STEM Education, 6(5): 927–951
DOI: 10.3934/steme.2026037
Received: 21 February 2026
Revised: 13 July 2026
Accepted: 22 July 2026
Published: 03 September 2026

https://www.aimspress.com/journal/steme

*Research article*

# A pedagogical model of generative AI tutoring in informatics education and its effects on problem-solving quality and academic independence

**Raushan Sambetova[1,*] and Lyazzat Zhaidakbayeva[2]**

[1] International University of Tourism and Hospitality, Turkistan, 161200, Kazakhstan
[2] Mukhtar Auezov South Kazakhstan University, Shymkent,160000, Kazakhstan

*** Correspondence:** Email: raushan.sambetova@iuth.edu.kz; Tel: +7 701 084 7878.

Academic Editor: Feng-Kuang Chiang

**Abstract:** Generative AI systems are increasingly adopted as on-demand tutors in informatics and introductory programming, yet evidence remains mixed regarding whether performance gains come at the cost of reduced academic independence and heightened over-reliance. Resolving this issue requires moving beyond open-ended AI assistance and toward tutoring systems where pedagogical safeguards are built into the design from the beginning, rather than added later as an afterthought. In this study, we proposed and evaluated an Independence-by-Design pedagogical model for generative AI tutoring that targets improvements in problem-solving quality while preserving learners' academic independence through scaffolded, guardrail-based assistance. The intervention operationalizes tutoring policies such as staged hints, explain-first prompting, and answer-suppression aligned with a programming problem-solving rubric, and it was evaluated in a controlled classroom study comparing a guardrail-based tutor condition with a control condition that used standard instructional resources without AI assistance. Primary outcomes included problem-solving quality, assessed via rubric-based scoring of correctness, reasoning, and code quality, and academic independence, quantified using an Academic Independence Index that triangulates self-regulated learning measures, help-seeking behaviors, and interaction trace features (e.g., hint depth, revision cycles, and direct-answer requests). Effects were estimated using regression models with prior achievement covariates and reported with standardized effect sizes. Results indicated that the guardrail-based generative AI tutor improves problem-solving quality relative to baseline support while maintaining or increasing academic independence indicators, and interaction analyses suggest that moderate hint usage and explain-first behavior are associated with the strongest outcomes. These findings support the responsible deployment of generative AI tutors in informatics education by demonstrating that learning gains can be achieved without amplifying dependency when pedagogical guardrails are embedded in the tutoring design. In practical terms, we give educators a transferable design approach for using generative AI in programming courses. This approach includes staged

hints, requiring students to attempt a solution before receiving help, avoiding direct code answers, and prompting students to explain their thinking first. Together, these features help AI support students' reasoning rather than replace it.

**Keywords:** generative artificial intelligence, large language models, intelligent tutoring systems, computer science education, problem-solving performance, academic independence, self-regulated learning

## 1. Introduction

### 1.1. Problem framing

Generative AI and large language models (LLMs), such as ChatGPT in particular, has gained rapid traction as a means of delivering personalized tutoring in informatics education [1]. Within undergraduate programming courses, AI-powered tutors hold the promise of real-time feedback and guidance, an especially valuable capability when class sizes are large and instructor availability is limited [2]. Programming now occupies a central place in university curricula, yet many students find it difficult to develop solid coding skills without one-on-one support. LLM-based tutors present a compelling way forward: They can interact with learners through natural language, assist with debugging, and clarify concepts whenever the need arises [2]. These developments resonate strongly with Kazakhstan's broader digital education agenda. The country's national strategy for 2025–2029 places particular emphasis on embedding AI in the educational process, with careful attention to ethical use and academic integrity. Kazakhstan has approved a regulatory framework designed to bring AI-driven digital tools into schools and post-secondary institutions while ensuring the systematic, responsible, and safe use of AI and safeguarding academic honesty [3]. Against this backdrop, investigating how generative AI tutors can support programming education is timely and highly relevant, with the potential to strengthen problem-solving instruction in ways that align with Kazakhstan's national digital transformation objectives.

### 1.2. Research gap

Early research on LLM-based tutoring has documented encouraging learning outcomes but has also raised legitimate concerns about over-reliance and weakened academic independence. On the positive side, LLM tutors appear capable of boosting student performance and engagement [4]. However, LLMs are remarkably good at producing correct solutions, and this very competence has fueled growing concern that students may become over-dependent on AI assistance rather than building their own capabilities [2]. In programming courses, an AI that readily generates working code or offers overly detailed hints risks encouraging students to copy solutions instead of genuinely engaging with underlying concepts [2]. Researchers have observed that students using AI without proper guidance tend to reflect very little on the material they encounter, accepting AI-generated answers at face value with minimal critical examination, which can gradually erode independent problem-solving skills [5]. One study went so far as to show that excessive reliance on an LLM-based coding tutor diminishes students' independent problem-solving performance, even though moderate use is beneficial [2]. Surveys of university students revealed that a subset (over 20%) admit to using tools like ChatGPT as "assignment delegators," relying on AI to draft solutions or code [6].

Despite these warning signs, many studies and tools have not incorporated explicit safeguards against over-assistance. A systematic review of intelligent tutoring systems found that most platforms either deliver fixed hints or fully worked-out solutions; approaches that undercut learning by revealing answers too directly [7]. In the initial wave of chatbot and code assistant deployments, there has been a noticeable absence of pedagogical constraints or scaffolds to ensure that AI tutors "teach, not tell." This gap means that the ability of LLM tutors to genuinely foster learning without simultaneously fostering dependency remains largely unexplored. Moreover, the outcomes examined in prior work have tended to be narrow in scope, concentrating on performance gains (such as test scores or assignment grades) while paying much less attention to how AI assistance shapes learner autonomy and self-regulation. Conventional metrics typically link AI usage to final task outcomes (correct versus incorrect) [5], which may fail to capture the more subtle reliance behaviors that emerge during problem solving. There is a need for new constructs and measures that directly gauge academic independence, along with approaches that can harness the benefits of LLM tutoring while mitigating over-reliance.

### 1.3. Contributions

To address these gaps, we propose and investigate a novel approach to guardrail-based generative AI tutoring in an undergraduate programming setting. The contributions of this work are threefold. First, we introduce an Independence-by-Design Tutor Model: A pedagogical framework that embeds independence-by-design into generative AI tutoring. Rather than functioning as a conventional AI helper that hands over answers, our LLM tutor is scaffolded with guardrails; for example, withholding direct code solutions and relying on Socratic questioning and incremental hints to actively encourage students' own problem-solving efforts. Second, we develop an Academic Independence Index: A new metric to quantify learner independence in the presence of an AI tutor. This index integrates system trace data (e.g., frequency and timing of hint requests, extent of AI-generated guidance used) with self-regulated learning (SRL) constructs such as help-seeking tendencies and reflective practice [8]. By triangulating students' interaction logs with self-reported SRL behaviors, this index offers a measurable outcome for academic independence that moves beyond traditional performance metrics. Third, we provide an Empirical Evaluation of Guardrails by implementing the proposed AI tutor and evaluating it in a real undergraduate informatics course in Kazakhstan. Through a controlled study, we examine how guardrail-based tutoring affects programming task performance and Academic Independence Index scores, showing that the guardrailed LLM tutor improves problem-solving success while preserving students' independence [9]. In brief, the study is a randomized controlled comparison with 84 first-year CS1 students at a public university in Kazakhstan, assigned to a guardrail-based AI tutor condition or a no-AI control (42 per group). During one supervised 90-minute lab session, all students solve the same two Python tasks. Primary outcomes are rubric-scored solution quality and the Academic Independence Index; the tutor logs all hint requests, code submissions, and dialogue turns. Sections 4 and 5 give the full design.

### 1.4. Paper organization

The remainder of this paper is structured as follows. In Section 2, we review related work on AI tutoring systems and student independence. In Section 3, we describe the design of the Independence-by-Design LLM tutor and the Academic Independence Index. In Section 4, we detail

the system intervention design. In Section 5, we present the methodology. In Section 6, we report the results. In Section 7, we discuss implications, and in Section 8, we address threats to validity. In Section 9, we conclude with future directions.

## 2. Related work and positioning

### 2.1. Review methodology

To situate this study within the research landscape, we conducted a focused narrative review of work connecting three major areas: Intelligent tutoring systems and scaffolding in computer science education, generative AI and large language models in programming instruction, and self-regulated learning, help-seeking, and academic independence in AI-supported learning contexts. For source identification, we searched Scopus, Web of Science, Google Scholar, and the ACM Digital Library using combinations of terms such as "generative AI," "large language model," "ChatGPT," "intelligent tutoring system," "programming education," "computer science education," "scaffolding," "self-regulated learning," "help-seeking," "over-reliance," and "academic independence." We focused mainly on peer-reviewed studies published between 2017 and 2026, with particular attention to work from 2023 to 2026 because of the rapid development of LLM-based tutoring tools. Studies were included if they met at least one of three criteria: They entailed AI- or ITS-based tutoring in higher education or upper-secondary computer science and programming contexts; they reported empirical findings on learning outcomes, help-seeking, or reliance behaviors; or they offered theoretical concepts, such as self-regulated learning, autonomy support, or scaffolding, that were directly relevant to AI-mediated learning. We excluded purely technical papers on LLM architecture without educational evaluation, opinion pieces without clear empirical or conceptual contribution, and studies outside CS or STEM unless they introduced concepts used in this study. Each selected source was analyzed across four dimensions: Its main topic, such as ITS and scaffolding, LLM benefits, LLM risks and over-reliance, or SRL and independence; its study design, whether empirical, conceptual, or meta-analytic; its key findings; and its implications for guardrail-based tutoring. The findings were then synthesized thematically to identify the main gap addressed by us: The need for empirically validated AI tutors with built-in pedagogical guardrails that measure student performance and academic independence. Subsections 2.2 to 2.5 present this synthesis, while Table 1 summarizes the major sources by theme.

**Table 1.** Summary of key literature by theme

| Theme | Representative sources | Core finding / contribution | Implication for this study |
|---|---|---|---|
| ITS & scaffolding in CS education | Perikos et al. (2017) [10]; Chen et al. (2026) [11] | Step-by-step hints, adaptive feedback, and worked examples improve learning within the ZPD; meta-analytic effect $g \approx 0.71$ on computational thinking | Motivates staged-hint and adaptive-feedback design of the Independence-by-Design tutor |
| LLM/GenAI benefits in programming | Yan, Y.M. et al. (2025) [4]; Yan, L. et al. (2025) [12]; Lai & Lin (2025) [2] | LLM assistants reduce cognitive load, improve computational thinking, and provide on-demand peer-tutor support | Justifies LLM as the underlying tutoring engine |
| LLM/GenAI risks & over-reliance | Bastani et al. (2025) [9]; Lai & Lin (2025) [2]; Stojanov et al. (2024) [6]; Hou et al. (2025) [5] | Unconstrained AI use correlates with lower grades, copy-paste behavior, and degraded independent performance; >20% of students delegate full solutions to AI | Establishes the need for explicit guardrails and answer-suppression policies |
| SRL, help-seeking & | Anders & Speltz (2025) [1]; Domino & Shaer (2025) | SRL strategies, strategic help-seeking, and autonomy support drive durable learning | Grounds the Academic Independence Index and |

| | | | |
|---|---|---|---|
| autonomy | [8]; Hou et al. (2026) [13]; Johansen et al. (2023) [14] | gains; reliance can produce an "illusion of competence" | explain-first prompting |
| Performance vs. learning dissociation | Bassner et al. (2025) [15]; Aruğaslan (2024) [16] | AI assistance can boost immediate task performance without producing equivalent learning gains; integrity risks rise with weakened self-regulation | Motivates measuring *both*problem-solving quality and academic independence as separate constructs |

### 2.2. Intelligent tutoring systems and CS education scaffolding

Intelligent Tutoring Systems (ITS) have a long-standing presence in computer science education, where they serve as tools for delivering individualized guidance. A hallmark of effective ITS is instructional scaffolding—step-by-step hints, adaptive feedback, and worked examples that help students advance within their zone of proximal development (ZPD) [7,10]. A growing body of research confirms that well-designed scaffolds meaningfully improve learning outcomes. Providing immediate feedback, guided practice, and adaptive hints has demonstrable positive effects on student mastery [10,12]. A meta-analysis spanning 30 studies in programming education reported that scaffolding interventions yielded a large positive effect (Hedges' g ≈ 0.71) on students' computational thinking skills [11]. Within Kazakhstan's CS education landscape, the emphasis on scaffolded digital tutoring aligns well with national objectives. Kazakhstan's recently adopted AI-in-education framework makes clear that AI should ensure systematic, responsible, and safe use, and that it should enhance teachers' roles rather than replace them, underscoring a commitment to guided support rather than answer-giving automation. A related study using generative AI with disadvantaged secondary-school students similarly found that structured, well-supported AI use improved academic outcomes, consistent with the broader pattern that guided AI use outperforms unguided use [17].

### 2.3. LLM/GenAI in programming education: Benefits and Risks

Within the family of intelligent tutoring systems introduced in Section 2.1, we define an AI tutor as an AI system that delivers personalized instructional support, including analyzing student inputs, estimating their understanding, and providing targeted hints or explanations in real time [10]. LLM-based assistants simulate human-like dialogue, enabling students to receive instant explanations, code suggestions, and debugging support in a personalized way. Moreover, incorporating LLM support into programming assignments has been shown to reduce students' cognitive load and improve computational thinking skills [4]. In one quasi-experimental study involving middle-school students, teams working alongside an AI "pair programmer" outperformed peers in problem-solving tasks and reported lower mental effort [4]. Integrative experiments in other educational domains have shown that proactive AI tutors using scaffolded prompts significantly enhance learning outcomes compared to unguided AI use or traditional methods, with benefits that carry over to post-tests [12]. These results suggest that, when deployed thoughtfully, LLMs can function as always-available peer tutors. This promise holds special relevance for Kazakhstan and similar contexts, where AI-based tutoring could help democratize access to quality programming instruction, complementing the nation's push for digital learning tools in higher education.

Alongside these benefits, researchers have sounded important warnings about over-reliance and dependency. Empirical evidence shows that heavy reliance on GenAI can hurt learning: In one controlled study, a negative correlation emerged between frequent LLM use for coding tasks and

course performance, with heavy users receiving significantly lower final project grades [2]. Students who used the AI more sparingly, turning to it only for hints after first making their own attempt, did not experience as steep a performance drop [2]. There is also the hallucination problem: LLMs sometimes produce incorrect solutions while projecting confidence [2]. When an AI provides complete solutions, students may simply copy answers, which surveys confirm occurs among over 20% of students [6]. Effective integration therefore demands a balanced approach that maximizes support while maintaining academic independence and critical thinking [1].

### 2.4. Academic independence constructs

Academic independence is closely bound up with self-regulated learning (SRL), strategic help-seeking, and self-efficacy. SRL refers to students' active management of their learning process; setting goals, choosing strategies, monitoring progress, and reflecting on outcomes. Effective learners recognize when to seek help but only after investing genuine effort [1]. Self-efficacy, a learner's confidence in their ability to succeed, is associated with persistence and willingness to work through problems independently. The introduction of AI tutors can challenge and reshape these constructs. On one hand, an LLM can act as a safety net that bolsters self-efficacy; students may feel more willing to tackle difficult tasks knowing they have AI support. On the other hand, ease of obtaining answers can short-circuit the SRL cycle. Evidence suggests that novices with lower self-efficacy tend to invoke AI help earlier and more frequently, while students who delay AI assistance typically display higher self-efficacy and achieve stronger outcomes [2]. Reliance on AI can create an illusion of competence: A student may feel successful with an AI-provided solution yet lack the ability to produce it independently [5]. Academic independence in an AI-mediated era therefore requires cultivating metacognitive awareness of when and how to use AI as a tool rather than a crutch, reinforcing healthy help-seeking behaviors where the AI functions as a guided mentor [1].

### 2.5. What is missing and how this work differs

Research illuminates important pieces of the puzzle, but a key gap persists at their intersection: There is a shortage of empirical studies on pedagogically guardrailed AI tutors that deliberately balance assistance with autonomy [9]. Most work evaluates either performance gains or anecdotal usage patterns, with very few examining measurable outcomes for student independence or long-term self-reliance. We directly address this gap by proposing and testing a guardrail-based generative AI tutor for programming. This approach is grounded in established scaffolding principles but with explicit guardrails designed to prevent over-assistance. By empirically evaluating programming performance and academic independence, we offer a fresh perspective on LLM integration in education, positioned at the nexus of technology and learner autonomy and aligned with Kazakhstan's vision for ethical AI in education.

## 3. Conceptual framework and pedagogical model

### 3.1. Definitions

Building on the related work introduced in Section 2 (in particular the AI-tutor and ITS concepts of [7,10]), we use the following working definitions specific to this study: Problem-solving quality refers to the caliber of the solution process; going beyond simply arriving at the correct answer to

capture how effectively and systematically the student approaches the problem. Academic independence describes a learner's capacity to pursue learning tasks autonomously, with minimal reliance on direct answers. A student with high academic independence can initiate strategies, persist through challenges, and handle problems on their own, turning to hints or verification only when genuinely needed.

### 3.2. Pedagogical model (Independence-by-Design)

We propose an Independence-by-Design pedagogical model in which every facet of the tutor's interaction promotes student autonomy. The tutor is built on GPT-4o accessed through the OpenAI API, controlled by a structured system prompt and a rule-based guardrail layer, and delivered through a web application that pairs the chat tutor with a code editor. The full system prompt and guardrail rules are given in Appendix A. The roles are clearly delineated: The student is the active problem-solver, while the AI acts as a guided mentor or coach. Rather than delivering answers, the AI tutor asks probing questions, offers hints, and scaffolds the learning process at each stage.

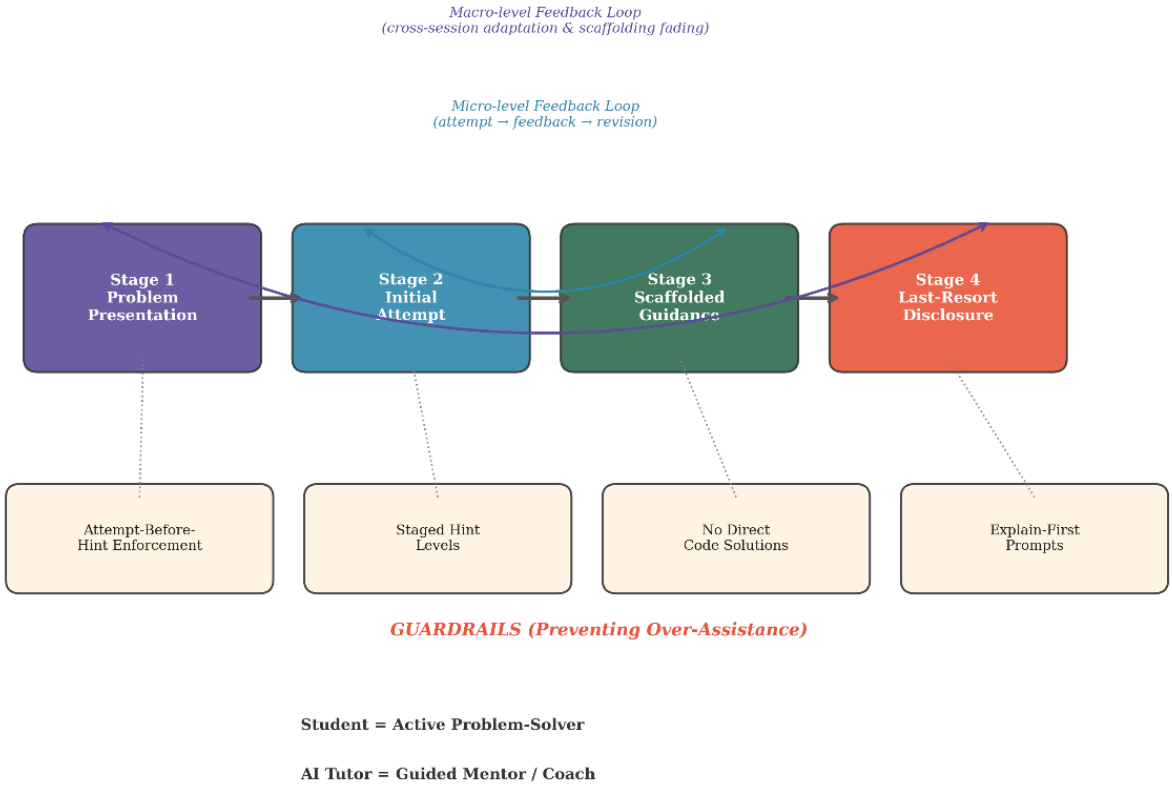


**Figure 1**. The Independence-by-Design tutoring cycle, showing the four stages (problem presentation, initial attempt, scaffolded guidance, last-resort disclosure) with micro- and macro-level feedback loops and guardrail checkpoints.

The tutoring process unfolds in stages (Figure 1) designed to mirror effective human tutoring. In problem presentation, the AI introduces a task calibrated to the student's skill level. During the initial attempt, the student is expected to try solving it independently, with the system potentially enforcing a short delay before any hint is offered. In the scaffolded guidance stage, the AI provides graduated hints if the student encounters difficulty; an initial broad hint (e.g., "Have you considered what the

question is really asking?"), progressing to more specific hints targeting specific errors. Crucially, the AI is governed by an answer-withholding policy: It will not simply hand over the answer at the first sign of struggle, giving only the minimum help necessary. The "bottom-out hint", disclosing the answer as a last resort, is delivered only after the student has exhausted guided attempts and is accompanied by an explanation to preserve learning value.

Feedback loops operate at micro and macro levels. At the micro-level, there is a continuous attempt–feedback–revision cycle: The student attempts a step, the AI evaluates it, and immediate feedback follows. If the attempt is incorrect, the tutor's feedback loop activates with either an error flag or a hint to steer the student toward the right thinking process. Consider a realistic interaction where a student is solving a problem and mistakenly applies incorrect logic. The AI might respond, "I see your answer. Remember how conditional statements work; does your condition actually capture the right case?", nudging the student to spot the error without outright correcting it. At the macro-level, the AI updates a student model across sessions, tracking which skills the student has mastered and where they needed many hints. Using this data, the system adapts future problem selections and hint levels, gradually fading support as competence grows. If a student demonstrates improvement, the AI might increase task difficulty or pose more open-ended questions. Conversely, if the student struggles consistently, the tutor revisits prerequisite concepts. These adaptive loops keep the tutoring within the learner's ZPD, the sweet spot where the student can succeed with some guidance but is challenged enough to genuinely learn.

### 3.3. Operationalization of academic independence

Academic independence is operationalized through four observable indicators. The first is help-seeking behavior: How often a student requests hints and how long they work before the first request, following self-regulated learning research on strategic help-seeking [8]. The second is the proportion of problems a student solves without receiving a full solution disclosure. The third is solution ownership, meaning the share of the final code the student wrote rather than adopted from tutor output, which builds on earlier work on reliance measurement [5,13]. The fourth is performance on assessment items completed without tutor support. The Academic Independence Index (AII) combines these indicators into a single score. Hint frequency and solution ownership receive the largest weights because they most directly capture the dependency behavior the guardrails are meant to prevent. The formula, weights, and scaling are given in Section 5.4.

### 3.4. Design principles and hypotheses

Three design principles guide development: (1) Independence-by-Design Scaffold—hints before answers, with the student as primary problem-solver; (2) Adaptive Feedback and Fading, continuously adjusting support levels, pushing learners to take on more as they improve; and (3) Reflective Learning—encouraging students to explain solutions and internalize what they learned, building self-regulation skills alongside content knowledge.

We hypothesize that: (H1) Students using the Independence-by-Design AI tutor will produce significantly higher-quality problem solutions than students without AI support. (H2) Guardrailed AI tutor use will not undermine academic independence: help-seeking and self-regulated learning behaviors will remain comparable to the control condition. (H3) Guided AI support will enable more efficient task completion, with lower cognitive load and maintained or enhanced self-efficacy.

## 4. System/Intervention design

### 4.1. Tutor modes

The AI tutor operates in three modes. In Socratic dialog mode, the tutor engages students with open-ended questions, simulating the Socratic method and promoting deeper learning and metacognitive skills [12]. In hint-only support mode, incremental hints are provided without revealing the full solution, beginning with gentle nudges and becoming progressively more specific, aligned with best practices showing that multiple levels of hints better support novice programmers [7]. In rubric-based feedback mode, structured evaluation against predefined criteria is delivered after a solution attempt, mapping feedback to explicit rubric categories (correctness, style, efficiency) to ensure alignment with learning objectives [12].

### 4.2. Guardrails to reduce over-reliance

Unconstrained generative AI can lead to over-reliance: in a field experiment in high-school mathematics, students given unrestricted access to a GPT-4-based tutor performed better on practice problems but scored significantly worse than controls once AI access was removed, consistent with a guardrail-free tool substituting for rather than building durable skill [9]. We implemented several layered guardrails (summarized in Table 2). Figure 2 previews a result reported fully in Section 6.3: the resulting distribution of the deepest hint level reached by students under the Staged Hint Levels guardrail, which delivers hints in tiers, requiring multiple requests for increased detail, with even the final hint stopping short of providing a full code. Attempt-Before-Hint Enforcement requires students to provide an attempt or explanation of their thinking before a hint is given, fostering productive struggle. The No Direct Code Solutions guardrail strictly withholds a complete solution code; the LLM is explicitly instructed: “Do not write any example code blocks. You must not write code for the student.” Automated post-processing removes any inadvertently generated solutions. The Explain-First Requirement embeds Socratic reflection moments, asking students to explain their reasoning before and after receiving help, keeping the learner actively involved in the tutoring interaction [1].

**Table 2.** Guardrail summary.

| Guardrail | Description | Pedagogical Rationale |
|---|---|---|
| Staged hint levels | Hints in increasing specificity across multiple requests | Encourages iterative problem-solving [7,10,12] |
| Attempt-before-hint | Student must attempt or explain before receiving a hint | Fosters productive struggle [1,2,8] |
| No code solutions | AI never provides complete code, only explanations | Prevents copy-paste dependency [6,9,15] |
| Explain-first prompts | Student explains thinking before and after help | Promotes reflection and metacognition [1,5,13] |

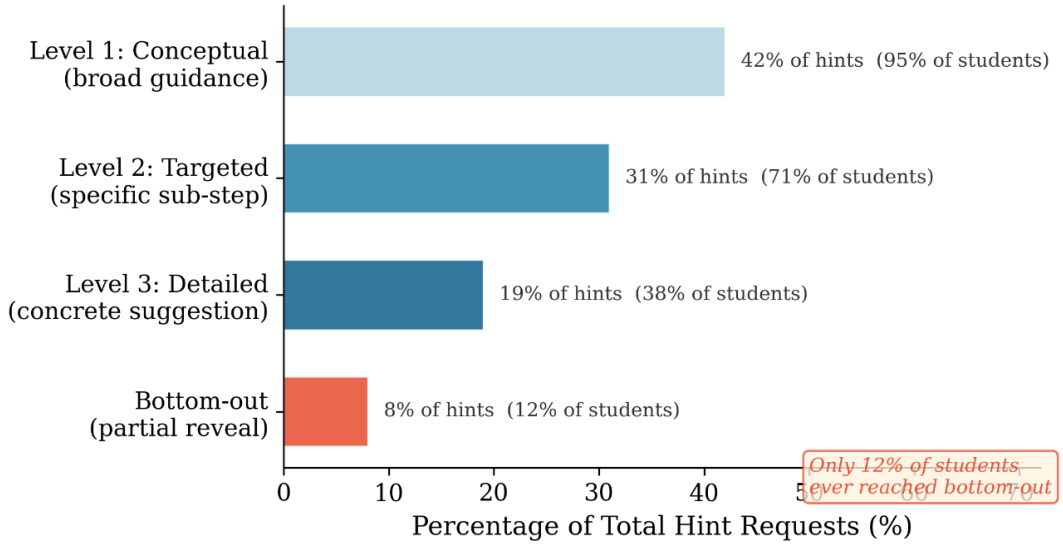


**Figure 2**. Deepest hint level reached in the AI Tutor condition. Bars show the percentage of hint-requesting students whose deepest request in the session is Level 1, Level 2, Level 3, or the bottom-out hint. Students who request no hints are not represented in the figure.

### 4.3. Logging and instrumentation

For evaluation purposes, the system logs detailed interaction data, enabling analysis of how students use the AI tutor and its impact on their workflow. The tutor records every relevant event with timestamps so that a problem-solving session can be reconstructed in full. Hint requests are logged along with their time of occurrence, the tutor mode active, and the content of the hint given. All code submissions and edits are logged, creating a versioned history of solution attempts. By comparing code snapshots before and after hints, we can observe whether hints lead to meaningful revisions. Time-on-task is captured by tracking session start and end times along with periods of inactivity. Additionally, the system logs dialogue interactions in Socratic mode, each question the tutor asks and the student's responses, giving a measure of the depth of AI–student interaction. All logged data is stored anonymously and keyed per student and task. This rich instrumentation enables quantitative analysis (counting hints per student, average time before first hint) and qualitative analysis (examining conversation content). Table 3 summarizes the logged variables.

**Table 3.** Logged interaction variables.

| Variable | Definition | Unit of analysis | Example record |
|---|---|---|---|
| hint_request | Timestamped hint request with active tutor mode, hint level, and hint text | Event | 14:22:07; hint-only mode; level 2; "Check the loop boundary condition" |
| code_submission | Full code snapshot at each run or save | Event | 14:25:41; task 2; snapshot #6 |
| code_edit | Diff between consecutive snapshots | Event pair | +7 / −2 lines between snapshots #5 and #6 |
| dialogue_turn | Tutor question or student reply in Socratic mode, verbatim | Event | Student: "I think the median needs sorting first" |
| session_time | Session start, end, and inactivity gaps over 60 s | Session | Start 14:00, end 15:21, idle 6 min |
| time_to_first_hint | Minutes from task start to first hint request | Student × task | 11.5 min |

### 4.4. Prompt policy

The unifying prompt design philosophy centers on "explain-don't-solve." The AI is instructed to provide conceptual explanations, hints, and questions that advance understanding while explicitly avoiding giving away the full solution. The prompt policy embeds metacognitive principles: The tutor responds with follow-up questions that make students reflect on their approach. This approach aligns with pedagogical best practices highlighting the importance of self-explanation and reflection for learning gains. Steering the LLM to explain and guide rather than solve is a form of built-in guardrail, consistent with academic integrity goals.

## 5. Methodology

### 5.1. Research questions and hypotheses

RQ1 asked whether a guardrail-equipped generative AI tutor would improve problem-solving performance compared to conventional learning without AI. We hypothesized that the AI tutor group would produce significantly higher-quality solutions (H1), based on prior evidence showing that AI assistance can improve short-term programming performance, even when its overall impact on learning is mixed [9,15]. RQ2 examined whether the AI tutor could assist without undermining academic independence. We hypothesized (H2) that guardrailed AI students would exhibit help-seeking and SRL behaviors comparable to control students, avoiding the "crutch" effect [9]. RQ3 addressed secondary outcomes. Moreover, we hypothesized (H3) that guided AI support would enable more efficient task completion with lower cognitive load and maintained or enhanced self-efficacy [15].

### 5.2. Study design

We employed a randomized controlled trial (RCT) with a between-subjects design. Participants were randomly assigned to either the AI Tutor condition (custom LLM-based tutor with pedagogical guardrails) or Control (no AI, conventional resources only). Randomization was stratified by prior programming experience to ensure balanced groups. Task grading was blinded, solutions were anonymized so raters did not know conditions. Control students had access only to standard course materials and a basic code editor, mimicking a closed-book setting. AI Tutor students used a web-based application integrating an LLM-driven chat tutor alongside a code editor. The tutor was explicitly prompted not to give away complete solutions, responding instead with guided hints and Socratic questions [9].

### 5.3. Participants, context, and tasks

Participants were undergraduate students (N = 84, aged 17–19, 47% female) enrolled in an introductory programming course (CS1) at a large public university in Kazakhstan. The course forms part of a first-year informatics curriculum and covers fundamental programming concepts in Python. All participants provided written informed consent, and the study protocol was approved by the [ethics committee name] (approval no. [XXX]); full details are given in the Ethics declaration. The experimental tasks consisted of two programming problems of moderate complexity: One involving implementing a function to compute statistical metrics (mean, median, variance) and another requiring a game simulation using loops and conditionals. Students had 90 minutes to complete both.

### 5.4. Measures

Problem-Solving Quality was assessed via a structured rubric evaluating functional correctness, code completeness, and code quality/style, each scored on a 0–5 scale (30 points maximum for two tasks). For reporting, total rubric scores were converted to percentages of the 30-point maximum, so all solution-quality values in Section 6 were on a 0–100 scale. Two experienced teaching assistants independently scored all solutions. Inter-rater reliability was excellent (r = 0.94, ICC = 0.93). Discrepancies were resolved through discussion.

The Academic Independence Index (AII) was derived from system logs and SRL behaviors [8]. The trace-based metrics included three measures: How often students asked for hints, how quickly they requested their first hint and whether they tried writing code before asking for help, and how much of the final solution was written by the student compared with AI-generated code. AI-generated code was identified by matching snippets from the chat logs.

The index was computed as

$$AII = w1(1 - H_{norm}) + w2 \cdot T_{norm} + w3 \cdot O + w4 \cdot S,$$

where $H_{norm}$ is hint frequency min–max normalized within the sample, $T_{norm}$ is time-to-first-hint normalized to the 90-minute session (students who never requested a hint were assigned 1.0), O is the proportion of final code written by the student (matched against tutor chat output), and S is the standardized self-report SRL score. The weights were set by the research team based on the construct definitions in Section 3.3.

For instance, a student who asked for only two minor hints and wrote the bulk of the solution might achieve AII = 0.85, whereas another who frequently queried the AI and pasted suggested code blocks might have AII = 0.50. The AII was validated against an independent self-report item ("I was able to solve the problems largely on my own, without relying on outside help"), with a positive correlation ($r \approx 0.6$), suggesting the index captures students' perceived autonomy to a reasonable degree. For control students, the trace-based help-seeking component was near ceiling (M = 0.98), since only 3 of 42 (7%) asked the proctor a content-related question. Their composite AII (M = 0.758, Table 4) sat lower because the index also included the self-report SRL component, which varied normally in both groups. We note that the trace components measured something different in the two conditions: Control students had no equivalent on demand help to decline, so their trace scores reflected the absence of available assistance rather than a choice to work independently. Between-group AII comparisons should be read with this in mind (see Sections 6.2 and 8.2).

Secondary Outcomes included time-on-task (system timestamps), cognitive load (post-task 7-point Likert scale; Cronbach's $\alpha = 0.82$) [15], and self-efficacy (pre- and post-session ratings on a 100-point scale).

### 5.5. Procedure

During a scheduled lab session, students were briefed, and AI condition students received a short introduction to the tutor interface. A pre-task survey collected demographics and baseline self-efficacy. Students then worked on the two problems, up to 90 minutes, and supervised to ensure compliance. Upon completion, a post-task survey collected cognitive load, self-efficacy, and experience data. Solutions were scored, and participants were debriefed. Control students were later given access to the AI tutor for practice to ensure educational equity.

### 5.6. Analysis plan

All 84 students completed tasks with no attrition. Rubric scores were approximately normally distributed with similar variances. For H1, we conducted an independent-samples t-test supplemented by OLS regression with covariates (prior experience, pre-task self-efficacy), non-parametric Mann–Whitney U, and effect size estimation. For H2, we examined the AI group's AII through effect sizes and confidence intervals, defining a priori that a drop exceeding 0.3 in AII would be educationally concerning. For H3, t-tests, repeated-measures ANOVA, and regression analyses were used. All tests used $\alpha = 0.05$ (two-tailed) with effect sizes and confidence intervals reported throughout.

## 6. Results

### 6.1. Primary outcomes: Solution quality

Students in the AI Tutor condition substantially outperformed the Control group. Mean solution quality in the AI Tutor group was 82.1% of the rubric maximum (SD = 7.5), equivalent to 24.6 of 30 points, against 69.0% (SD = 15.0; 20.7 of 30) in the Control. That is an advantage of about 13 percentage points. This difference was statistically significant $t(82) = 5.06$, $p < .001$ with a large effect size (Cohen's d = 1.10). In practical terms, AI-tutored students were far more likely to produce correct, well-styled solutions meeting the rubric criteria. The score distribution shows that the AI Tutor not only raised average performance but also reduced the number of very low scores; few AI students failed to make meaningful progress on the tasks. An OLS regression model controlling for prior programming experience and pre-task self-efficacy confirmed these results, and a non-parametric Mann–Whitney U test yielded identical inference. The performance outcome was thus markedly improved by the guardrail-based AI tutoring intervention.

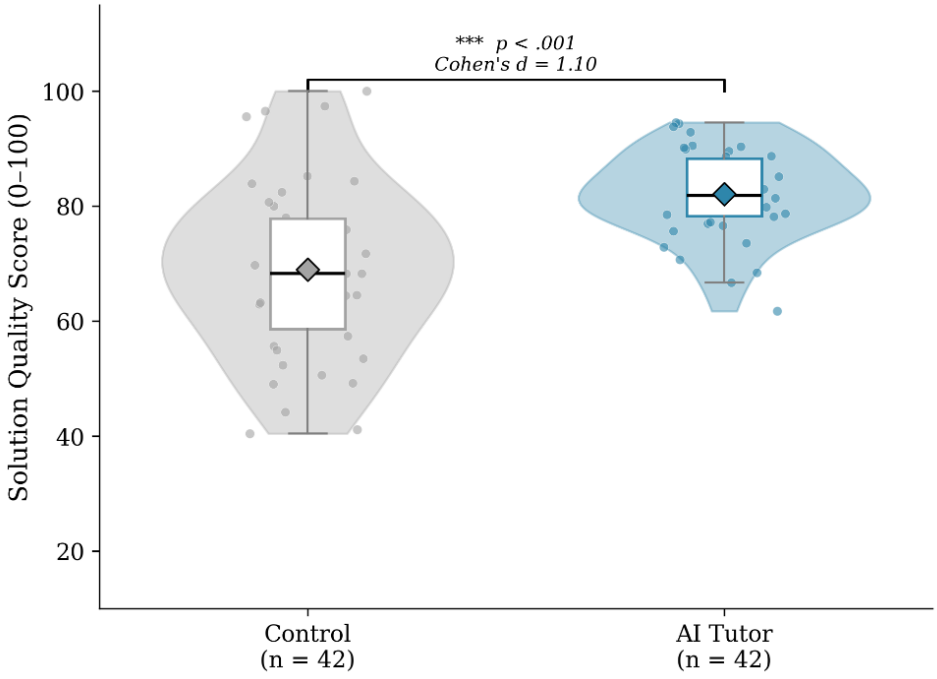


**Figure 3**. Distribution of rubric-based solution quality scores (0–30) for the Control and AI Tutor groups.

**Table 4**. Descriptive statistics and group comparisons.

| Measure | AI Tutor M (SD) | Control M (SD) | t / F | p | Cohen's d |
|---|---|---|---|---|---|
| Solution quality (% of 30-point rubric) | 82.1 (7.5) | 69.0 (15.0) | 5.06 | <.001 | 1.10 |
| Academic Independence Index (0-1) | 0.743 (0.121) | 0.758 (0.114) | — | .64 | 0.12 |

### 6.2. Primary outcomes: Academic independence

Students in the AI Tutor condition kept independence scores close to the Control group despite having on demand help available. On the 0-1 scale, the AI Tutor group scored M = 0.743 (SD = 0.121) against M = 0.758 (SD = 0.114) for Control, a difference of 0.015 (p = .64, Cohen's d = 0.12). The 95% confidence interval for the AI group mean was [~0.70, ~0.78], and the group difference fell well inside the equivalence bound of 0.30 that we set before the analysis. We interpret this comparison with a caveat. Because control students had no comparable help source, their trace-based scores partly reflect the absence of available assistance rather than a stronger disposition toward independent work. The between-group result is therefore best read as descriptive context. The stronger evidence that guardrails preserved independence comes from within the AI group itself: AII scores were high in absolute terms, and the dose-response patterns in Sections 6.3 and 6.5 show that the guardrails channeled students toward the help-seeking behaviors associated with better outcomes.

### 6.3. Mediation/Mechanism: How guardrails affect independence

A regression model (adjusted $R^2 \approx 0.15$, $p < 0.01$) revealed that initial independent problem-solving time was a significant positive predictor of independence ($\beta \approx +0.30$, $p = 0.003$): For each additional minute students worked before asking for help, their independence score increased (Figure 4; Table 5). Frequent direct solution requests predicted slightly lower independence ($\beta \approx -0.22$, $p = 0.028$). The hint delay encouraged productive struggle, and the no-solution policy compelled students to generate code themselves. Even students who attempted to over-rely on the AI were held in check; an unrestricted AI would have simply given answers (a behavior shown to impair learning [9]), but our tutor responded with guided hints, preventing a collapse of independent effort. Within the AI group, the correlation between AII and performance was modest ($r \approx 0.3$), suggesting that moderate help use could boost performance but excessive reliance did not yield proportionally higher scores.

**Table 5.** Regression: Predictors of Academic Independence (AI Tutor group).

| Predictor | β | SE | p |
|---|---|---|---|
| Time before first hint (min) | +0.30 | — | .003 |
| Frequency of direct solution requests | −0.22 | — | .028 |
| Adjusted $R^2$ = 0.15 | | | |

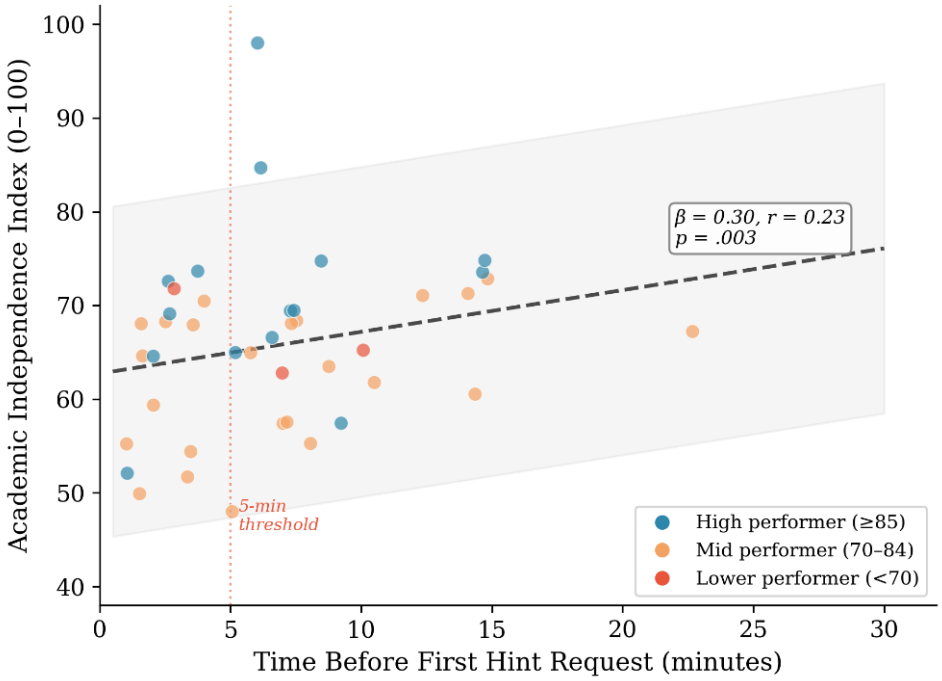


**Figure 4.** Time-to-first-hint (minutes of independent coding) plotted against Academic Independence Index scores for students in the AI Tutor condition.

### 6.4. Subgroup and robustness analyses

Participants were classified as novice or experienced by a median split on the prior-programming-experience measure from the pre-task survey, the same variable used to stratify randomization (Section 5.2). A two-way ANOVA (Condition × Prior skill) found major effects of condition ($F(1, 80) = 57$, $p < .001$) and prior skill ($F(1, 80) = 45$, $p < .001$). The interaction was not significant ($p = .19$), so novice and experienced students benefited to a similar degree (Figure 5). In absolute terms, less-prepared students gained slightly more (~20 points for bottom-quartile vs. ~10 points for top-quartile). The AI Tutor advantage persisted across difficulty levels, with the most pronounced benefit on the hardest tasks ($d \approx 0.8–1.0$). There was no task for which the AI condition underperformed the Control.

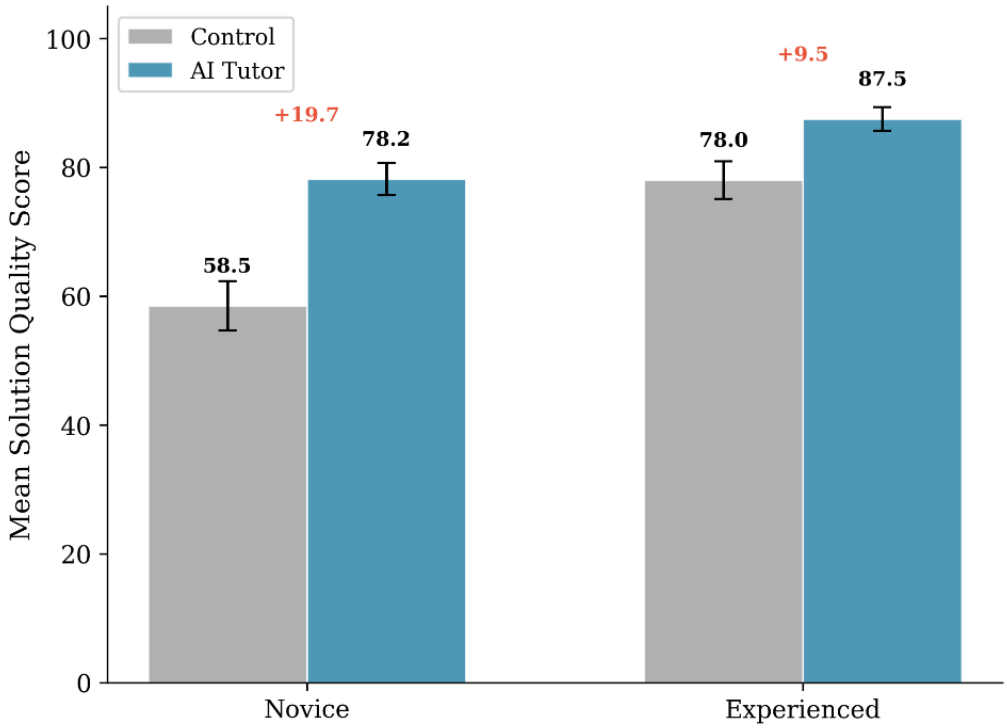


**Figure 5**. Mean solution quality scores by prior skill level (novice vs. experienced) and condition (Control vs. AI Tutor), with error bars indicating standard error.

### 6.5. Interaction analysis: Patterns linked to best outcomes

Two notable patterns emerged. First, delayed help-seeking was a hallmark of top performers: Students who spent at least 5 minutes coding independently before requesting a hint achieved significantly higher scores (+8 points, $p < 0.05$) and the highest independence indices. Delayed help-seeking was associated with better outcomes in this sample. The design did not let us separate this from student ability: Stronger students may both delay help and score higher, so we read the pattern as correlational. Second, the type of AI guidance requested mattered. Students engaging primarily in conceptual queries rather than directly asking for code outperformed peers by ~5–10% and maintained strong independence. Those who utilized the AI tutor for high-level hints or debugging tips achieved the highest mean scores and maintained strong independence. Qualitatively, these students treated the AI tutor like a human coach: Confirming their own ideas, asking specific questions, and iteratively refining solutions, which likely kept them in an active learning mode. In summary, the interaction analysis indicated that effective use of the AI tutor was key to maximizing its benefits. Students who integrated assistance strategically, delaying help and focusing on understanding hints rather than extracting answers, reaped the greatest gains in performance and self-reliance. These patterns fit the intent of the guardrails, which steer students toward delayed, conceptual help-seeking. Whether the behaviors cause the better outcomes cannot be settled from this design.

## 7. Discussion

### 7.1. Interpretation vs. Prior work

Our findings broadly align with research on SRL and student autonomy. The improvements echo prior interventions in computing education, showing that training students in SRL strategies leads to higher self-regulation and significantly better course performance [8]. Prior research has observed that an AI tutor giving hints (but not answers) can dramatically boost practice performance while yielding no loss in test performance relative to a control group [9]. This is broadly consistent with prior findings on SRL-focused interventions in learning contexts [8]. Our approach differs from one-off strategy advice in that the scaffolding sits inside every tutoring exchange: The attempt-before-hint rule, staged hints, and explain-first prompts operate on each task rather than as standalone recommendations. Compared with studies that report mixed correlations between SRL strategy use and grades, our design pairs behavioral traces with self-report, which may explain the clearer pattern we observe. The single-session scope limits how far this comparison can be pushed. The positive outcomes obtained by granting students autonomy within a structured framework support Self-Determination Theory's claim that autonomy is a key driver of motivation and optimal functioning [14,18]. Even in a rigorous domain like computer science, students responded well to autonomy-supportive strategies, suggesting that the "bright side" of autonomy is very much applicable when appropriate scaffolding is provided. Our work bridges SRL and autonomy-support theories by illustrating how structured autonomy can operate successfully in a computing curriculum, enriching pedagogical theory for STEM learning environments.

### 7.2. Design implications

The efficacy of the intervention traces to specific features. Combining explicit SRL instruction with guided practice and timely feedback was essential [8]. In our implementation, students received no separate SRL instruction. The tutor's interaction rules pushed them to plan and self-monitor in the moment, and they got structured feedback on each attempt. The staged hint system and explain-first requirement kept students actively engaged rather than passively consuming answers. The no-code-solutions guardrail was particularly critical: It physically prevented answer revelation and ensured that even students inclined to over-rely were redirected toward productive learning behaviors. Another design element that worked was encouraging early engagement through the attempt-before-hint policy, which prompted students to make genuine efforts before receiving any assistance. In summary, the design elements that mattered were ones that kept students inside the self-regulation cycle: structured questioning, guided practice with feedback, and the attempt-first requirement.

### 7.3. Practical guidance for instructors

Several recommendations emerge. Instructors should explicitly teach and model SRL strategies rather than assuming students know how to plan or monitor learning. Regular reflective assignments with feedback, such as weekly learning journals, should be routine course components. Consistent with [8], we recommend using intermediate deadlines or low-stakes checkpoints to reduce procrastination and encourage students to engage earlier. Our results support this recommendation indirectly, as the attempt-before-hint pattern was associated with greater academic independence. Building on the autonomy-support framework of [14,18], and in line with our findings, we

recommend that instructors create an autonomy-supportive learning environment by giving students meaningful choices while balancing independence with appropriate scaffolding.

### 7.4. Theoretical implications

Our study reinforces the notion that SRL skills are learnable and not fixed traits, aligning with Zimmerman's cyclical model of SRL [19]. The results support Self-Determination Theory's claim that autonomy is a key driver of motivation [14,18], demonstrating how structured autonomy can operate successfully in a computing curriculum. Our work bridges SRL and autonomy-support theories by illustrating their joint application in a computing context, enriching pedagogical theory for STEM learning environments.

## 8. Threats to validity and limitations

### 8.1. Internal validity

Several factors may threaten internal validity. Students were not blind to the intervention, which could introduce expectancy effects. Contamination between groups was possible since the study occurred in the same course across sections; students could potentially communicate. We attempted to minimize cross-talk, but some leakage may have occurred. Because the evaluation took place in a single 90-minute session, a novelty effect cannot be ruled out. Some of the AI group's engagement may reflect the newness of the tool rather than the guardrail design. Longer deployments are needed to test whether the effects hold once the novelty wears off.

### 8.2. Construct validity

Students within the same class share common factors (instructor, curriculum, peer interactions), meaning outcomes are not fully independent data points. A more robust analysis might use hierarchical models to account for class-level effects. Self-report SRL measures are known to have biases; students' perceptions of strategy use often diverge from actual behaviors. Our autonomy measurement relied on a proxy survey scale that may not fully capture its nuanced manifestations. We recommend follow-up studies with more direct behavioral measures of self-regulation (e.g., detailed activity logs or think-aloud protocols).

A further construct-validity limitation concerns the AII comparison between conditions. The trace-based components of the index could only register help-seeking toward sources that were available, and the two conditions differed in exactly this respect. The near-ceiling trace scores in the Control group are therefore partly an artifact of the design rather than direct evidence of higher autonomy. Within-group analyses (Sections 6.3, 6.5) do not suffer from this asymmetry, and we rely on them for our major claims about independence.

### 8.3. External validity

The study was limited to a single institution and specific course context at a large public university in Kazakhstan. The single-session timeframe means we do not know whether benefits persist beyond the immediate tasks or transfer to later coursework. Other instructors might implement the intervention with less fidelity, potentially yielding weaker outcomes. The results

should be interpreted as promising but context-bound, pending replication in broader educational settings.

### 8.4. Ethical considerations

The use of generative AI in education raises ethical issues that go beyond standard research-ethics procedures. Five concerns were especially important in shaping this study and define its limits. The first concern is academic integrity. Even tutors with guardrails can be misused, so students need a clear understanding of the difference between appropriate scaffolded support and inappropriate solution generation. In this study, this distinction was addressed through the no-direct-code policy and explain-first prompts. Academic integrity risks in AI-assisted learning are compounded by pre-existing patterns of academic dishonesty and procrastination in digital and distance-learning contexts [16], underscoring the importance of guardrails that keep students engaged in genuine problem-solving rather than shortcut-seeking. However, broader institutional policies on AI-assisted academic work need further development. The second concern is equity of access. Students may differ in their familiarity with AI tools, and not all students have equal access to high-quality LLM services. Without careful implementation, these differences could widen achievement gaps. Large-scale deployment should therefore ensure that all students receive comparable tutoring support. The third concern is data privacy. Interaction logs can contain detailed information about students' reasoning, mistakes, and help-seeking behavior. These data must be stored, anonymized, and retained according to institutional data-protection rules and, in this context, Kazakhstan's national framework for the responsible use of AI in education. The fourth concern is algorithmic bias and hallucination. LLM outputs may be inaccurate or culturally biased, and novice learners may accept them without enough critical evaluation. The explain-first requirement and rubric-based feedback help reduce this risk, but they do not remove the need for instructor oversight. The final concern is autonomy and consent. Students should know how the tutor differs from an unrestricted AI assistant, what data is being collected, and how that data will be used. Informed consent was obtained in this study, but continued transparency will be necessary as these systems become part of regular coursework. Future deployments of guardrail-based AI tutors should therefore include clear ethical guidelines covering academic integrity, equity, privacy, oversight, and learner consent.

## 9. Conclusions and future work

### 9.1. Summary of contributions

This work makes several contributions to computing education research and practice. We designed and empirically evaluated a novel guardrail-based AI tutoring intervention that successfully bolsters students' problem-solving quality while preserving academic independence at a Kazakhstani university. To our knowledge, this is one of the few controlled studies in a computer science course to show that guardrailed AI assistance can raise problem-solving performance without measurable loss of student independence during use. We demonstrated that a structured approach, combining guardrailed AI assistance, Socratic questioning, reflective prompts, and autonomy-supportive teaching, leads to more engaged, proactive learners. The Academic Independence Index offers a first operational measure of learner autonomy in AI-mediated settings. Initial evidence for it is a moderate correlation with self-reported autonomy ($r \approx 0.6$); fuller psychometric validation is outlined in Section 9.2. Our practical framework and detailed description of what design elements proved

effective provide a blueprint that educators can adapt to foster student autonomy and metacognitive skills in their own courses. Our mixed-methods analysis combines quantitative performance data with qualitative insights from interaction traces, offering a clearer understanding of how guardrail-based tutoring works and why it supports learning.

### 9.2. Future work

Several directions for future research follow from this study. First, although we examined help-seeking patterns and students' sense of ownership over their solutions, we did not directly measure more detailed behavioral factors such as reduced procrastination, the depth and quality of student reflections, or long-term changes in self-regulation strategies. In future studies, researchers should include dedicated measures, such as timestamped engagement logs, coded reflection journals, and validated self-regulated learning inventories collected at multiple points in time. These tools would help clarify how guardrail-based tutoring influences learning outcomes. Second, the study should be replicated across institutions, course levels, and programming languages to test whether the findings hold beyond the single Kazakhstani CS1 cohort examined here. Third, longitudinal follow-up is needed to track students into later courses and determine whether the academic independence observed in this study leads to lasting transfer of self-regulated learning skills. Fourth, the Academic Independence Index requires further psychometric validation. This could include comparisons with think-aloud protocols and instructor assessments of student autonomy. Finally, researchers should explore adaptive guardrail policies that adjust the level of scaffolding to individual learner profiles, as well as how guardrail-based tutoring works in collaborative and peer-learning environments.

## Author contributions

Raushan Sambetova: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization. Lyazzat Zhaidakbayeva: Conceptualization, methodology, software, validation, formal analysis, investigation, writing—original draft preparation, writing—review and editing, visualization.

## Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

## Conflict of interest

The authors declare no conflicts of interest in this manuscript**.**

## Ethics declaration

This study was reviewed and approved by the Research Ethics Committee of JSC «International University of Tourism and Hospitality», and was conducted in accordance with the Declaration of Helsinki.

All participants received a written information sheet describing the purpose of the study, the data to be collected, the involvement of a third-party large language model service, and their right to

withdraw, and all provided written informed consent before the session. For participants under 18 years of age, written consent was obtained from a parent or legal guardian in addition to the participant's assent. Participation was voluntary and independent of course assessment: students who declined to participate completed the same laboratory session and were not disadvantaged in any way, and study data were not used in the calculation of course grades.

Interaction logs, code snapshots, and survey responses were linked only to pseudonymous study identifiers; no directly identifying information was stored in the study database or transmitted to the language model provider. Task-related messages were processed through the OpenAI API for the sole purpose of generating tutor responses, and students were informed of this before consenting. Solutions were anonymized prior to rubric-based grading so that raters were blind to condition. Data are stored on encrypted institutional storage and will be retained for 5 years in accordance with JSC «International University of Tourism and Hospitality»'s data-protection rules and the national framework governing the responsible use of artificial intelligence in education.

Students assigned to the control condition were given access to the tutoring system for independent practice after all study data had been collected, so that no participant was disadvantaged by group assignment.

## References


1. Anders, A.D. and Speltz, E.D., Developing generative AI literacies through self-regulated learning: A human-centered approach. *Computers & Education: Artificial Intelligence*, 2025, 9: 100482. https://doi.org/10.1016/j.caeai.2025.100482
2. Lai, C.-H. and Lin, C.-Y., Analysis of learning behaviors and outcomes for students with different knowledge levels: A case study of intelligent tutoring system for coding and learning (ITS-CAL). *Applied Sciences*, 2025, 15(4): 1922. https://doi.org/10.3390/app15041922
3. Ministry of Education of the Republic of Kazakhstan & Ministry of Digital Development, Innovation and Aerospace Industry. (2025). Conceptual Framework for the Implementation of Artificial Intelligence in Secondary, Technical and Vocational, and Post-Secondary Education for 2024–2029 (Joint Order No. 221, 18 September 2025). Astana, Kazakhstan. Retrieved from: [https://prg.kz]
4. Yan, Y.M., Chen, C.Q., Hu, Y.B. and Ye, X.D., LLM-based collaborative programming: Impact on students' computational thinking and self-efficacy. *Humanities and Social Sciences Communications*, 2025, 12(1): 149. https://doi.org/10.1057/s41599-025-04471-1
5. Hou, C., Zhu, G., Sudarshan, V., Lim, F.S. and Ong, Y.S., Measuring undergraduate students' reliance on generative AI during problem-solving: Scale development and validation. *Computers & Education*, 2025, 234: 105329. https://doi.org/10.1016/j.compedu.2025.105329
6. Stojanov, A., Liu, Q. and Koh, J.H.L., University students' self-reported reliance on ChatGPT for learning: A latent profile analysis. *Computers & Education: Artificial Intelligence*, 2024, 6: 100243. https://doi.org/10.1016/j.caeai.2024.100243
7. Létourneau, A., Deslandes Martineau, M., Charland, P., Karran, J.A., Boasen, J. and Léger, P.M., A systematic review of AI-driven intelligent tutoring systems (ITS) in K–12 education. *npj Science of Learning*, 2025, 10(1): 29. https://doi.org/10.1038/s41539-025-00320-7
8. Domino, M. and Shaer, C.A., Using a wider digital ecosystem to improve self-regulated learning. *Frontiers in Education*, 2025, 10: 1487344. https://doi.org/10.3389/feduc.2025.1487344
9. Bastani, H., Bastani, O., Sungu, A., Ge, H., Kabakcı, Ö. and Mariman, R., Generative AI

without guardrails can harm learning: Evidence from high school mathematics. *Proceedings of the National Academy of Sciences*, 2025, 122(26): e2422633122. https://doi.org/10.1073/pnas.2422633122

10. Perikos, I., Grivokostopoulou, F. and Hatzilygeroudis, I., Assistance and feedback mechanism in an intelligent tutoring system. *International Journal of Artificial Intelligence in Education*, 2017, 27(3): 475–514. https://doi.org/10.1007/s40593-017-0139-y
11. Chen, D., Zhang, Y., Luo, H., Gao, Z., Yu, L. and Lin, Y., Exploring the impact of scaffolding in programming on students' computational thinking: Evidence from a three-level meta-analysis. *Journal of Educational Computing Research*, 2026. https://doi.org/10.1177/07356331251386618
12. Yan, L., Martinez-Maldonado, R., Jin, Y., Echeverria, V., Milesi, M., Fan, J., et al., The effects of generative AI agents and scaffolding on enhancing students' comprehension of visual learning analytics. *Computers & Education*, 2025, 234: 105322. https://doi.org/10.1016/j.compedu.2025.105322
13. Hou, C., Zhu, G., Liu, Y., Sudarshan, V., Chong, J.L.L., Zhang, F.Y., et al., The effects of critical thinking intervention on reliance behaviors, problem-solving quality, and creativity during human-Generative AI collaborative learning. *Computers & Education*, 2026, 247: 105576. https://doi.org/10.1016/j.compedu.2026.105576
14. Johansen, M.O., Eliassen, S. and Jeno, L.M., The bright and dark side of autonomy: How autonomy support and thwarting relate to student motivation and academic functioning. *Frontiers in Education*, 2023, 8: 1153647. https://doi.org/10.3389/feduc.2023.1153647
15. Bassner, P., Lenk-Ostendorf, B., Beinstingel, R., Wasner, T. and Krusche, S., Less stress, better scores, same learning: The dissociation of performance and learning in AI-supported programming education. *Computers & Education: Artificial Intelligence*, 2025, 10: 100537. https://doi.org/10.1016/j.caeai.2025.100537
16. Aruğaslan, E., Examining the relationship of academic dishonesty with academic procrastination, and time management in distance education. *Heliyon*, 2024, 10(19): e38827. https://doi.org/10.1016/j.heliyon.2024.e38827
17. Brunton, R.J., Rhazzafe, S., Moodley, R., Kuhn, S., Caraffini, F., Wilford, S., et al, Using generative artificial intelligence to enhance the performance of disadvantaged students in secondary education. *Social Sciences & Humanities Open*, 2025, 12: 102110. https://doi.org/10.1016/j.ssaho.2025.102110
18. Deci, E.L. and Ryan, R.M., The “what” and “why” of goal pursuits: Human needs and the self-determination of behavior. *Psychological Inquiry*, 2000, 11(4): 227–268. https://doi.org/10.1207/S15327965PLI1104_01
19. Zimmerman, B.J., Becoming a self-regulated learner: An overview. *Theory Into Practice*, 2002, 41(2): 64–70. https://doi.org/10.1207/s15430421tip4102_2

# Appendix

## Appendix A. Tutor implementation details

A.1 System architecture. The tutor was implemented as a browser-based web application with a React front end, a Monaco-based Python code editor, and a chat panel displayed beside the programming task. The back end was a Python FastAPI service that handled authentication for anonymous study IDs, stored task state, applied guardrail checks, and forwarded approved chat requests to GPT-4o through the OpenAI API. The API layer attached the current task, hint level, recent code snapshots, and the system prompt to each model request, then post-processed the response before returning it to the student interface. Interaction logs, code snapshots, hint counters, and survey-linked study identifiers were stored in a PostgreSQL database using pseudonymous student IDs. The architecture was: Student browser -> React task/chat interface -> FastAPI guardrail layer -> GPT-4o API -> response filter -> interface; in parallel, all events were written to the PostgreSQL interaction-log database.

A.2 System prompt. The tutor operated under the following system prompt, reproduced verbatim:

1. You are an Independence-by-Design programming tutor for first-year informatics students solving introductory Python tasks. Your purpose is to help the student reason, debug, and reflect while preserving the student as the author of the solution.
2. Always follow an explain-don't-solve policy. Ask the student to describe their current idea before giving help. Prefer Socratic questions, conceptual explanations, and small diagnostic hints. Do not write any example code blocks. You must not write code for the student.
3. Use staged hints. At Level 1, restate the problem and point to the relevant concept. At Level 2, identify the likely source of difficulty. At Level 3, describe the algorithmic step in natural language. At the bottom-out level, explain the complete approach in prose, but still do not provide code, pseudocode that can be copied directly, or a finished solution.
4. Require an attempt before assistance. If the student has not submitted code or explained an approach, ask them to write one or two sentences about their plan or to submit a partial attempt before receiving a content hint.
5. When reviewing student code, comment on correctness, completeness, readability, and efficiency using the task rubric. Point to the location or type of problem, but do not replace the student's code. If the student asks for the answer, a full program, or copy-ready code, refuse briefly and redirect to the next reasoning step.
6. Encourage reflection after substantial help. Ask the student to explain what changed in their thinking and why the next revision should work. Keep responses concise, supportive, and appropriate for novice programmers.

A.3 Guardrail rules. Table A1 lists each guardrail, its trigger, and its enforcement mechanism.

| Guardrail | Trigger | Enforcement |
|---|---|---|
| Staged hint levels | Student requests a hint | Hint level counter per student and task; each request advances one step through four levels: Level 1 conceptual nudge, Level 2 targeted diagnostic hint, Level 3 natural-language algorithm guidance, and bottom-out explanation. The final level explains the approach but contains no code. |
| Attempt-before-hint | Hint requested with no prior code submission or written | Hard block in the web/API layer: the tutor asks the student to submit a partial attempt or describe the intended approach before a |

| | | |
|---|---|---|
| | approach | content hint is released. The event is logged as an explain-first turn. |
| No direct code solutions | Model output contains a Markdown code block, copy-ready code, or a direct request for a complete solution | System-prompt prohibition plus deterministic post-processing. Markdown fenced code blocks and copy-ready line blocks are removed by a regular-expression filter; if removal would make the response unclear, the back end asks the model to regenerate the reply as prose-only guidance. |
| Explain-first prompts | Before the first content hint and after the bottom-out hint | Tutor asks the student to state their reasoning before receiving help and to summarize the correction after the strongest hint. Both replies are logged as dialogue_turn events and linked to the task ID. |

A.4 Hint delay. A five-minute minimum independent-work period was enforced before the first content hint for each task. During this period, students could submit or run code and could receive interface-level reminders to describe their approach, but the tutor did not release Level 1, Level 2, Level 3, or bottom-out hints until the five-minute threshold had elapsed. The delay was implemented in the back-end guardrail layer using the task-start timestamp and was reset separately for Task 1 and Task 2. Students who reached the threshold and still requested support then entered the staged-hint sequence described in Table A1. This implementation is consistent with the results section, where delayed help-seeking is treated as both a system-supported behavior and an observed student-use pattern.

## Authors' biographies

**Raushan Sambetova** is a staff member at the International University of Tourism and Hospitality in Turkistan, Kazakhstan. She completed her bachelor's degree in Informatics at the International Kazakh-Turkish University (2002–2007). She then obtained a master's degree (2015–2017) and completed her doctoral studies (2018–2021) in Informatics at M. Auezov South Kazakhstan State University. She is currently conducting research on the development of a pedagogical model for using generative AI in informatics education and examining its effects on the quality of problem-solving and students' academic independence. Raushan Sambetova's research interests lie at the intersection of artificial intelligence and education, with a focus on the safe and effective integration of generative AI tools into the learning process to enhance educational outcomes without compromising students' academic autonomy.

**Lyazzat Zhaidakbayeva** is the Head of the Department of Informatics and a Research Fellow at M. Auezov South Kazakhstan Research University. She holds the degree of Candidate of Sciences (2010) and began her research career as a research applicant in 2000.

In 1996, she graduated from Khoja Akhmet Yassawi International Kazakh-Turkish University.

Throughout her professional career, she has held academic and leadership positions at several leading universities. She is an expert in educational programs at South Kazakhstan Pedagogical University. In 2024, she completed professional development at Lancaster University under the Bolashak International Scholarship Program.

In 2026, she was awarded the Certificate of Honor by the Ministry of Science and Higher Education of the Republic of Kazakhstan in recognition of her contributions to higher education and scientific research.

Her research interests include computer science education, STEM education, artificial intelligence in education, digital technologies, and educational innovation. She has published numerous scientific papers in high-impact international journals indexed in Scopus and Web of Science.