participants completed a final assessment. Both groups completed a perception questionnaire regarding the learning environment.

3.4. Data Collection Instruments

Programming Performance

Objective performance was measured by two metrics:

1. Test Case Success Rate: The average percentage of automated test cases passed across all ten assignments.

2. Assignment Passing Rate: The percentage of assignments that met the required minimum passing threshold (e.g., 90% test case success).

Usability and Perception

The primary instrument for measuring user perception was the System Usability Scale (SUS) [6]. The SUS is a validated, 10-item scale providing a single score from 0 to 100, which offers a quick, reliable measure of a tool's perceived usability. This was administered to the experimental group for both tutorB@t and tutorBot+ separately.

Intent to Reuse was assessed using two direct survey questions (e.g., "How likely are you to use this tool for future programming courses?").

3.5. Data Analysis

Quantitative data (performance metrics, SUS scores) were analyzed using standard statistical methods in the social sciences. Specifically, an independent samples t-test was used to compare the Test Case Success Rate and Assignment Passing Rate between the experimental and control groups. Descriptive statistics were used to report and interpret the SUS scores against industry benchmarks. Qualitative feedback was analyzed thematically to provide context for the quantitative results.

IV. Results

4.1. Student Programming Performance and Course Outcomes

A comparison of the experimental group (AI-assisted) and the control group (traditional feedback) on the primary performance metrics yielded no statistically significant difference over the duration of the study.

Metric	Control Group (N=54) Mean (SD)	Experimental Group (N=37) Mean (SD)	p-value (t-test)
Average Test Case Success Rate (%)	78.4 (15.1)	79.9 (14.5)	
Assignment Passing Rate (%)	68.5%	73.0%	

The experimental group demonstrated a marginally higher mean test case success rate and assignment passing rate, but this difference was not statistically significant. The study found no evidence that the integration of automated GenAI feedback, in its current form, is directly associated with a significant difference in objective programming performance or course outcomes.

4.2. Tool Usability and Perceptions

In stark contrast to the performance metrics, student perceptions of the tools were highly positive, confirming the first research question regarding usability and perceived utility.

The System Usability Scale (SUS) results were as follows:

AI Tool	SUS Score (0-100)	Interpretation (Based on Industry Benchmarks)
tutorB@t (Conversational Chatbot)	**70.6**	Acceptable / Good
tutorBot+ (Platform-Integrated Feedback)	**65.2**	Acceptable

A score of 68 is generally considered the average benchmark for usability. Both tools achieved scores above or near this threshold, indicating that students found them relatively easy to use, intuitive, and non-frustrating. The higher score for the conversational chatbot (tutorB@t) is associated with a perception that the interactive dialogue format may be slightly more user-friendly and engaging.

4.3. Intent to Reuse

The high usability translated directly into a strong intent to reuse both tools, suggesting students see value in having GenAI as a learning resource, even if the measured performance gains were minimal.

- 81% of students in the experimental group reported they would be likely or very likely to use the conversational chatbot (tutorB@t) again in future programming courses.
- 79% of students reported they would be likely or very likely to reuse the platform-integrated feedback (tutorBot+).

4.4. Qualitative Insights on AI Feedback Value

The thematic analysis of open-ended student comments provided critical context for the high perception scores. The value proposition of the GenAI tools consistently revolved around two themes: immediacy and accessibility.

Students repeatedly highlighted the advantage of receiving feedback immediately upon submission, contrasting it favorably with the several-day delay often associated with human grading. This timely feedback allowed them to fix errors and progress with their learning while the problem was still cognitively salient, fulfilling a core pedagogical requirement [13], [27]. Furthermore, the 24/7 accessibility was cited as crucial for students who often worked outside standard office hours. The AI served as a reliable, always-available helper, particularly for debugging small, frustrating issues [1].

V. Discussion

5.1. Interpretation of Core Findings

The central, and perhaps most critical, finding of this study is the decoupling of positive student perception from measurable academic performance [32], [38]. Students found both the conversational (tutorB@t) and the integrated (tutorBot+) tools usable and valuable, achieving acceptable SUS scores of 70.6 and 65.2, respectively. Yet, the presence of these tools was not associated with a significant advantage in objective passing rates or test success compared to the control group.

This suggests that GenAI, in its current implementation, successfully addresses the usability and motivational aspects (ARCS model's Confidence/Satisfaction [15]) of programming education by providing fast, accessible assistance. However, it may not yet fully address the deeper pedagogical challenge of forcing students to engage in productive struggle and genuinely learn from their mistakes. The AI's value appears to be primarily in its role as a supportive aid—a time-saver and frustration-reducer—rather than a guaranteed driver of learning outcomes [31].

5.2. Analyzing the Performance Plateau: The Challenge of Productive Scaffolding and Metacognitive Load

The most critical finding—the high user satisfaction coupled with non-significant academic performance gains—demands a deeper, theoretically grounded explanation beyond general factors like non-homogeneity or external pressures. We propose that the plateau in performance is directly related to the difficulty of designing GenAI feedback that optimizes the student's productive struggle and minimizes extraneous cognitive load [33].

As discussed in Section 2.1.3, deep learning from an error requires a student to engage in metacognitive regulation: monitoring the impasse, evaluating the failure, and revising the plan [29], [34]. Our GenAI tools, while highly capable, struggled to execute this pedagogical necessity perfectly, potentially leading to three metacognitive failure modes:

5.2.1. Failure Mode 1: Bypassing the Impasse via Over-Scaffolding

The core principle of managing the productive struggle is providing minimal intervention [33]. While our tools were explicitly engineered to avoid giving the final answer, the LLM's natural language generation capabilities sometimes produced feedback that was too comprehensive. For example, when a student submitted code with a logic error in a loop condition, the AI might have correctly identified the required correction in terms of concept (e.g., "You need to iterate up to and including the length, not just up to the length") but failed to force the student to find the exact syntactic correction (vs.).

In the conversational tutorB@t model, this was compounded by the student's ability to "pester" the AI with follow-up questions until the hint became functionally equivalent to the solution. The student's persistent questioning, while indicative of engagement (satisfying the ARCS model [15]), effectively allowed them to bypass the necessary intellectual challenge—the impasse—potentially leading to a fast solution without the corresponding metacognitive effort required for durable learning [34]. The high intent to reuse (81%) for this conversational tool is associated with students recognizing its potential for this efficient, albeit pedagogically suboptimal, shortcut.

5.2.2. Failure Mode 2: Increased Extraneous Cognitive Load via Verbose Explanations

In contrast to the over-scaffolding issue, the non-conversational tutorBot+ tool, designed for maximum specificity by grounding its output in Judge0 test results, sometimes generated verbose and technically dense explanations. While the LLM's ability to synthesize a detailed diagnosis from raw compiler output is technically impressive [1], the resultant paragraph could overwhelm a novice programmer, increasing their extraneous cognitive load [33].

Instead of focusing their cognitive resources on the germane load (the effort required for understanding the underlying concept), students had to expend significant resources simply trying to decode the AI's complex, natural language explanation. This phenomenon may explain the slightly lower usability score for tutorBot+ (65.2) compared to the chatbot (70.6). The feedback was theoretically more correct and grounded, but its delivery hindered the student's ability to move efficiently through the Evaluation and Revision phases of metacognitive regulation. The sheer volume of information, even when technically accurate, is sometimes considered counterproductive to the goal of efficient learning [24].

5.2.3. Failure Mode 3: Misalignment of Feedback Type and Metacognitive Phase

The core design difference between the two tools may have led to an inherent misalignment with metacognitive needs:

- tutorB@t (Conversational): Optimized for conceptual queries and Planning phases (Phase 1). Students could ask about strategy before writing code. However, its lack of direct access to run-time errors made it less effective for the Evaluation phase (Phase 3). Students had to manually paste errors, losing context.

- tutorBot+ (Integrated): Optimized for the Evaluation and Revision phases (Phases 3 and 4) by providing immediate, grounded feedback on failed execution. However, it offered no opportunity for dialogue, making it less useful for students stuck in the Planning phase with a blank screen.

The lack of performance gains suggests that students often used the tools at the wrong time or in a way that did not meet their immediate learning needs. For example, using the highly conceptual tutorB@t when the actual problem was a simple syntax error best diagnosed by tutorBot+'s test output, or vice versa. This misalignment may have resulted in wasted effort and frustration, neutralizing any potential performance benefit derived from the timeliness of the feedback [13].

5.3. Comparison with Existing Literature

Our findings align with other GenAI studies that report high positive student sentiment [4], [10], [39]. The high intent to reuse (over 79% for both tools) confirms the acceptance of these technologies in the classroom.

The comparison between the two tools is particularly illuminating. The higher usability score for the conversational tutorB@t (70.6) suggests that students inherently prefer the dialogic, flexible interface for their debugging needs. However, the slightly lower score for the platform-integrated tutorBot+ (65.2), which was designed to be highly specific and less conversational, suggests a trade-off: Specificity is often achieved at the expense of user experience. Future design work must reconcile the need for the targeted, grounded feedback of the platform tool with the perceived ease and friendliness of the conversational one [33].

5.4. Implications for Practice and Theory

The findings highlight that simply providing access to

GenAI is a necessary, but insufficient, condition for improving learning outcomes. The integration must be pedagogically informed, particularly in how it manages the metacognitive process.

5.4.1. Practical Implications: Designing for Metacognitive Engagement

Practitioners utilizing GenAI for programming feedback must shift the focus from mere accuracy (which LLMs generally handle well [9]) to pedagogical efficacy—the ability to promote genuine learning.

1. Iterative and Non-Verbal Scaffolding: Future tool design, especially for integrated systems like tutorBot+, should move toward multi-stage, iterative hints rather than a single large explanation. This could involve an initial non-verbal hint (e.g., highlighting the line of code [41]), followed by a conceptual hint, and only then a detailed explanation. This design forces the student to re-engage with the problem before escalating the level of help, thereby enforcing the productive struggle.

2. Contextual Feedback for Planning: For conversational tools like tutorB@t, the prompt engineering must be refined to be highly demanding of the student's existing plan before offering guidance. For instance, the AI should refuse to proceed until the student articulates their intended algorithm or current failure hypothesis, directly targeting the Monitoring phase [29].

3. Mandatory Reflection: Implementing a mandatory reflective step before allowing students to copy code from the platform, as suggested by literature on self-regulated learning. For example, a pop-up requiring the student to type "I have identified the error as X, and my plan is to try Y" before revealing the AI's hint. This ensures the student engages in explicit metacognitive revision [26].

5.4.2. Theoretical Implications: Moving Beyond Performance Metrics

The high perception scores (SUS 70.6 and 65.2) and the high intent to reuse (over 79%) suggest that the GenAI tools are highly effective at addressing the affective domain of learning, even if their impact on the cognitive domain (performance) is minimal in a short-term study.

This requires a theoretical pivot in the evaluation of educational AI. We must formally incorporate motivational models, such as the ARCS framework [15], and self-efficacy instruments [38] into future methodologies. If GenAI reduces frustration, increases persistence, and boosts a student's confidence in their ability to debug—all factors associated with timely and personalized feedback—then it is profoundly valuable, even without an immediate lift in test scores. This value is captured not by traditional performance metrics, but by measuring engagement, persistence, and self-regulation over a longer duration [31]. Future studies, as detailed in Section 5.6, must therefore prioritize the collection of these affective and motivational data points.

5.5. Study Limitations

Acknowledging the limitations is crucial for interpreting these results. Key limitations include:

- The quasi-experimental design hinders definitive causal inference due to potential pre-existing differences between the groups.
- The duration of the study (one semester) may have been too short for students to fully adapt their learning behaviors to the new resources.
- The reliance on a single LLM (GPT-4o-mini) limits generalizability across the rapidly evolving GenAI landscape.

5.6. Future Work

To capitalize on the high user acceptance and resolve the performance ambiguity, future research will concentrate on three strategic pillars, focusing heavily on metacognitive scaffolding and affective impact.

5.6.1. Integrating Affective and Motivational Measures

To fully capture the value demonstrated by the high perception and reuse rates, future iterations of this study will incorporate validated psychometric instruments alongside performance tracking [38].

Specifically, we will integrate the Instructional Materials Motivation Survey (IMMS) [8] to systematically assess the motivational impact of the AI tools. The IMMS is designed around the four components of the ARCS model [15]:

- Attention: Does the novelty and interactivity of the AI hold the student's interest?
- Relevance: Does the AI connect the problem to the student's long-term goals?
- Confidence: Does the AI help the student feel successful and capable?
- Satisfaction: Does the student feel rewarded and pleased with the learning outcome?

This quantitative affective data will be triangulated with semi-structured interviews and think-aloud protocols during debugging sessions. The goal is to move beyond the subjective "I liked it" to the measurable "It increased my persistence when facing an impasse," providing the

necessary evidence to support the theoretical value of GenAI as an engagement and confidence builder [38].

5.6.2. Refined Scaffolding and Accessibility Design

Future development will focus on integrating the best features of both tools while optimizing for the productive struggle:

1. The Hybrid Interface: We will develop a single, hybrid interface that presents the grounded, Judge0-based feedback of tutorBot+ as a default, non-verbal notification. The student can then click an "Explain/Chat" button to initiate the tutorB@t conversational dialogue, ensuring that conceptual support is always secondary to the execution-grounded diagnosis.

2. Explicit Metacognitive Prompts: Prompt engineering for GPT-4o-mini will be revised to include explicit instructions for the AI to address the student's Planning and Monitoring phases directly. For instance, the prompt will instruct the AI to phrase its feedback as a question that forces the student to analyze their assumption rather than their code line (e.g., "Review your initial plan. Are you sure you accounted for the edge case where the input list is empty?").

3. W3C ATAG Compliance: Accessibility is a pedagogical imperative [17]. Future versions will undergo a thorough accessibility audit, focusing specifically on W3C ATAG compliance, ensuring that keyboard navigation, screen reader compatibility, and clear text contrast are integrated into the final design, guaranteeing equitable access to this powerful feedback mechanism.

5.6.3. Longitudinal Study and Scaffolding Adjustment

A long-term, multi-cohort study is planned to overcome the limitation of the single-semester intervention. This study will track student performance across multiple courses and systematically compare three groups:

1. Control: Traditional static feedback.

2. Minimal Scaffolding: AI provides only error location and an abstract hint.

3. Optimal Scaffolding: AI dynamically adjusts hint specificity based on the student's submission history and time spent at the impasse.

This longitudinal approach will allow researchers to observe whether the cumulative effect of timely, high-quality feedback eventually is associated with the development of superior long-term metacognitive skills and sustained performance gains in subsequent, unassisted assignments.

VI. Conclusion

The integration of Generative AI, specifically GPT-4o-mini, into undergraduate programming education through a conversational chatbot (tutorB@t) and a platform-embedded tool (tutorBot+) is associated with a significant step forward in addressing the challenge of timely and scalable feedback. Students overwhelmingly found both tools useful and easy to use, reflected in acceptable SUS scores and high intent to reuse. This confirms the critical value of AI as an accessible, always-available support system for novice programmers.

However, the lack of a statistically significant difference in objective performance metrics underscores a key finding: High utility does not guarantee high academic impact. This plateau is likely attributable to complex pedagogical and confounding factors, including student behaviors driven by academic pressure and the intrinsic difficulty of interpreting and applying automated feedback effectively, particularly concerning the necessary metacognitive regulation.

Ultimately, the power of GenAI lies not in replacing the human instruction model, but in providing a reliable, personalized lifeline to students whenever they are stuck. Future efforts in this domain must focus intently on pedagogical design—ensuring the AI promotes productive struggle, measuring its impact on motivation and self-efficacy, and iteratively refining the interface to blend specificity with usability. It is through this balanced approach that we can transition GenAI from a helpful novelty to a true enhancer of learning outcomes.

References

[1] Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-generation for programming exercises with GPT4. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1. ITiCSE 2024. ACM (pp. 31–37). https://doi.org/10.1145/3649217.3653594

[2] Bailey, R., & Garner, M. (2010). Is the retroalimentación in higher education assessment worth the paper it is written on? Teachers' reflections on their practices. Teaching in Higher Education, 15(2), 187–198. https://doi.org/10.1080/13562511003620019

[3] Bangs, J. (2007). Teaching perfect and imperfect competition with context-rich problems. SSRN Electronic Journal, 92(3), 463. https://doi.org/10.2139/ssrn.1024000

[4] Bassner, P., Frankford, E., & Krusche, S. (2024). Iris: an AI-driven virtual tutor for computer science education. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1. Milan, Italy: Association for Computing Machinery (pp. 394–

400). https://doi.org/10.1145/3649217.3653543

[5] Billis, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., & Saunders, W. (2023). Language models can explain neurons in language models. Available at https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

[6] Brooke, J. (1986). SUS—a quick and dirty usability scale. In: Usability Evaluation in Industry. United Kingdom: Taylor & Francis (pp. 189–194).

[7] Bull, C., & Kharrufa, A. (2024). Generative artificial intelligence assistants in software development education: a vision for integrating generative artificial intelligence into educational practice, not instinctively defending against it. IEEE Sof1tware, 41(2), 52–59). https://doi.org/10.1109/ms.2023.3300574

[8] Cardoso-Júnior, A., & Faria, R. M. D. D. (2021). Psychometric assessment of the Instructional Materials Motivation Survey (IMMS) instrument in a remote learning environment. Revista Brasileira de Educação Médica, 45(4), e197. https://doi.org/10.1590/1981-5271v45.4-20210066.ing

[9] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei2

[10] Kumar Tiwari, S. (2023). Integration of AI and machine learning with automation testing in digital transformation. International Journal of Applied Engineering & Technology, 5(S1), 95–103. Roman Science Publications.

[11] Kesarpu, S., & Hari Prasad Dasari. (2025). Kafka Event Sourcing for Real-Time Risk Analysis. International Journal of Computational and Experimental Science and Engineering, 11(3). https://doi.org/10.22399/ijcesen.3715

[12] Singh, V. (2024). The impact of artificial intelligence on compliance and regulatory reporting. J. Electrical Systems, 20(11s), 4322–4328. https://doi.org/10.52783/jes.8484

[13] Real-Time Financial Data Processing Using Apache Spark and Kafka. (2025). International Journal of Data Science and Machine Learning, 5(01), 137-169. https://doi.org/10.55640/ijdsml-05-01-16