December 2024

Design and Evaluation Attributes for Scalable, Cost-Effective Personalization of LLM Tutors in Programming Education

Arun Rai
Georgia State University, arunrai@gsu.edu

Liwei Chen
University of Cincinnati, liwei.chen@uc.edu

Cynthia Breazeal
Massachusetts Institute of Technology, CYNTHIAB@MEDIA.MIT.edu

Balasubramaniam Ramesh
Georgia State University, bramesh@gsu.edu

Yuan Long
Georgia State University, ylong4@gsu.edu

See next page for additional authors

Presenter Information
Arun Rai, Liwei Chen, Cynthia Breazeal, Balasubramaniam Ramesh, Yuan Long, and Andrea Aria

Design and Evaluation Attributes for Scalable, Cost-Effective Personalization of LLM Tutors in Programming Education

Completed Research Paper

Arun Rai
Georgia State University
Atlanta, GA, USA
arunrai@gsu.edu

Liwei Chen
University of Cincinnati
Cincinnati, OH, USA
liwei.chen@uc.edu

Cynthia Breazeal
Massachusetts Institute of Technology
Cambridge, MA, USA
cynthiab@media.mit.edu

Balasubramaniam Ramesh
Georgia State University
Atlanta, GA, USA
bramesh@gsu.edu

Yuan Long
Georgia State University
Atlanta, GA, USA
ylong4@gsu.edu

Andrea Aria
Georgia State University
Atlanta, GA, USA
aaria@gsu.edu

Abstract

This paper examines the design and evaluation of Large Language Model (LLM) tutors for Python programming, focusing on personalization that accommodates diverse student backgrounds. It highlights the challenges faced by socioeconomically disadvantaged students in computing courses and proposes LLM tutors as a solution to provide inclusive educational support. The study explores two LLM tutors, Khanmigo and CS50.ai, assessing their ability to offer personalized learning experiences. By employing a focus group methodology at a public minority-serving institution, the research evaluates how these tutors meet varied educational goals and adapt to students' diverse needs. The findings underscore the importance of advanced techniques to tailor interactions and integrate programming tools based on students' progress. This research contributes to the understanding of educational technologies in computing education and provides insights into the design and implementation of LLM tutors that effectively support equitable student success.

Keywords: LLM, personalization, programming education

Introduction

In the fast-paced digital economy, where computing-related careers can be synonymous with opportunity, fostering equitable student success is of paramount importance. Students from low-income and first-generation backgrounds face unique hurdles including financial strain, social exclusion, lack of mentoring resources, and lack of digital resources such as high bandwidth and up-to-date digital devices. Stereotype threats and high DFW rates in foundational courses which disproportionately affect these students underscore the need for inclusive education solutions that empower rather than inhibit (Gumbel 2020).

A major foundation for an educational pursuit in computing fields is the Introduction to Computing course, typically anchored in programming languages such as Python. Success in this initial foundational phase can instill self-efficacy and inspire a desire to pursue a computing major. However, without early positive interventions, challenges faced in this course can seriously erode these motivations (Tinto 1993). The support system within these courses, particularly the role of human tutors, is crucial. In addition to limited access to such resources, not all students can readily access even the available help due to work commitments, family responsibilities, and technological constraints (Warschauer and Matuchniak 2010). As a result, human tutors are often unavailable when most needed. Additionally, human tutors cannot easily extend their services to more students without reducing support quality or increasing costs, limiting scalability (Bettinger and Baker 2014). Human tutors may also struggle to relate to individual students' needs due to their different backgrounds (Yosso 2005). Moreover, the psychological barrier of fearing judgment for asking questions can further distance students from the help they need (Steele 1997).

As students increasingly turn to generative artificial intelligence (GenAI) resources like ChatGPT, they encounter both benefits and limitations. While ChatGPT typically provides solutions to commonly used exercises and exam questions, this can significantly limit learning by reducing the need for problem-solving and critical thinking (Zawacki-Richter et al. 2019; Extance 2023). Arecent MIT report (Klopfer et al. 2024) explores how these challenges be addressed by leveraging GenAI technologies to create "tutoring or study buddy" systems. The development of large language model (LLM) tutors is an innovative approach to design *cost-effective, personalized* learning experiences *at scale*, motivating us to explore their potential in addressing the diverse educational needs of students. The unique capabilities of LLM tutors to engage in human-like conversations enhances student motivation and engagement. They provide 24/7 accessibility, crucial for educational equity, and adapt in real-time to individual learning needs, offering personalized support that traditional human tutoring often cannot match. Two exemplar LLM tutors are *Khanmigo*, developed by Khan Academy which supports a wide range of subjects and levels of education (Extance 2023), and *CS50.ai*, developed by Harvard University specifically for Python programming (Liu et al. 2024). These platforms are designed to provide help in a non-intimidating environment, thereby eliminating many traditional tutoring challenges. They also allow for the evaluation of how well such systems meet the diverse educational goals and adapt to the varied lives and experiences of students from different socio-economic backgrounds. Moreover, the potential privacy safeguards these platforms can afford will encourage free inquiry without the fear of stigmatization. However, the promise of technology in transforming education often falls short (Reich 2020). The potential for LLMs to truly personalize learning and disrupt traditional methods remains a significant question. Thus, the design and subsequent evaluation of these tutors must carefully consider potential drawbacks, such as over-generalizing learning styles by imposing the learning styles of certain groups, communicating with biases, and hallucinations.

Our review of the literature on AI in computing education reveals that important progress has been made in understanding the potential advantages and risks of LLM tutors (Denny et al. 2024). Moreover, the information systems (IS) literature has established a solid base of understanding on how to determine design requirements and evaluation considerations from the perspectives of users and other stakeholders including developers, regulators and auditors (Hamilton and Chervany 1981; DeLone and McLean 1992; Xu et al. 2013). This requires discovering the variety of design and evaluation considerations that fulfill the diverse needs of stakeholders in a specific socio-economic and learning institution context.

Accordingly, our research objective is to identify the design and evaluation attributes of LLM tutors that support scalable, cost-effective personalization, particularly at universities with high student diversity. We consider diversity in terms of demographics (gender, ethnicity) and socio-economic backgrounds (income levels, work and family demands), which manifest as diverse learning styles, performance, and needs for support. Such high diversity characterizes the students at MSIs, the context in which we situate our work. These attributes can provide insights to guide the design of LLM tutors and to evaluate their effectiveness in meeting the varied learning preferences and needs of students. However, our near-term solutions are quite limited, as we often have to rely on more affordable foundation models like GPT-3.5, which may not be trained on the most recent content. This limitation highlights the importance of our research, which advocates aligning, integrating, and adapting to ensure LLM tutors to effectively incorporate the most current and relevant educational materials.

We employ a two-step process to lay the foundations for our empirical work. First, we examine the literature within the field of Information Systems (IS) to identify design attributes that have been previously recognized as crucial for educational support systems and user interactions. Second, we employ the revised Bloom's taxonomy (Anderson et al. 2001), a hierarchical model used to classify learning, teaching, and assessing into levels of cognitive complexity, to differentiate between learning objectives relevant to Python programming courses, such as understanding fundamental concepts, debugging code, preparing for quizzes, and developing complete programs. By applying Bloom's taxonomy, we establish a structured framework to assess how well LLM tutors can cater to each of these learning objectives. This dual approach of reviewing IS literature for design attributes and categorizing learning objectives through the Bloom's taxonomy lays a solid foundation for the subsequent stages of our empirical research and to discern how we go beyond the IS research in understanding how to evaluate LLM tutors.

Our empirical research is conducted at a prominent urban minority-serving university (MSI) classified as R1 under the Carnegie Classification of Institutions of Higher Education with a nationally top-10 ranked Information Systems program. To answer our research question, we juxtapose the problem (i.e., the specific learning objectives) and the *diversity of students* with current cost-effective LLM solutions (i.e., Khanmigo, CS50.ai). Through their active use of the tools for a particular goal, we can assess whether their respective needs are fulfilled effectively and which needs are not fulfilled. We utilize a focus group methodology, taking place in an in-person setting, where a group of students actively engage with *Khanmigo* and *CS50.ai* to assist their learning for specific learning objectives. Through analysis of survey responses from focus group participants, chat logs with the LLM tutor, and discussions, we identify how different features of LLM tutors align with and support the varied learning objectives of students. This approach allows us to pinpoint the characteristics of LLM tutors that are most instrumental in facilitating effective learning experiences and how student preferences and needs for support from LLMs vary. Overall, we contribute to how to design and evaluate LLM tutors for equitable student success in computing education, particularly programming.

Background Literature

We consider two streams of literature to inform our work: AI technologies in computing education, referred to as AI Ed-Tech, and the IS literature on the design and evaluation requirements based on user preferences.

AI Technologies in Computing Education

Educational technologies have traditionally employed *instructionist* approaches, where learning is teacher-directed and students passively receive information. Platforms such as *Coursera* and *edX* exemplify this approach through their Massive Open Online Course. While these platforms have widened access to education, they often struggle with engagement and retention among diverse learners due to their one-size-fits-all nature (Reich 2022). In contrast, *constructionist* approaches, championed by Seymour Papert, emphasize active, learner-centered education (Reich 2022). Platforms like *Scratch* and *MIT App Inventor* are examples of this approach in computing education, allowing learners to actively engage with content through hands-on projects, which supports deeper understanding and retention.

In the university setting, the integration of AI with other digital technologies is reshaping educational strategies, especially in supporting students' coursework. AI-enabled academic engagement tools represent a significant advancement in educational technologies to enhance the learning experience by offloading routine tasks and promoting active learning. For example, Jill Watson, an AI teaching assistant developed at the Georgia Institute of Technology, manages routine inquiries and grading (Goel and Polepeddi 2018). This support allows instructors to focus more on direct engagement with students and supports students' overall academic learning experiences.

Further advancing educational technologies, GenAI has brought transformative changes in computing education (Finnie-Ansley et al. 2022; Becker et al. 2022). LLMs are central to this evolution as they dynamically generate content that adapts to diverse educational needs (Extance 2023). Among the various types of LLMs, *code LLMs* are trained on extensive programming datasets. These models assist students by enhancing their understanding and generation of code, providing timely, real-time assistance that simplifies complex concepts. *Chat LLMs* utilize conversational interfaces that personalize feedback based on individual student responses, creating engaging and responsive learning environments. *Multi-modal*

LLMs integrate various data types, such as text and images, to enrich educational content and make learning more interactive and accessible, particularly in visually oriented subjects.

The applications of GenAI in education are diverse and innovative. For instance, Quizlet's AI feature personalizes quiz questions and tailors them to challenge individual learning processes. Coursera leverages GenAI to create personalized learning pathways and succinctly summarizes video lectures, aligning content with specific career goals and learning preferences. LLM tutors like Khanmigo and Harvard's CS50.ai use these technologies to offer customized tutoring and enhanced problem-solving support, boosting learning efficiency and effectiveness in computing education.

Overall, the stream of research on LLM in computing education has surfaced the potential of the technology to transform teaching methodologies and learning experiences (Denny et al. 2024). LLMs have demonstrated high performance across various educational applications, notably in assisting with standardized test preparation and providing real-time academic support in computing fields. Specifically, LLMs like ChatGPT have demonstrated capabilities beyond simple question answering. They serve as effective tools for reading and writing assistance, helping students perform better in certain academic courses. Moreover, LLMs have been shown to have the potential to create personalized learning environments and automating assessment processes, thus supporting both student and teacher needs.

Despite their advantages, the deployment of LLMs in education raises several ethical and practical concerns. These include issues such as plagiarism, potential biases in AI-generated content, overreliance on technology, and usability for non-English speakers. These challenges necessitate careful consideration to ensure equitable and effective use of LLMs in educational contexts, echoing Reich (2020)'s caution that new technologies often do not lead to the anticipated transformative changes.

Design and Evaluation of IS from Users' Perspective

We reviewed the existing literature to identify key criteria, based on users' perspectives, to design and evaluate information systems in general and educational support systems in particular.

Availability and affordability. A foundational aspect of IS evaluation lies in the cost-benefit analysis, which underscores the importance of affordability as a key criterion. Affordability ensures that technological solutions reach a broad user base, particularly benefiting underrepresented and economically disadvantaged groups (Hsieh et al. 2008). For educational technologies, it advocates for the development of educational technologies that are affordable on widely available hardware and do not require high bandwidth or the latest technology to operate effectively (Rodriguez-Segura 2022).

Information quality and system quality. Information quality and system quality are critical IS evaluation criteria (DeLone and McLean 1992). On the one hand, high-quality information, characterized by accuracy, relevance, and timeliness, enhances user satisfaction (Xu et al. 2013). For educational technologies, it is crucial that systems provide contextually appropriate and pedagogically tailored information to boost comprehension and retention (Ma et al. 2014). On the other hand, system quality involves the technical capabilities of an IS to provide reliable, flexible, and responsive interactions, and is crucial for maintaining user engagement and system effectiveness. System quality also encompasses security and privacy measures like advanced encryption and secure protocols to protect data integrity and foster a secure learning environment (Pearce et al. 2022).

Service quality and personalization. In recognition of the expanding service role within the IS function, the framework for system evaluation has evolved to include service quality (Kettinger and Lee 1994; Pitt et al. 1995; Jiang et al. 2002). Personalization emerges as a core component, reflecting the system's ability to adapt its services to meet individual user preferences and needs (Xu et al. 2013). In educational technologies, personalization includes adapting content to students' socio-economic backgrounds, personality traits, academic histories, and communication styles. Technologies like LLM tutors utilize adaptive mechanisms to adjust the complexity and delivery style of content in real-time, based on ongoing assessments of users' learning progress (Ma et al. 2014). Moreover, empathetic personalization focuses on recognizing and responding to learners' emotional states, offering real-time motivational feedback and support (Ma et al. 2014).

Knowledge gaps. Despite the recognized importance of personalization in IS evaluation, knowledge gaps remain in the following aspects. *First,* discussions of IS evaluation remain at a generic, system-wide level, which may not adequately address the nuanced requirements of specific educational functions. For instance, the criteria useful for evaluating an LLM tutor's effectiveness in aiding conceptual understanding in programming might differ markedly from those needed for debugging tasks. There is a critical need to develop function-level evaluation criteria of LLM tutors in specific educational contexts. *Second,* the current personalization design may not effectively capture the hidden learning and engagement status, which are crucial for providing truly personalized learning experiences. Research is needed to develop mechanisms to detect these hidden statuses to allow for more accurate and timely adjustments in the learning process. *Third,* another significant challenge is designing LLM tutors that not only offer personalization but also do so at scale, catering to the diverse needs of a large number of students. Exploring innovative design and deployment strategies that leverage data analytics and machine learning could provide pathways to achieving this goal.

Learning Objectives for Programming Based on Bloom's Taxonomy

As we delve into the nuances of educational technology, it becomes essential to align the personalization design with structured educational frameworks that guide content delivery and assessment. One such framework is Bloom's Taxonomy, a well-established model that categorizes learning objectives and outcomes. Developed in the mid-20th century by Benjamin Bloom and colleagues (1956), the taxonomy categorizes educational goals into cognitive, psychomotor, and affective domains, ranging from simple recall of facts to complex cognitive processing. Subsequent revisions have expanded the taxonomy to better suit the evolving needs of modern education (Adams 2015). A commonly used revision introduced a two-dimensional framework that separates cognitive processes from knowledge types, providing a more detailed structure for designing and assessing educational content (Krathwohl 2002). This revised taxonomy crafts learning objectives that provide a comprehensive path from basic understanding to creative application in various education settings, including programming education (see Table 1).

The effectiveness of the revised Bloom's taxonomy has been demonstrated in prior studies. Lahtinen (2007) evaluated student performance in an introductory programming course across various taxonomy levels. Further applications of the revised taxonomy in programming education have been explored by Starr et al. (2008), Oliver et al. (2004), and Shneider and Gladkikh (2006). Starr's work distinguishes between basic and advanced skills—categorizing foundational skills like Remember and Understand for beginners, and more complex skills such as Evaluate and Create for advanced learners. Similarly, Oliver's team developed a Bloom rating system to assess educational effectiveness across multiple courses, demonstrating the taxonomy's adaptability to diverse educational needs. Shneider and Gladkikh (2006) applied the taxonomy to structure questioning techniques in various subjects, including Programming with Visual Basic. Their work illustrates how the taxonomy can guide the instructional strategies tailored to diverse assessment needs and achieve personalized self-directed learning. Together, these applications underscore the robustness of Bloom's taxonomy as a strategic tool in programming education, enhancing the personalization and adaptability of teaching strategies to accommodate varying learner profiles and needs.

Bloom's Taxonomy	**Description**	**Examples in Programming Education**		
		Problem Type	**Provided Input**	**Expected Solution**
Remember	Recognize and recall key facts and basic concepts	Concept Comprehension: Define the given terms	Concept or term	Description
Understand	Interpret and explain ideas or concepts	Concept Comprehension: Understand a new programming concept	Explanation of a new programming concept	One statement code or a short explanation about this concept
Apply	Use the information in new and different settings	Debugging: Apply standard debugging techniques to fix a code issue	A reminder of a concept already known by the students	Answering questions related to applying already known concept
Analyze	Examine ideas and identify relationships by breaking information into component parts	Debugging: Find the error in the given snippet of a code	A block of a code and description of its functionality having some syntax, runtime and logic errors	A functional block of code without any type of errors

Evaluate	Assess arguments or solutions and make decisions based on reasoned judgment	Concept Comprehension: Justification of using a new concept in the code Debugging: Evaluate different solutions to fix complex issues	New concept provided	Answer questions that show the reasoning behind using this new concept
Create	Generate new ideas, concepts, or solutions by combining elements in novel ways	Program Development: Writing a program	Requirements of the program need to be written	A functional program

Table 1. Learning Objectives Based on Bloom's Taxonomy in Programming Education

Methodology

Empirical Setting

Python Course at a Minority-Serving Institution

The study was conducted in an introductory Python programming course at a prominent urban MSI with a highly regarded IS program. Many of these students juggle part-time or full-time jobs to manage their tuition and living expenses, which leaves them limited time for academic activities such as completing assignments, taking quizzes, and attending lectures. The course is a core requirement in the undergraduate IS program and teaches the basics of programming in Python. It requires at least a 'B' grade for students to progress in their majors. There are three in-person sections and one online section, each with an enrollment of 55 students, totaling 220 students. On average, students spend between 2.5 and 5 hours per week on coursework. To support classroom learning, the course includes two optional virtual lab sessions each week. These sessions, which last about 1.5 hours each, are facilitated by teaching assistants. Despite being designed to support student learning, attendance at these labs is low, often ranging from 10% to 15%. The low attendance rates can be attributed to scheduling conflicts and other challenges that the students face.

Two Cost-Effective LLM Tutors

For our study, we chose to evaluate two LLM tutors, Khanmigo and CS50.ai, due to their popularity and unique features to create personalized learning experiences, and their affordability. Khanmigo, developed by Khan Academy, is a refined GPT-4 model that caters to students, teachers, and parents across various subjects including math, physics, and programming (Extance 2023). It offers a subscription at four dollars monthly or 44 dollars annually. Distinctively, Khanmigo allows users to interact with AI-simulated personas of historical or literary characters, making learning more relatable and immersive. Additionally, it supports deeper learning by providing prompting suggestions after each response to help users formulate thoughtful follow-up questions.

CS50.ai, developed by Harvard University, is a personal AI assistant for students in Harvard's introductory computer science course, CS50 (Liu et al. 2024). Launched in the summer of 2023, CS50.ai utilizes retrieval-augmented-generation (RAG) technology to provide answers based on course lectures and supports programming languages such as C, Python, SQL, JavaScript, CSS, and HTML. It integrates seamlessly with the Harvard Ed forum and Visual Studio Code Integrated Development Environment (IDE) for real-time student engagement. CS50.ai is free for the educational community and has been well received by Harvard students, leading to a reduction in human tutors.

Since the GPT-4 model is not free, both LLM tutors employ usage throttling to limit interaction frequency. Specifically, Khanmigo uses an "AI battery", indicating consumption level, to limit daily message interactions, while CS50.ai allocates "hearts" that deplete and regenerate over time, controlling the frequency of student inquiries. Regarding response accuracy, Khanmigo displays disclaimers to acknowledge potential errors, while CS50.ai uses RAG technology to reduce hallucinations.

LLM Tutor Functions Corresponding to the Levels in the Bloom's Taxonomy

When evaluating the two LLM tutors for the introductory programming course, we identified four core functions based on learning objectives aligned with the six levels of Bloom's Taxonomy. These functions include Concept Comprehension, Debugging, Quiz Preparation, and Program Development. Each function corresponds to different cognitive levels within the taxonomy: Concept Comprehension addresses Understanding and Remembering, Debugging involves Applying and Analyzing, Quiz Preparation targets Evaluating, and Program Development corresponds to Creating (Table 2). This categorization ensures that our investigation covers a comprehensive range of cognitive processes, providing insights into how LLM tutors can support students through the various stages of learning and mastery in computing education.

Research Design

Focus Groups by LLM Tutors and the Functions

We use the focus group method to explore how students evaluate LLM tutors in support of various learning objectives for the introductory programming course. The method is particularly well-suited to the emerging nature of LLM applications in computing education, where the rapidly evolving interactions between students and these technologies require a flexible and exploratory approach. We designed four focus groups, each tailored to assess the distinct functions of LLM tutors in relation to specific learning tasks about programming.

We invited all students enrolled in the introductory programming course to participate in the focus groups to capture diverse perspectives and experiences with LLM tutors. Additionally, we obtained Institutional Review Board approval before conducting the study, ensuring participant confidentiality, informed consent, and adherence to ethical guidelines throughout the research process.

	Focus Group 1	**Focus Group 2**	**Focus Group 3**	**Focus Group 4**
Function	Concept Comprehension	Debugging	Program Development	Quiz Preparation
Learning Objectives Based on Bloom's Taxonomy	*Remember* essential programming concepts *Understand* entire programs or specific code snippets	*Apply* standard debugging techniques to fix a code issue *Analyze* whether the code meets specified requirements *Evaluate* the code based on established coding standards and performance criteria	*Create* new programs or combine existing ones to address a problem	*Remember* and *understand* loop concepts *Apply, analyze*, and *evaluate* knowledge of loops to debug code *Create* a program using loops
Programming Tasks	Explain dictionary	Detect and resolve errors in loops	Develop programs with the use of loops	Prepare for a quiz on loops that includes concept comprehension, debugging, and writing a program
Students (N)	11	16	**21**	12

Table 2. Focus Group Design by LLM Tutors and Functions

Focus Group Procedures

Before participating in the focus group activities, all participants were required to engage with the two LLM tutors, Khanmigo and CS50.ai, through a set of predefined exercises. These tasks are designed to familiarize participants with the interface of each LLM tutor. The exercises were structured to cover similar content across both tutors to ensure a comparable experience. In addition, all participants were required to complete a survey designed to capture their socio-economic background, learning needs, expectations, and perceptions of LLM tutoring systems. This survey helped us assess the students' contextual learning needs and provided a baseline for assessing changes in their perceptions and understanding following interactions with the AI tutors.

During the focus groups, a pre-test was administered to assess the participants' baseline knowledge of the programming topics to be covered during the focus group. This test helped gauge the initial competence of each participant and provided data for comparison with post-test results to measure learning outcomes.

We randomly assigned the sequence in which participants interacted with the two LLM tutors to avoid any sequence bias in the responses. After the pre-test, each participant interacted with the first assigned LLM tutor to complete a specific task. Upon completing a specific task, participants were invited to evaluate the LLM tutor based on their interaction experience. This evaluation was captured through an online survey that included both quantitative questions and open-ended comment questions. Following a similar format, participants engaged with the second AI tutor for a specific task. Participants completed the same evaluation survey to provide feedback.

Once participants had interacted with both LLM tutors, they were asked to compare them directly. This comparison focused on several aspects, such as relative advantages and disadvantages and overall preference between the two systems.

At the conclusion of the individual tasks, a post-test with a similar level of difficulty to the pre-test was administered. This test measured the learning gains attributed to the two LLM tutors and helped assess whether interacting with LLM tutors enhanced learning outcomes.

Following the completion of the individual tasks, participants engaged in a group discussion moderated by a researcher. This discussion allowed participants to share their experiences and insights from using the LLM tutors. The moderator guided the discussion with prompts related to the strengths and weaknesses of each AI tutor, the perceived impact on their learning, and the comparison between LLM tutors and human tutors. The group discussion data is invaluable for understanding the collective user experience and for gathering more nuanced feedback that might not be fully captured through individual evaluations.

Data Collection Method

We collected data from multiple sources. Specifically, we collected survey data with both quantitative responses and qualitative comments to capture students' contextual needs prior to their interaction with the LLM tutors. Throughout the focus group sessions, several methods were employed to collect comprehensive data: survey responses were gathered to track participants' interaction experiences and preferences, chat histories were logged to analyze interactions with the LLM tutors, and audio recordings of group discussions were made to capture dynamic exchanges and collective insights.

Additionally, archival data including demographic information, class performance records, and TA session attendance were collected to provide a deeper context for understanding the participants' academic environments and background influences.

Analysis and Findings

We analyzed the qualitative data from open-ended survey responses, LLM tutor chat logs, and group discussions through coding and thematic analysis to uncover the nuances of interactions between students and LLM tutors. Additionally, we employed natural language processing (NLP) techniques to parse these interactions and identify patterns. To elaborate, we used topic modeling to uncover prevalent themes in student interactions, sentiment analysis to gauge emotional responses, and keyword-based extraction to identify key issues in student feedback based on their experience with the LLM tutors. These methods provided a comprehensive view of how students engage with LLM tutoring, allowing us to assess both the cognitive and emotional aspects of their learning experiences effectively.

Demographics of Participating Students

Our study involved a diverse group of 60 students, providing insight into their backgrounds and the challenges they face which may affect their interaction with learning technologies. Of these students, 40.7% were male, and 59.3% were female. Additionally, 61% of the students were eligible for Pell grants. The racial composition of the sample included 54% African American students, 27.1% Asian students, 1.7% American Indian students, and 8.5% White students. Moreover, 18.6% of the students identified as Hispanic. Notably, among the sampled students, 37.1% are non-native English speakers navigating academic challenges in a second language. Furthermore, 63.9% are first-generation college students, possibly facing unique educational and socio-economic challenges.

Employment is a significant factor for our participants: 14 students work full-time, and 28 hold part-time jobs. These employment commitments indicate that nearly 70% of the participants are balancing their studies with work. Additionally, 24 students have responsibilities for caring for family members, adding another layer of complexity to their time management and daily stress. The employment and caregiving responsibility might affect their ability and energy for academic engagements.

These demographic characteristics suggest the varying needs and challenges these students face, which are likely to influence how they interact with and benefit from LLM tutors.

Analysis of LLM Tutors' Evaluations by Learning Objectives

We analyzed students' responses to open-ended questions and their feedback in group discussions, and summarized our findings in Table 3, which details students' likes and dislikes by functions and LLM tutors. Overall, students value the non-judgmental nature and access to the two AI tutors—Khanmigo and CS50.ai. Despite this, dissatisfaction arises with both tutors due to a lack of code visualization, insufficient customization to individual needs, and the absence of integration with IDEs for coding tasks.

Function	LLM Tutor	Likes	Dislikes
Concept Comprehension	Khanmigo	Breaks down problem Offers prompting suggestions Descriptive answers with relatable examples	Indirect responses and pushes to building block concepts Inefficient process
	CS50.ai	Direct and precise answers Clarified confusion Response personalized for coding	Not tailored to users' comprehension styles Wordy explanation
Debugging	Khanmigo	Prompts to consider building blocks concepts Iterative problem-solving process	Response too general Inefficient process Does not answer questions directly
	CS50.ai	Identifies error directly Suggests solutions Guides debugging with questions	Does not provide direct solutions Wordy and long
Quiz Preparation	Khanmigo	Tutor-guided prompting	Unhelpful hints
	CS50.ai	Direct and less Socratic style of guidance	Not tailored to users' stage of learning
Program Development	Khanmigo	Break down problems Discover code issue without asking Tutor-guided prompts instead of user- steered conversations	Does not provide solutions Tutor-guided prompts constrain user-steered conversations
	CS50.ai	Specific guidance for programming Direct answers	Ineffective guidance to get started
Generic	Khanmigo	Beginner friendly Easy-to-understand language Human-like conversation Expresses empathy Not judgmental Persona choices	UI distracting to use Does not support code editing Fake empathy Too many persona choices
	CS50.ai	Simple UI Supports code editing Not judgmental	Not beginner friendly Assumes prior knowledge User throttling Bot-like conversation

Table 3. LLM Tutors Comparison by Functions

Individual differences are prominent, revealing that each student's preferences can sometimes be contradictory within a single tool or function. For instance, while some students appreciate Khanmigo's empathetic approach in concept comprehension, others find it less effective for their learning needs. Similarly, in program development, some students benefit from Khanmigo's detailed guidance, while those with more experience view it as a hindrance to their learning pace.

When comparing the two LLM tutors, Khamigo generally receives more favorable feedback than CS50.ai. This preference could be attributed to Khanmigo's beginner-friendly approach, as it serves a wide range of users from pre-K to college students at Khan Academy. In contrast, CS50.ai, offered at Harvard, targets all students, including non-majors, who are typically better prepared and more motivated. Students particularly like Khamigo's prompting suggestion feature, which helps them formulate questions when they are unsure what to ask. They also value how Khamigo breaks down complex problems into subproblems, guiding them through each step of the problem-solving process.

Analysis of Chat Logs for LLM Tutors

We analyzed the chat logs between students and the two LLM tutors using text mining techniques. After discarding files with misformatting or irrelevant content, we processed 76 chat logs. In general, we observed that students asked more questions when interacting with Khanmigo than with CS50.ai across all four functions (Figure 1). On average, students responded more than once to prompts from the LLM tutors, particularly in the program development function (Figure 2).

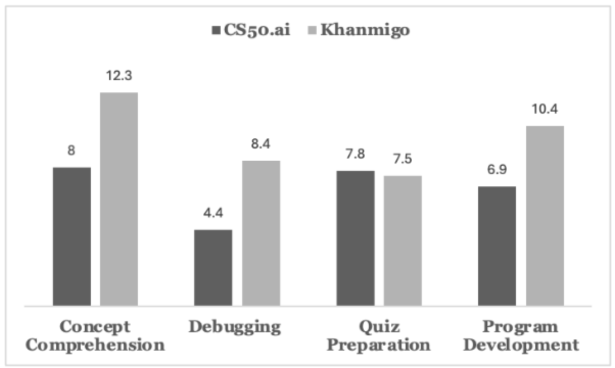

Figure 1. Average Number of Questions asked by Students by Functions and LMM Tutors

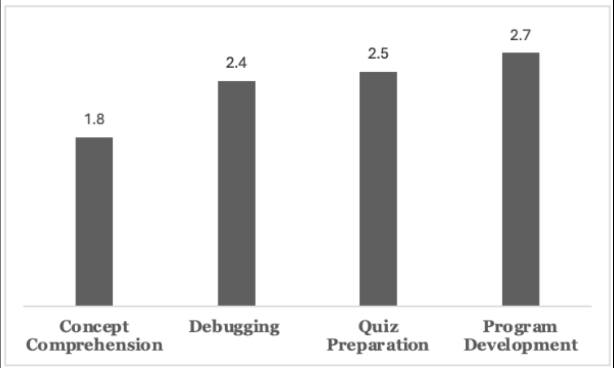

Figure 2. Average Number of Responses to LLM Tutors' Prompts by Functions

Importantly, we observed heterogeneous interaction styles across students with the LLM tutors:

Socratic approach. Khanmigo uses a Socratic method, asking questions to guide students, which while appreciated by some, causes others to disengage if they prefer to receive direct answers. For example, Khanmigo often prompts students to critically assess their code with questions like, "*The code should print 'Prime' at the end no matter what, right? So, why might it not be printing anything at all?*" This method encourages deep reflection but can frustrate those who seek quicker solutions. In contrast, CS50.ai tends to offer more direct answers and employs a less Socratic style of guidance. CS50.ai frequently identifies specific issues, stating, "*It seems like you're trying to print all prime numbers up to a certain number. However, there are a few issues with your code,*" which provides students with immediate corrections and clear directions for improvement.

Prompting variation. CS50.ai's interactions vary significantly depending on how students present their queries. Those who provide detailed prompts receive more tailored responses, whereas those who submit vague requests often find the guidance insufficient. For example, when a student asked, "how do i loop through each character in the string," CS50.ai provided a detailed and practical response:

> "*In C, you can loop through each character in a string using a for loop. Here's a general idea:*
> *for (int i = 0; str[i] != '\0'; i++)*
> *{*
> *// Access each character using str[i]*
> *}*
> *In this loop, str[i] gives you the character at position i in the string. The loop continues until it encounters the null character ('\0'), which marks the end of the string.*"

Conversely, when a student asked, "*How do I start my loop?*" CS50.ai provided a code snippet in C/C++ without explaining how to adapt it to the specific student problem of counting lowercase letters in Python:

> *"Here's a small snippet to get you started:*
> *for (int i = 0, n = strlen(input_string); i < n; i++)*
> *{*
> *// Your code here*
> *}."*

Engagement. Khanmigo boosts student engagement by using relatable examples. For instance, while CS50.ai explained a Python dictionary in technical terms: "*Each item in a dictionary consists of two elements: a key and a value. The key is used to identify the item and the value is the data that is associated with the key,*" Khanmigo offered a more vivid explanation: "*A dictionary in Python, you say? It's a bit like a magical book! Imagine a book where you can look up a word (we call this the 'key') and find a meaning or a value (we call this the 'value'). In Python, a dictionary works the same way! You use a key to get a value. Can you think of an example where you might use a dictionary in Python?*" This type of analogy, associating Python dictionaries with everyday objects like books, not only made the concept more accessible but also engaged students by connecting to familiar experiences. However, we found that students' reactions to these examples varied, as some found certain content less compelling.

Tutor-guided prompting. We noticed some students struggled with formulating questions for debugging and program development with CS50.ai. In contrast, Khanmigo not only answers queries but also suggests follow-up questions, which has received mixed reactions. Analysis of the chat log showed that over 63% of students utilized tutor-suggested prompts. For example, a student asked Khanmigo "*How to write a program that counts the number of lowercase letters in a string in Python?*" Khanmigo responded by suggesting prompts to help break down the problem into manageable steps, such as "*how to identify if a character is lowercase,*" and "*how to use islower() in my program.*" These guided prompts guided the student to explore the problem further and eventually complete the program. Notably, while some students followed these prompts without adding their own input, others resisted this guided approach, preferring to steer the conversation themselves.

Academic performance association. We further scrutinized the chat logs of students with varying academic performances, examining their interaction styles in relation to their midterm exam grades, which were completed before their participation in the focus groups. Our observation revealed distinct interaction styles between students who performed well and those who did not in the introductory programming class (Table 4). Specifically, high performers are generally more responsive to the AI tutors' guidance and actively apply corrections, showing a keenness to explore advanced features. Low performers, while often receptive to guidance, sometimes struggle with basic concepts and require repeated clarifications, indicating varied paths of engagement and learning. The observations collectively indicate the diverse pathways of evaluation and interaction among students, highlighting the need for LLM tutors that can adapt flexibly to different educational needs and learning styles.

	Khanmigo	CS50.ai
High Performer (top 30%)	● Actively follow the tutor's prompts and instructions, demonstrating a strong understanding of basic concepts and exploring advanced features. ● Proactively address potential issues and complexities, and respond positively to corrections, immediately attempting to rectify mistakes after receiving feedback.	● Show willingness to learn by asking questions built on previous information, participating in exercises, and applying newly gained knowledge in practical examples. ● Persistently engage in the face of new challenges, follow instructions closely, and frequently seek confirmation to ensure correctness. ● Respond positively to motivational feedback.
Low Performers (bottom 30%)	● Consistently seek and apply syntax clarification. ● Show receptiveness to guidance, follow the laid-out learning path, and request direct clarification when needed. ● Actively refine approaches based on tutor suggestions, build on incremental information, and apply concepts. ● Show willingness to learn, despite basic programming errors.	● Demonstrate a clarification-based approach to learning by asking a series of questions. ● Follow the tutor's guidance diligently to understand syntax and semantics, asking specific questions for precise understanding. ● Do not challenge the information even when prompted to think critically.

Table 4. Interactions with LLM Tutors by High and Low Performers

Summary of Focus Group Insights

The analysis of student feedback from the focus groups reveals significant diversity in how students engage with different LLM tutoring styles in Python programming courses. Preferences split between the Socratic, probing style of Khanmigo, which encourages exploration of foundational principles, and the direct, solution-oriented approach of CS50.ai that emphasizes straightforward explanations across various learning objectives. We elaborate on the differences in preferences to derive design and evaluation implications for LLM tutors.

Preferences for tutoring styles based on learning objectives. (1) *Understanding Concepts*: Students who value deep conceptual understanding tend to favor Khanmigo's method, which promotes critical thinking, while those focused on immediate clarity and precision prefer CS50.ai's direct answers. (2) *Debugging Code*: Preferences vary between direct error identification and solutions offered by CS50.ai and the deeper and iterative problem-solving engagement prompted by Khanmigo. (3) *Preparing for Quizzes*: CS50.ai's domain-specific knowledge is preferred for efficient quiz preparation, contrasting with Khanmigo's less focused Socratic interactions. (4) *Developing Programs*: CS50.ai is favored for its specific, technical guidance in practical programming tasks, highlighting the need for precise, actionable feedback.

Need for seamless and comprehensive linking to course content. Feedback from focus groups indicates a desire among students for LLM tutors to not only assist with immediate queries and problems but also connect these discussions to broader course materials such as lectures, assignments, videos, and archives of in-person tutoring sessions. This capability would provide students with a contextual learning experience, deepening their understanding and reinforcing learning through multiple touchpoints. Also, students would like the tutor to restrict its suggestions only to concepts that have been covered in their classes rather than more advanced concepts that have not yet been covered.

User throttling and session control. CS50.ai employs a user throttling mechanism where students are allocated a finite number of "hearts" that are consumed with each prompt. This feature frustrated many students, particularly when hearts were spent due to misunderstandings or misinterpretations by the tutor. The depletion of hearts because of perceived tutor errors led to dissatisfaction, as it restricted their ability to engage fully with the learning material. An emerging insight suggests that the LLM tutors could be improved by allowing students to earn points if they perform well competitively, enhancing motivation and engagement by rewarding success, rather than solely reducing the amount of use.

Operational performance concerns. Several students, across the focus groups, expressed dissatisfaction with the operational performance of both tutoring systems, particularly in terms of response times. Delays and slow interactions were noted as significant hindrances to learning, impacting the overall effectiveness and user experience of the tutoring sessions.

Challenges with user interface and lack of IDE integration. Students reported challenges with the user interface, specifically the lack of integration between the tutors and the IDE used for programming. The absence of an integrated IDE meant that students found it tedious to transfer content between the tutoring environment and the IDE, impacting their efficiency and overall learning experience.

Variability in student interaction styles. Our analysis of the chat logs revealed the variability in student interaction styles—from those seeking quick, specific answers to those engaging in extended, concept-exploring discussions—highlights the need for tutors to adapt responses based on the depth and nature of prompts. This variability, from the initial prompt to the chain of interaction, indicates that effective LLM tutors must be capable of both direct problem-solving and facilitating deeper inquiry.

Role of empathy in tutor interactions. Khanmigo's attempts to offer emotional support and encouragement, such as affirmations and motivational comments, were appreciated by some students for enhancing confidence. However, others found these attempts at empathy less credible, especially when the tutor praised efforts that were off the mark. To enhance credibility, expressed empathy should be reserved for situations where the student is on the right track and such encouragement can genuinely boost self-efficacy. Moreover, offering multilingual support could enhance the usability for non-English speakers, making all students feel understood and supported regardless of their native language. In contrast, CS50.ai's matter-of-fact style, devoid of expressed empathy, was not seen as a drawback by many students, who valued its precision and to-the-point interactions, particularly when efficiency was prioritized over a detailed exploration of challenging concepts.

Implications for the Design and Evaluation of LLM Tutors

We interpret the implications of the focus groups' insights for the design and evaluation of LLM tutors to cost-effectively personalize the interactions at scale—whereby they fulfill the diverse learning styles and preferences of students and cater to the nature of learning objectives at different levels of complexity in the learning process. In doing so, we also surface how we contribute to the current understanding in the literature on AI Ed-Tech and that on IS design and evaluation based on user preferences.

First, contributing to the insight that aligning the design and evaluation of IS to requirements has been long recognized as necessary for IS success (Ramesh et al. 2010), we surface two alignment considerations for LLM tutors.

Aligning tutoring methods to learning objectives. Educational technologies, especially those involving LLM tutors, to enable student learning must incorporate adaptive tutoring methods that are responsive not only to varying student preferences for interaction styles but also to different learning objectives (Newman et al., 2013; Xie et al. 2019). The underlying rationale is in sync with the guidelines in the IS literature to dynamically align the technology with the varying requirements of the user and the task (Benbya et al. 2023; Goodhue and Thompson 1995; Granić 2022) and to elaborate on the contextual differences to understand the design requirements and the nature of IS use that is effective (e.g., Hong et al. 2014; Burton-Jones and Volkoff 2017).

We add to this understanding by surfacing how the nature of adaptive tutoring needs to vary across levels of Bloom's Taxonomy. For example, direct responses may be more effective for objectives at lower levels of Bloom's Taxonomy, such as understanding basic concepts, while a Socratic, inquiry-based method may be better suited for higher levels, such as applying concepts to diagnose and remedy errors, and developing complex programming projects, especially when there are multiple pathways to developing a solution.

Aligning tutor interaction styles to student learning styles. Tutors need to be responsive to the diversity in student interaction styles, from those who prefer concise, technical inquiries to those who engage in broader, conceptual discussions. This requires sophisticated natural language processing capabilities and an understanding of different educational contexts. Evaluations should measure the tutor's ability to adequately respond to varied types of inquiries and adapt teaching strategies accordingly.

Second, integration with complementary resources is a key requirement to develop novel ways to create value with AI (Brynjolfsson et al. 2019). We identify integration with two complementary resources as focal design and evaluation considerations for LLM tutors.

Integrating with development tools. Integration of LLM tutors with essential programming tools, particularly Integrated Development Environments is crucial. This integration should provide a seamless experience where students can write, test, and debug code within the same environment they are receiving tutoring, thus reducing the need to switch contexts and improving learning efficiency. Such a seamless experience reflects the IS design principle of integration and interoperability, ensuring that LLM tutors work effectively within the existing educational infrastructure. Evaluations should assess how well these integrations minimize workflow disruption and enhance user satisfaction.

Integrating with course content through retrieval-augmented generation. To achieve seamless integration with the variety of course content, LLM tutors should employ advanced techniques such as RAG. This approach allows the tutor to dynamically retrieve relevant information from a vast database of course content while generating responses. By doing so, the tutor can provide not only immediate answers but also contextually relevant resources and explanations, aligning with user-centered design principles and directly linking tutoring interactions to specific parts of the course material.

Incorporating RAG and other similar techniques into LLM tutors could greatly enhance the 'stickiness' of the interactions—meaning that students are more likely to engage frequently and deeply with the tutor, leading to better retention and understanding. This seamless integration can ensure that every interaction with the tutor is both a point of learning and a gateway to further exploration of the course content.

For designers, this underscores the importance of creating LLM tutors that can intelligently access and integrate diverse educational resources. Evaluation metrics should therefore not only assess the accuracy and relevance of the responses provided by the tutor but also measure how effectively these responses are integrated with the course content. This will ensure that the tutor is truly augmenting the learning experience by making all relevant materials readily accessible and usable within the learning context. Further, this will also restrict the guidance provided by the tutor to the concepts that have been covered in class sessions thus far rather than introducing concepts that the student has not yet been exposed to.

Third, in dynamic contexts such as learning where exogenous and endogenous factors evolve, a system needs to be able to adapt with precision to meet shifting needs (Malgonde et al. 2022). A key implication is that LLM tutor needs to adapt their interventions, engagement, and controls with precision to fulfill the evolving cognitive and affective needs of students and managing costs while rendering effective learning support in an institutional context.

Adapting push-pull interventions to student learning needs. By analyzing student feedback, such as upvotes and downvotes on responses, LLM tutors can calibrate their support to meet the specific learning needs of students as these evolve. In alignment with the IS design principle of adapting a solution to evolving user requirements, LLM tutors need to dynamically calibrate push (proactive) and pull (reactive) support to the students' learning needs, and thereby promote their attainment of learning objectives.

Moreover, an effective strategy to enhance the personalization of LLM tutor interactions can involve calibrating the push-pull interventions based on utilization of knowledge graphs. These graphs can be used to map out the entire body of knowledge relevant to a course or subject area. By tracking a student's progress and attention allocated over time to different concepts and applications, the tutor knows where a student is in their learning journey and can identify knowledge gaps and areas of strength. As such, the knowledge graph for each student should dynamically update as the student interacts with the tutor and the course material, reflecting their evolving understanding and focus areas.

This continuous updating allows the LLM tutor to deliver highly specific interventions—both push and pull—tailored to the immediate needs of the student over time. For instance, if a student consistently struggles with a particular concept, the tutor can proactively offer additional resources or simplified explanations before the student becomes too frustrated. Such a process can prevent a student from slipping into undesirable states that can lead to erosion of confidence, self-esteem, and resignation.

By leveraging knowledge graphs, tutors can make their interventions not only more specific but also more engaging to the student's current context and learning phase. This specificity to the student and their journey in a course enhances the 'stickiness' of interactions, encouraging deeper and more frequent engagement with the tutor. Students are more likely to perceive the tutor as a helpful and integral part of their learning process, which fosters a positive learning environment and can boost educational outcomes.

The design of LLM tutors that utilize knowledge graphs requires sophisticated algorithms capable of not only tracking and analyzing student interactions but also integrating this data into a coherent framework that supports adaptive learning strategies. Evaluation of such systems should focus on the effectiveness of the knowledge graph in enhancing personalization. Metrics should assess how well the tutor adapts to students' learning progress and the impacts of personalized interventions on learning outcomes.

Adapting emotional engagement to student affective needs. Tutors should be designed to recognize and respond to both the emotional and cognitive needs of students. While some students appreciate and benefit from emotional encouragement, others may find it lacks credibility, especially if it does not align with their actual performance. Tutors should have the capability to adapt their level of empathetic feedback based on student responses and preferences. Evaluations should consider student perceptions of tutor empathy and its impact on learning outcomes.

Adapting operational controls to achieve cost-effective LLM tutoring. The operational performance of tutoring systems, including response times and system reliability, must be a key focus in design and evaluation. Slow or unreliable connectivity, in general and at peak times such as during exam and assignment times, can trigger frustration, thereby significantly hindering learning and reducing user satisfaction. Continuous performance optimization and rigorous testing, alongside changes in technology capability and compute costs, should be conducted to ensure systems meet the expectations of students and other key stakeholders in computing educational settings at different types of universities.

Design considerations on operational control mechanisms should also include user-controlled session management to avoid frustration associated with rigid system-imposed rigid controls like the "hearts" system in CS50.ai. Tutors should provide flexibility in how interaction limits are implemented, possibly by allowing students to earn additional interaction opportunities through engagement or performance. Additionally, it's important to evaluate the impact of these mechanisms on student learning and system usability. Implementing such controls serves dual purposes: it provides a cooling-off period to promote thoughtful inquiry, potentially enhancing learning, and they help manage the operational costs of running LLMs. However, while cost management is necessary, the educational effectiveness of the cooling-off period still requires further exploration.

In sum, these implications enhance our understanding of how to effectively design and evaluate LLM tutors to meet the diverse needs of students. The ability to automate and personalize interactions through RAG enables LLM tutors to deliver tailored, real-time responses, reducing the need for human intervention while maintaining high-quality support. Additionally, the 24/7 availability increases accessibility, especially for students from under-resourced backgrounds. Exemplary LLM tutors demonstrate affordability, though the evolving costs of popular LLMs require adaptable design strategies to sustain cost-effectiveness.

Future Research Avenues

Future research can develop LLM functionalities and assessment approaches based on the design and evaluation implications that we have identified based on the focus groups. Such LLM functionalities can be further assessed in future focus groups and also assessed in terms of their effects by using pre- and post-tests, lab experiments, or randomized control trial research designs. As LLM technology continues to evolve, it will be essential to reassess the emerging capabilities like multimodal interactions and evaluate students' learning outcomes when working with LLM tutors across different learning settings (e.g., programming languages, course levels, types of institutions). Moreover, future research will need to investigate how to achieve classroom integration of LLMs. Additionally, it is important to consider the long-term implications of LLM integration for computing education on the meta-cognition of learners: what aspects are supported by the tutor and what new demands are imposed by it. Moreover, it is important to consider the requirements of teachers and administrators, to understand the tensions in these requirements, and to develop approaches to manage the tensions. In doing so, we can develop a much more complete understanding of how LLMs tutors can be designed and deployed in a personalized, scalable, and cost-effective manner and used responsibly by students, teachers, teaching assistants, and administrators.

Future research can track students' interaction with LLM tutors and employ NLP and process mining techniques to analyze students' transitions in learning state, progression in knowledge acquisition, and changes in sentiment over time. Specifically, NLP will allow us to identify prevalent themes, sentiments, and interaction patterns with LLM tutors. Process mining will further enable a thorough examination of learning behavior and engagement. Collectively, these analyses will reveal the interaction nuances between students and LLM tutors, offering a granular view of their learning processes and experiences.

Since students have (free) access to resources like ChatGPT 3.5 that provide correct solutions to commonly used assignments and exam questions, it is important for the developers of LLM Tutors to provide compelling reasons for students to use these tutors rather than ChatGPT 3.5 or other resources. This may require the tutor to take a proactive role in the student's learning journey, say for example, engaging the student when they have not yet commenced their work on an upcoming assignment. Also, instructors need to make significant changes in their evaluations of student performance, by focusing on a student's understanding and mastery of knowledge elements (in a knowledge graph) rather than typical assessment methods like tests and coding assignments. Such an assessment is feasible only when the LLM tutor can continually monitor and assess student interactions.

Building on the framework of Bloom's taxonomy, future studies can utilize the four types of knowledge (i.e., factual, conceptual, procedural, and metacognitive knowledge) to tailor the development of computational thinking skills for diverse learning needs. Employing this framework could advance our understanding of how knowledge dimensions integrate with cognitive processes in educational settings. This approach will enrich theoretical models of learning by demonstrating interactions and dependencies between factual, conceptual, procedural, and metacognitive knowledge within various learning environments.

The need to develop policies to address ethical and academic integrity considerations that are crucial for maintaining the credibility and effectiveness of the educational experiences is heighted with increasing access to LLM tutors. Future research should develop policies to safeguard academic rigor and standards that ensure that assessment of student performance reflects genuine knowledge and effort. Additionally, such policies should be designed to foster an environment of honesty, trust, and fairness, which is essential for meaningful learning experiences.

Acknowledgements

This research was funded by Axim Collaborative (Grant #S6304/PO#1048777). Balasubramaniam Ramesh's research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911F-23-2-024. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Adams, N. (2015). Bloom's taxonomy of cognitive learning objectives. *Journal of the Medical Library Association, 103*(3), 152-153. https://doi.org/10.3163/1536-5050.103.3.010

Benbya, H., Leidner, D., & Preston, D. (2019). Research curation on information systems alignment. *MIS Quarterly*, www.misqresearchcurations.org/blog/2019/3/14/information-systems-alignment.

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E.A. (2023). Programming is hard-or at least it used to be: Educational opportunities and challenges of AI code generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education, Vol. 1,* 500-506. https://doi.org/10.1145/3545945.3569759

Bettinger, E., & Baker, R. (2011). The effects of student coaching in college: An evaluation of a randomized experiment in student mentoring. *NBER* Working Paper No. 16881. https://doi.org/10.3386/w16881

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1964). *Taxonomy of Educational Objectives* (Vol. 2). New York: Longmans, Green.

Brynjolfsson, E., Rock, D., & Syverson, C. (2019). *Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics*, 23-60, University of Chicago Press.

Burton-Jones, A., & Volkoff, O. (2017). How can we develop contextualized theories of effective use? A demonstration in the context of community-care electronic health records. *Information Systems Research, 28*(3), 468-489. https://doi.org/10.1287/isre.2017.0702

Denny, P., Prather, J., Becker, B.A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B.N., Santos, E.A. & Sarsa, S. (2024). Computing education in the era of generative AI. *Communications of the ACM, 67*(2), 56-67. https://doi.org/10.1145/3624720

Extance, A. (2023). ChatGPT has entered the classroom: How LLMs could transform education. *Nature 623*(7987), 474-477. https://doi.org/10.1038/d41586-023-03507-3

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI codex on introductory programming. *Proceedings of the 24th Australasian Computing Education Conference,* 10-19. https://doi.org/10.1145/3511861.3511863

Goodhue, D.L. & Thompson, R.L. (1995). Task-technology fit and individual performance. *MIS Quarterly, 19*(2), 213-236. https://doi.org/10.2307/249689

Goel, A. K., & Polepeddi, L. (2018). Jill Watson: A virtual teaching assistant for online education. *Learning Engineering for Online Education*, 120-143, Routledge.

Granić, A. (2022). Educational technology adoption: A systematic review. *Education and Information Technologies, 27*(7), 9725-9744. https://doi.org/10.1007/s10639-022-11053-0

Gumbel, A. (2020). *Won't Lose This Dream: How an Upstart Urban University Rewrote the Rules of a Broken System*. The New Press.

Hamilton, S., & Chervany, N. L. (1981). Evaluating information system effectiveness: Comparing evaluation approaches. *MIS Quarterly, 5*(3), 55-69. https://doi.org/10.2307/249291

Hong, W., Chan, F.K., Thong, J. Y., Chasalow, L.C., & Dhillon, G. (2014). A framework and guidelines for context-specific theorizing in information systems research. *Information Systems Research, 25*(1), 111-136. https://doi.org/10.1287/isre.2013.0501

Hsieh, J.P.A., Rai, A., & Keil, M. (2008). Understanding digital inequality: Comparing continued use behavioral models of the socio-economically advantaged and disadvantaged. *MIS Quarterly, 32*(1), 97-126. https://doi.org/10.2307/25148830

Klopfer, E., Reich, J., Abelson, H. & Breazeal, C. (2024). Generative AI and K-12 education: An MIT perspective. https://mit-genai.pubpub.org/pub/4k9msp17/release/1?readingCollection=0e231e9c

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into Practice*, *41*(4), 212-218. https://doi.org/10.1207/s15430421tip4104_2

Lahtinen, E. (2007). A categorization of novice programmers: A cluster analysis study. *Proceedings of the Annual Workshop of Programming Interest Group*. Joensuu, Finland. Vol. 16, 32-41.

Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., & Malan, D. J. (2024). Teaching CS50 with AI: Leveraging generative AI in computer science education. *Proceedings of the ACM Technical Symposium on Computer Science Education, 1*, 750-756. https://doi.org/10.1145/3626252.3630938

Luckin, R. & Holmes, W. (2016). *Intelligence Unleashed: An Argument for AI in Education*. Pearson.

Ma, W., Adesope, O.O., Nesbit, J.C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *J. of Educational Psychology, 106*(4), 901-918. http://dx.doi.org/10.1037/a0037123

Malgonde, O.S., Zhang, H., Padmanabhan, B., & Limayem, M. (2022). Managing digital platforms with robust multi-sided recommender systems. *Journal of Management Information Systems, 39*(4), 938-968. https://doi.org/10.1080/07421222.2022.2127440

Newman, A., Bryant, G., Stokes, P., & Squeo, T. (2013). Learning to Adapt: Understanding the Adaptive Learning Supplier Landscape. *Tyton Partners White Paper*.

Oliver, D., Dobele, A., Greber, M., & Roberts, T. (2004). This course has a bloom rating of 3.9. *Proceedings of the Sixth Australasian Conference on Computing Education*, 227–231. Dunedin, New Zealand.

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of Github Copilot's code contributions. *IEEE Symposium on Security and Privacy,* 754-768. https://doi.org/ 10.1109/SP46214.2022.9833571

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, *20*(5), 449-480. https://doi.org/10.1109/MS.2008.1

Reich, J., (2020). *Failure to Disrupt: Why Technology Alone Can't Transform Education*. Harvard University Press.

Reich, J. (2022). Learning analytics and learning at scale. *Lang, C., Siemens, G. Wise, A.F., Gašević, D., Merceron, A. (eds.) Handbook of Learning Analytics, 2nd ed. Vancouver, Canada: SOLAR*.

Rodriguez-Segura, D. (2022). EdTech in developing countries: A review of the evidence. *The World Bank Research Observer,* 37(2), 171-203. https://doi.org/10.1093/wbro/lkab011

Shneider, E., & Gladkikh, O. (2006). Designing questioning strategies for information technology courses. *Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications,* 243-248.

Starr, C. W., Manaris, B., & Stalvey, R. H. (2008). Bloom's taxonomy revisited: Specifying assessable learning objectives in computer science. *ACM SIGCSE Bulletin, 40*(1), 261-265. https://doi.org/10.1145/1352322.1352227

Steele, C. M. (1997). A threat in the air: How stereotypes shape intellectual identity and performance. *American Psychologist, 52*(6), 613-629. https://doi.org/10.1037/0003-066X.52.6.613

Tinto, V. (2012). *Leaving college: Rethinking the causes and cures of student attrition*. U. of Chicago Press.

Warschauer, M., & Matuchniak, T. (2010). New technology and digital worlds: Analyzing evidence of equity in access, use, and outcomes. *Review of Research in Education, 34*(1), 179-225. https://doi.org/10.3102/0091732X09349791

Xie, H., Chu, H. C., Hwang, G. J., & Wang, C. C. (2019). Trends and Development in Technology-Enhanced Adaptive/Personalized Learning: A Systematic Review of Journal Publications from 2007 to 2017. *Computers and Education, 140,* 103599. https://doi.org/10.1016/j.compedu.2019.103599

Xu, J., Benbasat, I., & Cenfetelli, R. T. (2013). Integrating service quality with system and information quality: An empirical test in the e-service context. *MIS Quarterly, 37*(3), 777-794. https://doi.org/10.25300/misq/2013/37.3.05

Yosso, T. J. (2005). Whose culture has capital? A critical race theory discussion of community cultural wealth. *Race ethnicity and education, 8*(1), 69-91. https://doi.org/10.1080/1361332052000341006

Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education–Where are the educators? *International Journal of Educational Technology in Higher Education, 16*(1), 1-27. https://doi.org/10.1186/s41239-019-0171-0