Benyamin Tabarsi[1], Tahreem Yasir[1], Heidi Reichert[1], Xiaoyi Tian[1], Shiva Gadireddy[1], Clara Dimarco[1], Daniel Briceno[1], and Tiffany Barnes[1]

[1]Department of Computer Science, North Carolina State University

December 02, 2025

Abstract

The variety of help that large language models (LLMs) provide has made them popular among students across fields. Computing education has been particularly affected, as LLMs can handle coding tasks effectively and provide feedback. This has raised hopes for supporting students, while creating concerns about learning and academic integrity. Researchers have responded by developing tools that leverage LLMs' potential while mitigating risks. Despite growing empirical studies on LLM-driven tools, there is no comprehensive study examining these tools in computing education, critical for understanding the future of learning systems. We review LLM-driven, student-facing tools in undergraduate computing education, examining their pedagogical approaches, technical design, evaluation approaches, and educational impacts. We also discuss how these tools impact the broader educational system, including institutional and pedagogical structures. Following PRISMA guidelines, we systematically searched three major libraries, conducted rigorous screening, and analyzed 52 papers. Our findings reveal that prompt engineering and multi-stage pipelines are the dominant technical approaches, with guardrails and prompt chaining also widely adopted. Pedagogically, most systems provide scaffolding, problem-based learning, and direct instruction. For evaluation, researchers commonly relied on student surveys and interaction logs, with limited assessment of long-term learning outcomes. Evidence shows generally positive impacts on student performance, efficiency, and engagement, although significant challenges remain in delivering adequate scaffolding, preventing over-reliance, and supporting the development of meta-cognitive and problem-solving skills. Future work should focus on longitudinal outcomes, systematic evaluation of learning effectiveness, and the integration of established pedagogical frameworks. This review provides researchers, developers, and educators with a roadmap for designing and studying the next generation of AI-enhanced learning tools.

Herald of Advancement, Disruption, or Both: A Systematic Literature Review on Student-Facing LLM Tools in Undergraduate Computing Education

Benyamin Tabarsi[a], Tahreem Yasir[a], Heidi Reichert[a], Xiaoyi Tian[a,*], Shiva Gadireddy[a], Clara DiMarco[a], Daniel Briceno[a], Tiffany Barnes[a]

[a]*Department of Computer Science, North Carolina State University, Raleigh, North Carolina, 27695, USA*

Abstract

The variety of help that large language models (LLMs) provide has made them popular among students across fields. Computing education has been particularly affected, as LLMs can handle coding tasks effectively and provide feedback. This has raised hopes for supporting students, while creating concerns about learning and academic integrity. Researchers have responded by developing tools that leverage LLMs' potential while mitigating risks. Despite growing empirical studies on LLM-driven tools, there is no comprehensive study examining these tools in computing education, critical for understanding the future of learning systems. We review LLM-driven, student-facing tools in undergraduate computing education, examining their pedagogical approaches, technical design, evaluation approaches, and educational impacts. We also discuss how these tools impact the broader educational system, including institutional and pedagogical structures. Following PRISMA guidelines, we systematically searched three major libraries, conducted rigorous screening, and analyzed 52 papers. Our findings reveal that prompt engineering and multi-stage pipelines are the dominant technical approaches, with guardrails and prompt chaining also widely adopted. Pedagogically, most systems provide scaffolding, problem-based learning, and direct instruction. For evaluation, researchers commonly relied on student surveys and interaction logs, with limited assessment of long-term learn-

*Corresponding author
Email address: `xtian9@ncsu.edu` (Xiaoyi Tian)

ing outcomes. Evidence shows generally positive impacts on student performance, efficiency, and engagement, although significant challenges remain in delivering adequate scaffolding, preventing over-reliance, and supporting the development of meta-cognitive and problem-solving skills. Future work should focus on longitudinal outcomes, systematic evaluation of learning effectiveness, and the integration of established pedagogical frameworks. This review provides researchers, developers, and educators with a roadmap for designing and studying the next generation of AI-enhanced learning tools.

Keywords: Systematic literature review, Large Language Models (LLMs), Computing education, GenAI-driven Educational technology

---

1. Introduction

Over the past few decades, Artificial Intelligence (AI) has evolved from rule-based systems to today's generative systems that converse and reason with near-human fluency. Large language models (LLMs), a relatively recent and pivotal advancement in this evolution, have extensively impacted many domains, including education, healthcare, and video games (Denny et al., 2024c; Rodrigues and Teixeira Lopes, 2025; Sweetser, 2024). Computing education is a field with some of the earliest and most active adopters of LLMs Prather et al. (2025); Mahon et al. (2024), largely due to their capabilities to parse natural-language prompts, generate code, and provide feedback (Sadat Shanto et al., 2025; Shuvo et al., 2025; Koutcheme et al., 2024). The impact has been substantial: GPT-4 achieves scores of 99.5% on introductory computer science (CS) exams (Denny et al., 2024c), and AI systems exceed many students' performance in existing assessments (Finnie-Ansley et al., 2023). Not surprisingly, many engineering and CS students now routinely use LLMs to help them learn course materials and complete assignments (Bernabei et al., 2023; Zamfirescu-Pereira et al., 2025).

However, the same potential that attracts learners carries important concerns, such as surface learning, overreliance (Kazemitabaar et al., 2025; Qu et al., 2025), and academic integrity (Richards et al., 2024; Pang and Vahid, 2024). More fundamentally, these tools are increasingly challenging traditional approaches to curriculum design and assessment (Prather et al., 2023; Feng et al., 2025). This has left educators and policy makers struggling with the impacts of LLMs and adopting divergent approaches (Sætra, 2023).

Despite these tensions and uncertainties, the transformation of work from the human-computer frontier to the human-AI frontier makes adapting existing curricula with LLMs essential rather than optional (Walter, 2024), particularly in computing education. As all fields increasingly require professionals to interact with AI to solve problems, the early-adopting field of computing education serves as a crucial testing ground for understanding how students learn with AI systems. Studying LLM integration in computing education is therefore likely to uncover implications for designing future AI-based learning systems across disciplines. However, the rapid development of this field has outpaced systematic efforts to synthesize the current LLM applications, evaluate their effectiveness, and identify key directions for future work.

Several reviews conducted field-level examinations of how AI and LLMs intersect with computing education. Early work by Prather et al. (2023) reviewed 71 papers and conducted international surveys of students and instructors, interviews with educators, and a benchmark of LLM capabilities in computing education datasets, presenting a comprehensive analysis of opportunities and challenges raised by generative AI (GenAI). Álvarez Ariza et al. (2025) conducted a scoping review of 24 empirical studies across K-12 and university settings, analyzing how GenAI integration affects engineering and computing education. Raihan et al. (2024) synthesized 125 studies on LLMs in CS education by educational level, affected sub-disciplines, methodologies, programming languages, and specific LLMs employed, identifying introductory programming courses as a common focus and GPT models as the dominant technology employed. Taking a broader perspective, Tan et al. (2025b) explored 127 papers focused on AI-enabled adaptive learning platforms covering different disciplines and levels, analyzing pedagogical foundations, AI implementations, real-world challenges, and successes; this highlighted personalization's positive impacts on student performance, issues like privacy, and the need for faculty support. While these reviews have mapped the overall landscape of AI and LLMs in computing education, they do not provide a focused image of student-facing LLM tools, supported by empirical evidence.

Some other reviews have focused on programming education or LLMs as coding assistants. In a review of 119 papers, Manorat et al. (2025) investigated the impact of AI on programming courses and categorized applications into four instructional design areas (course design, implementation, assessment, and monitoring) to provide an overview of practical tools for instructors. Cambaz and Zhang (2024) reviewed 21 papers for the primary uses and ethics of AI-driven code-generation models in programming education, finding that instructors use them to create assignments and students use them as virtual tutors. Pirzado et al. (2024) reviewed 72 studies analyzing LLM behavior and performance as coding assistants, concluding that current LLM limitations such as limited debugging capabilities make them still immature for integration as coding assistants. Synthesizing 35 controlled studies, Alanazi et al. (2025) revealed the high potential of AI tools in programming education and the need to update pedagogical strategies to counteract risks such as student over-reliance and a decrease in problem-solving skills. Taken together, these reviews illuminate programming education but do not offer a focused synthesis of LLM-driven tools across computing beyond programming.

Overall, prior work has emphasized (a) broad GenAI/LLM landscapes in education, (b) mixed audiences (students, instructors, institutions), (c) blends of general-purpose LLMs and customized tools which are context-specific or research-tailored, (d) programming education specifically, or (e) multiple grade bands across K-12 and higher education. There is a need to build a focused and holistic picture of student-facing LLM tools in undergraduate computing education: why these tools are created, how they are framed pedagogically and technically, what data are collected to evaluate them, which facets are assessed, and what impacts on learning outcomes, behaviors, and perceptions are supported by empirical user studies.

Our systematic literature review addresses this gap by focusing exclusively on customized, student-facing LLM tools to surface pedagogical and technical adaptations and innovations in the literature. This focus is crucial for several reasons. First, customized LLM tools reveal deliberate pedagogical design decisions and educational adaptations that are obscured in general-purpose LLMs like ChatGPT. We focus on these purpose-built tools to provide insights into how educators and computing education researchers intentionally integrate AI technology to support specific learning objectives. Second, students are comparatively more vulnerable users in education; the design choices in student-facing tools impact the knowledge and skills of future graduates and influence policies and practices for courses and assessments. Third, student-facing LLM tools serve as primary indicators of current and emerging educational changes. Understanding the impact of these purpose-built tools on students allows us to examine the broader implications for education and design pedagogically sound and effective learning tools to help students grow their expertise and ability to solve complex problems, even with the help of GenAI tools. These findings from computing education can give us early insights into the potential future impacts that LLMs will have on learning systems across educational domains.

Consequently, our review addresses the following research questions:

1. Which pedagogical approaches guide the design and use of customized, student-facing LLM tools for computing education?

2. What technical approaches are employed to develop these tools?

3. What empirical data sources and evaluation metrics are used to assess these tools?

4. How do these tools impact learners?

Building on the answers to our research questions, we explore system-level implications of LLMs that extend past individual classrooms to institutional policies and long-term educational structures. Thus, we discuss what current literature says about the computer science and these findings' projection onto necessary changes in computing curricula and education as a whole. We argue that academia should be proactive in adapting its curricula and assessments and promoting productive human-AI collaboration rather than resisting these technologies.

Following PRISMA guidelines (Page et al., 2021), we conducted a systematic search across three major databases (ACM, IEEE, and ScienceDirect) for studies published between 2021 and mid-2025. After rigorous screening for eligibility and quality, 52 studies were included. Our overarching goal is to create a unified evidence base on student-facing tools backed by user studies, to guide researchers, developers, and educators in building and deploying GenAI tools for learning.

The rest of this article is organized as follows. Section 2 describes the systematic review approach adopted for this study, describing the search strings, the inclusion/exclusion criteria, quality screening, and the procedure for analyzing included studies. Section 3 describes results organized by our research questions. Section 4 synthesizes the results, showing key insights, research gaps, and implications for systemic transformation of computing education. Section 5 summarizes the threads to validity of this study. Section 6 concludes the study by summarizing its contributions and highlighting the significance of our findings.

2. Methodology

Our study follows the PRISMA guidelines (Page et al., 2021), and the ACM SIGSOFT Empirical Standards for Systematic Reviews (Ralph et al., 2021). Adapting PRISMA for our research context [1], we conducted this review in four phases: (i) literature search, (ii) article screening based on inclusion and exclusion criteria, (iii) article screening for quality assessment, and (iv) data coding and analysis. The flowchart of our search and paper screening process is shown in Figure 1.

2.1. Literature Search Strategy

We conducted the literature search within three major databases in the field of computer science and engineering[2]: ACM Digital Library, IEEE Xplore, and ScienceDirect, with the date range of January 2021 to the middle of June 2025. We selected 2021 as the starting point to capture the past five years of leading research in this field. To reduce noise and improve precision, the search was limited to title, abstract, and author-defined keywords. We identify keywords within five concept blocks related to our review focus: computing topics, educational context, population, tool framing, and LLM models, as shown in Table 1. These strings were iteratively refined until gains in retrieval plateaued. We used wildcard expansions for terms appearing in multiple forms (e.g., GPT-4), but kept terms that are typically without variants (e.g., Claude and OpenAI) in their exact form. The search strings were adjusted to accommodate each database's syntax requirements (e.g, IEEE Xplore's limit on ten wildcards, and ScienceDirect's eight Boolean connector limit).

Our initial literature search yielded 1,725 papers across all databases. We then removed time-ineligible papers (published before 2021) and non-research publications (e.g., reviews or book chapters) using search engine filters, and eliminated duplicates automatically through Covidence (Covidence), resulting in 43 papers.

[1]We did not undertake certain PRISMA-specified steps, e.g., specification of effect measures, or certainty-of-evidence grading, since our synthesis is qualitative and the included studies are methodologically heterogeneous. We performed a study-level quality appraisal (relevant but not identical to PRISMA's risk of bias) using an adapted CASP checklist described in Section 2.3

[2]We refer to computing to be inclusive of related fields such as computer science and computer engineering.

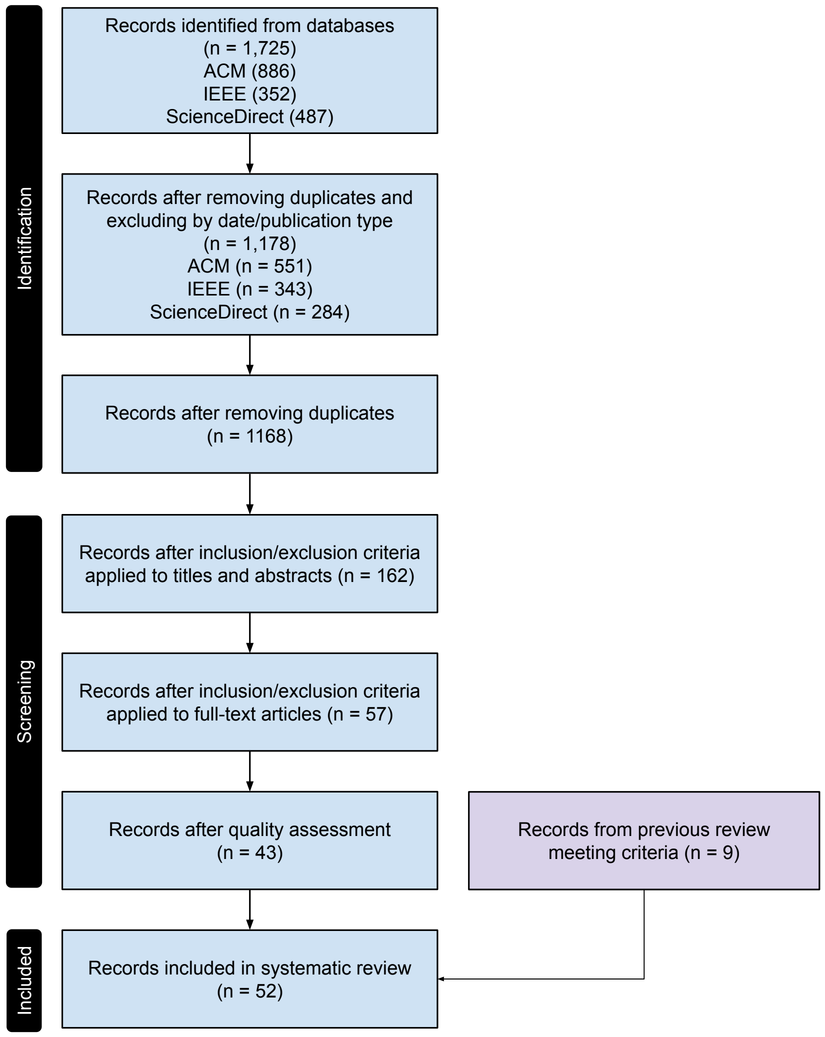
Identification
Screening
Included
Records identified from databases
(n = 1,725)
ACM (886)
IEEE (352)
ScienceDirect (487)
Records after removing duplicates and excluding by date/publication type
(n = 1,178)
ACM (n = 551)
IEEE (n = 343)
ScienceDirect (n = 284)
Records after removing duplicates
(n = 1168)
Records after inclusion/exclusion criteria applied to titles and abstracts (n = 162)
Records after inclusion/exclusion criteria applied to full-text articles (n = 57)
Records after quality assessment
(n = 43)
Records from previous review meeting criteria (n = 9)
Records included in systematic review
(n = 52)

Figure 1: Flowchart of Paper Identification and Screening Process

Table 1: Concept blocks and database-specific search strings

Database / Concept block		**Search string**
ACM DL	Computing Topics	programming OR comput* OR “software engineering” OR “data structures” OR algorithms OR databases OR “operating systems" OR “computer networks" OR "machine learning" OR "artificial intelligence" OR "human-computer interaction" OR cybersecurity
	Educational Context	undergraduate* OR universit* OR colleg* OR course* OR class*
	Population	student* OR learner* OR novice*
	Tool Framing	chatbot* OR assistant* OR tool* OR system* OR platform*
	LLMs	LLM* OR (“large language" AND model*) OR “generative AI" OR GPT* OR ChatGPT OR OpenAI OR Copilot OR Claude OR Gemini OR Grok OR Llama OR Mistral OR DeepSeek OR Qwen
IEEE Xplore	Computing Topics	programming OR comput* OR “software engineering" OR “data structures" OR algorithms OR databases OR “operating systems" OR “computer networks" OR “machine learning" OR “artificial intelligence" OR “human-computer interaction" OR cybersecurity
	Educational Context	undergraduate OR university OR college OR course OR class
	Population	student OR learner OR novice
	Tool Framing	chatbot OR assistant OR tool OR system OR platform
	LLMs	LLM* OR “large language model" OR “generative ai" OR gpt* OR chatgpt OR openai OR copilot OR claude OR gemini OR grok OR llama OR mistral OR deepseek OR qwen
ScienceDirect	Computing Topics	programming OR computing OR "computer science"
	Educational Context	undergraduate* OR universit* OR colleg* OR course* OR class*
	Population	student* OR learner* OR novice*
	Tool Framing	chatbot* OR assistant* OR tool* OR system* OR platform*
	LLMs	“large language model" OR LLM* OR “generative AI" OR GPT* OR ChatGPT OR OpenAI OR Copilot OR Claude OR Gemini OR Grok OR Llama OR Mistral OR DeepSeek OR Qwen

2.2. Screening: Inclusion and Exclusion Criteria

We established five inclusion and six exclusion criteria to address two levels of requirements. At the broad level, we required peer-reviewed papers published within the last five years, written in English, not duplicates, containing at least four pages of main text (excluding references), and accessible through university subscriptions. At a more granular level, studies needed to: (i) focus on student-facing tools, excluding systems designed for instructors or teaching staff; (ii) operate within computing education contexts or involve primarily computing students [3]; (iii) target undergraduate education; (iv) involve tools beyond simple wrappers or commercial LLM products (e.g., ChatGPT), such as research prototypes, custom LLM technologies, or LLM extensions incorporating novel design, and (v) incorporate a user study. It should be noted that, in a few cases where multiple studies examined the same tool, we included them, as user studies could still provide valuable insights for this review.

This screening was conducted in two stages. Stage 1 was a title and abstract review conducted by the first author, yielding 162 papers. Stage 2 was a full-text screening conducted by the second author with active guidance from the first author, reducing the set to 57 papers.

Table 2: Inclusion and exclusion criteria

Inclusion Criteria	**Exclusion Criteria**
1. Paper published in 2021 or later	1. Paper is not in English.
2. Paper involves a student-facing LLM tool	2. Paper is not a peer-reviewed research study.
3. Research is within computing education or mostly focused on computing students	3. Paper is not accessible via university subscriptions.
	4. Paper is a duplicate study.
4. Research conducted in undergraduate setting	5. Main text (excl. references) is under four pages.
5. Research includes a user study with students.	6. Tool is a standalone commercial product.

2.3. Screening: Quality Assessment

We adapted the Critical Appraisal Skills Programme (CASP) qualitative studies checklist (Lorås et al., 2021; noa, 2025) for selecting rigorous studies to include. The list is as follows:

1. Was there a clear statement of the *aims* of the research?

[3]This condition included studies where tools were not specifically designed for computing education but primarily involved computing students.

2. Was the *research design* appropriate to address the aims of the research?

3. Does the paper clearly determine the research *methods* (subjects, instruments, data collection, data analysis)?

4. Was the *data analysis* sufficiently rigorous?

5. Is there a clear statement of *findings*?

6. Does the paper meet all of the above criteria? (if so, include)

These criteria evaluate essential aspects of papers across study types, from design methodology to result presentation, which help ensure the relevance, rigor, and clarity of included studies. The final criterion regarding research value serves as a decisive factor. In other words, papers must meet all criteria for inclusion, with any single “no" resulting in exclusion. Two researchers independently applied these criteria to all papers and discussed disagreements to reach consensus, which yielded 43 papers. These papers were then organized in a spreadsheet for coding and analysis.

2.4. Inclusion of Supplemental Papers

Our systematic literature review evolved through iterative refinement. Initially, we explored the broad intersection of LLMs and computing education; however, the exponential growth of studies in this area made meaningful synthesis challenging and necessitated a more targeted focus. We therefore narrowed our scope to student-facing LLM tools with empirical data in undergraduate computing education. After consolidating papers under this refined focus (2018-2024, n = 43) and beginning our analysis, we decided to conduct a new comprehensive systematic review following a refined search strategy, as described in this section, to ensure rigor, reproducibility, and consistency. When comparing results from this refined search, i.e., the basis of this paper, against our initial broader review, we identified several relevant papers (n = 9) that met our inclusion and quality criteria but were not captured by the new search parameters, likely due to our initially broader search terms. To provide a comprehensive synthesis of the literature, we incorporated these additional papers into our final analysis, resulting in a total of 52 papers for review.

2.5. Paper Review and Coding

We extracted data from papers on two high-level dimensions: research question-driven categories and descriptive categories. For the former, we extracted research purposes, pedagogical approaches, technical approaches, data collection methods, evaluation metrics, and impacts. For the latter, we extracted contextual information including grade level, subject domain, user interface (UI) modality, geographic location, and publication year. The complete list of papers with their detailed corresponding codes is presented in Appendix A.

We followed Braun and Clarke's six-phase reflexive thematic analysis for the coding process: familiarization, coding, theme development, theme review, theme definition, and write-up (Braun et al., 2019; Braun and Clarke, 2024) for most categories. Four researchers divided the categories, reviewed papers independently, coded for the specific categories, and met regularly to discuss and consolidate the coding taxonomy and coding results for each paper. To ensure accurate representation of studies, researchers reviewed all manuscripts to extract data for their assigned categories.

3. Results

This section provides a descriptive overview of the included studies, followed by findings organized by research question. Figure 2 presents a comprehensive framework that synthesizes the key characteristics of student-facing LLM tools identified in our systematic review, including pedagogical approaches (RQ1), technical approaches (RQ2), and evaluation strategies (RQ3). A complete list of papers with detailed labels is provided in Appendix A.

3.1. Descriptive Overview of Included Studies

3.1.1. Year and Country

Although our inclusion window spanned five years, all studies in the final pool were published from 2023 onward, with 4 in 2023, 36 in 2024, and 12 in 2025. This is likely because public access to LLM tools dramatically increased in Fall 2022. Figure 3 illustrates the geographic distribution by country, with the United States accounting for the most studies (16), followed by New Zealand (6) and Germany (4).

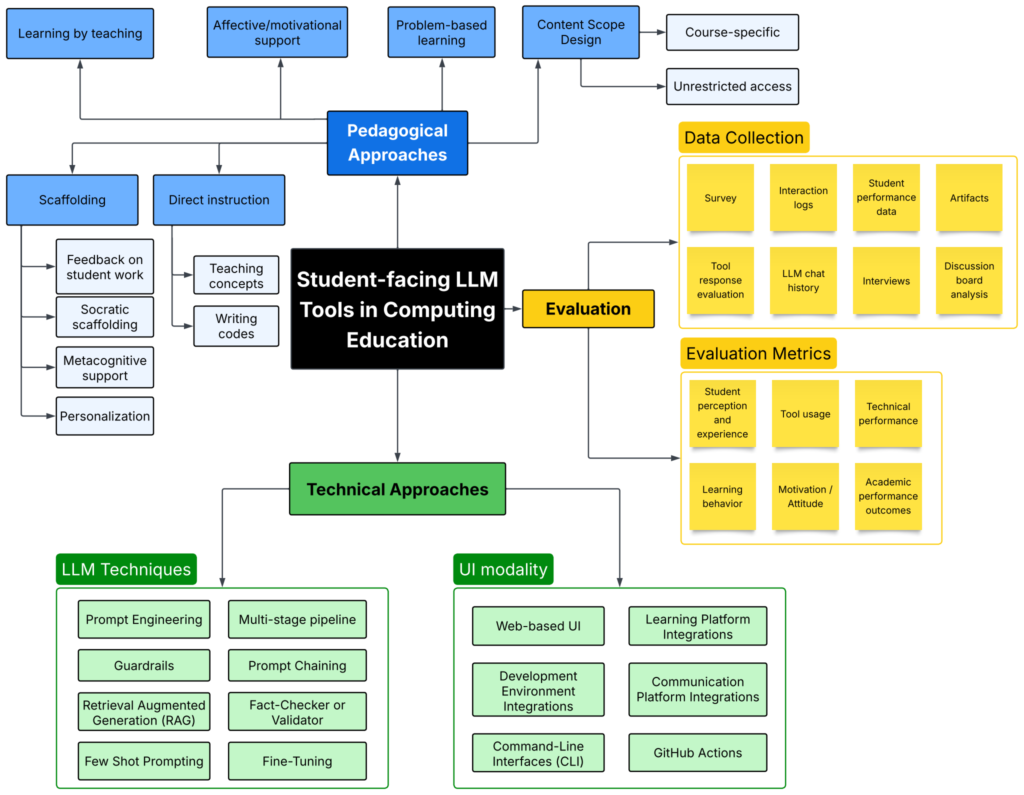

Figure 2: Conceptual framework of student-facing LLM tools in computing education, showing pedagogical approaches, LLM adaptation methods, and evaluation strategies identified in the systematic literature review.

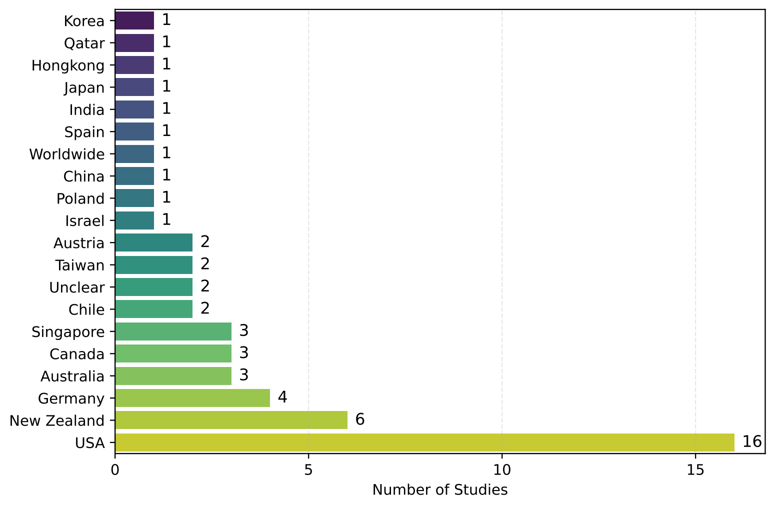

Figure 3: Geographic Distribution of Studies

3.1.2. Grade and Subject

We categorized the papers into two groups based on their user study population: early-stage (first and second-year students) and late-stage (third and fourth-year students) undergraduate students. The majority of papers (n = 38) focused on early-stage undergraduate courses, while only three (n = 3) concentrated on late-stage undergraduate courses. An additional four papers (n = 4) included mixed populations, comprising studies that deployed their tools in multiple year levels (Kuramitsu et al., 2023; Lui et al., 2024; Qadir, 2025). While all studies examined computer science or computer engineering courses, we did not categorize papers by specific course subject due to the varying nomenclature used across institutions. Nevertheless, introductory programming was a standout in frequency. Other courses included cybersecurity, algorithms and data structures, advanced digital design, capstone software engineering, design thinking, databases and information systems, and object-oriented programming.

3.2. RQ1. Pedagogical Approaches Guiding Tools' Design

To analyze the pedagogical foundations of LLM-based educational tools (RQ1), we developed a taxonomy drawing from established learning science principles (Weinstein et al., 2018), research on effective tutoring practices (both human and AI tutors) (Zhang et al., 2024; Mercer and Howe, 2012), and

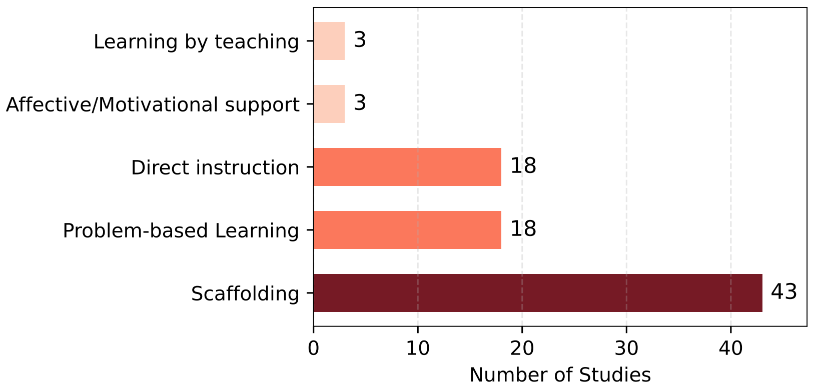

Figure 4: Distribution of Pedagogical Approaches Employed Across Studies

recent work on pedagogical capability evaluation of LLM-powered AI tutors (Maurya et al., 2025). This taxonomy includes six broad categories: scaffolding, problem-based learning, direct instruction, affective/motivational support, learning by teaching, and content scope design. We coded the type of pedagogical support by analyzing system features, prompt design strategies, and user interaction patterns described in the reviewed literature. The count for each of these is demonstrated in Figure 4, and detailed categories labeled for each paper are in Appendix B.

3.2.1. Scaffolding

Scaffolding emerged as the most prevalent pedagogical approach implemented in the reviewed tools (n = 43). This approach aligns with Vygotsky's zone of proximal development theory, where the tools provide support to help students achieve learning goals they cannot accomplish independently (Vygotsky, 1978). Scaffolding usually involves active learning principles that engage students (Chi and Wylie, 2014). Examples of these active learning principles can be seen within the five scaffolding subcategories we defined based on papers that explicitly mentioned specific approaches, including: 1) feedback, 2) Socratic methods, 3) metacognitive supports, and 4) personalization, described in more detail below.

1) ***Feedback on student work*** (n = 18) was the most common subcategory, where tools provided targeted responses to student work, offering corrections, suggestions, and guidance. For instance, Taylor et al. (2024) provides actionable error explanations and guidance to help students understand and resolve problems. Similarly, Gipy (Gabbay and Cohen, 2024) generates LLM feedback on student assignment submissions regarding whether the solution is correct or incorrect, and if incorrect, identifies the errors and discrepancies. In Smith et al. (2024), students write an explanation (EiPE: Explain-in-Plain-English), and the system provides immediate feedback by generating code from their explanation and testing it for correctness, allowing students to see whether their conceptual understanding aligns with functional code.

2) ***Socratic scaffolding*** (n = 14) instructs the LLM not to reveal direct answers to students (Knezic et al., 2010). This "no direct answer" approach is applied to most systems and reflects the pedagogical principle that student-generated knowledge is more durable than passively received information.

3) ***Metacognitive support*** (n = 7) refers to the tool prompting students to reflect on their thinking processes through open-ended questions. This is closely relevant to self-regulated learning theory (Zimmerman, 2002). For example, Menezes et al. (2024) asks students "standup" questions such as "What did you do yesterday? What are you doing today? Is anything blocking you?", which are commonly used in the technology industry.

4) ***Adaptive and personalized learning*** (n = 6) represents systems that adapt their feedback and pace based on student responses or detected misconceptions, tracking user input patterns, and adjusting challenge levels. For instance, Yang et al. (2024a) presents different types of hints as students progress through programming exercises.

3.2.2. Problem-based learning

Problem-based learning (n = 18) is the second most common approach utilized by these tools that guides students through problem-solving steps in open-ended problems (typically programming assignments). For instance, Bassner et al. (2024) accesses the problem statement, student code, and automated feedback to provide tailored advice that helps students navigate their programming challenges.

3.2.3. Direct instruction

Direct instruction (n = 18) includes tools that explicitly provide educational content rather than guiding students to discover for themselves. This contains two primary subcategories: (1) teaching concepts, where systems generate step-by-step explanations, generating exercises for students and correcting their answers (Neumann et al., 2025) and answering students' conceptual questions (Kazemitabaar et al., 2024), and (2) writing codes, where systems generate complete code solutions, sample code, and documentation (Kuramitsu et al., 2023; Birillo et al., 2024).

3.2.4. Affective/motivational support

Affective/motivational support (n = 3) provides encouragement, emotional support, and normalizes errors as part of the learning process. For example, Goddard et al. (2024) designed a self-disclosing chatbot to promote student self-disclosure of learning challenges and to help students feel less alone. CodeHelp uses prompts to generate responses that are "positive and encouraging" (Denny et al., 2024b).

3.2.5. Learning by teaching

This (n = 3) is an effective pedagogical approach where learners take on the role of instructors, explaining concepts to the AI or treating it as a learning partner in collaborative problem-solving scenarios (Allen and Feldman, 1973). For example, Rogers et al. (2025) introduced an LLM teachable agent "MatlabTutee" and showed it can replicate the ideal learning-by-teaching interaction achieved with a human partner, encouraging students to practice elaborating and explaining topics at levels similar to human interaction.

3.2.6. Content Scope Design

Another pedagogical design dimension we considered was **content scope design**: whether the LLM was specifically designed to provide responses limited to **course-specific content** to maintain curriculum alignment, or rather provide **unrestricted access** for open-ended exploration. We only tagged systems that explicitly addressed this design choice. Our results reveal that 15 systems were implemented with high constraints, limited to course-specific contexts, while three systems did not impose constraints on the context or types of help offered. These low-constraint systems usually focus on exploring students' help-seeking behavior in the field (Lyu et al., 2024; Shin et al., 2024) and the impact of other design characteristics, such as avatar design (Tan et al., 2025a).

3.3. RQ2. Technical Approaches Employed in Tools' Development

For RQ2, we analyzed how LLM tools are developed along two dimensions: their user interface modalities and the underlying LLM techniques. We identified six types of UI modalities and eleven LLM techniques. This section provides a detailed description of these technical approaches.

3.3.1. UI Modalities

We categorized the tools' UI modalities into six groups: (1) Web-based UI, (2) Learning Platform Integrations, (3) Development Environment Integrations, (4) Communication Platform Integrations, (5) Command-Line Interfaces (CLI), and (6) GitHub Actions. Their distribution across studies is shown in Figure 5. Three of these are general: (1) web-based UIs are separately accessed chatbots on the web, (2) learning platforms such as Moodle or Canvas, or (3) communication platforms such as Slack. The remaining groups are related to how students access coding environments within their CS classes. Web-based UIs represented the vast majority of modalities in 39 of the studies (Pădurean et al., 2025; Pua et al., 2025; Smith et al., 2024), thanks to features like cross-platform accessibility, no user-side installations, and centralized updates. Learning platform integrations ranked second with six studies (Fan et al., 2025; Pankiewicz and Baker, 2024; Neumann et al., 2025). These enable assistance directly where users engage with lessons, assessments, or assignments. Development environment integrations (e.g., Visual Studio Code, where students write code) followed by five studies (Kuramitsu et al., 2023; Birillo et al., 2024; Woodrow et al., 2024), allowing for lower context switching during coding and debugging, a highly beneficial feature for a learner. Communication platform integrations (e.g., within Slack) were next (Goddard et al., 2024; Neyem et al., 2024b; Menezes et al., 2024) as they can provide AI-powered assistance within readily established team communication workflows, and their interaction styles are already via natural language, as are LLMs. CLI integrations were selected by researchers offering compiler-integrated AI (Renzella et al., 2025; Taylor et al., 2024), and GitHub Actions integration was used to incorporate AI-powered code review feedback directly into development pipelines (Crandall et al., 2023).

3.3.2. LLM techniques

We identified eleven LLM techniques proposed or adopted into customized LLM tools across the reviewed papers. Since this field is still developing, different authors define and apply these concepts with varying degrees of precision, and some of these approaches may overlap. While we worked to classify them consistently, some variation in how these approaches are conceptualized remains inevitable. The distribution of these approaches is illustrated in Figure 6.

1) Prompt Engineering (n = 51) is the most frequent approach used to build the customized LLM tools in the reviewed studies. Prompt engineering

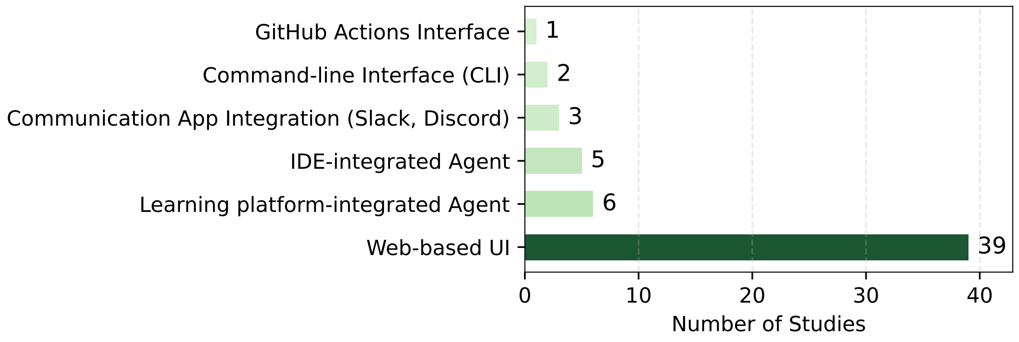

Figure 5: Distribution of UI Modalities Across Studies

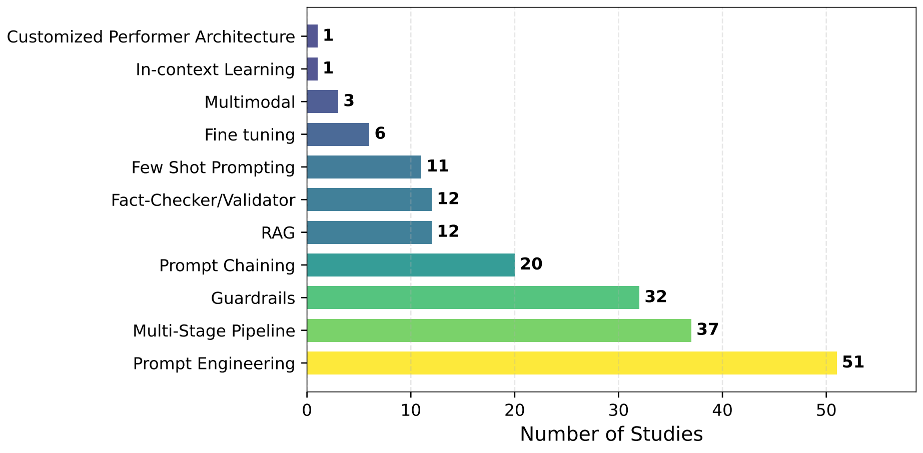

Figure 6: Distribution of Data Collection Methods Used across Studies

provides a fixed system instruction in the prompt (e.g., “do not give direct answer to students”) along with additional context, ranging from the student’s query alone to external information (e.g., runtime state or evaluation results), incorporated into the prompt prior to model inference. For example, KOGI, developed by Kuramitsu et al. (2023), integrates runtime data from the Jupyter coding environment into the prompts before sending them to LLM. Rivera et al. (2024) integrates multiple elements into the prompt, including examples of complex data structures, the task’s function header (renamed to avoid bias), the student’s plan, step-by-step instructions for converting the plan into code, and relevant functions defined by the student in prior tasks.

2) Multi-Stage Pipeline (n = 37) includes tools that involve multiple stages, with typically different roles assigned to each stage. For example, CodeTailor, created by Hou et al. (2024), where a student submits incorrect code, the tool generates a personalized, correct solution similar to the student’s code, and creates an interactive Parsons puzzle for students to learn from their mistakes. During this process, when the initial LLM-generated solution fails correctness tests or lacks sufficient similarity to the student’s code, the system makes additional API calls with updated prompts, including failure reasons and the code from the previous attempts. As another example, Neyem et al. (2024b) proposed a Slack-based tool for helping students in improving their software project stand-up reports, which receives students’ stand-up reports, uses LLMs to process them and generate suggestions, sends these recommendations to students, and allow students to keep or revise their report and submit them, which the bot collects and stores along with the original reports and suggestions.

3) Guardrails is the third common approach in tools (n = 32), which involves using guardrails, which can be constraints expressed via instructions in the prompt or mechanisms beyond wording, e.g., code-block detection. For instance, Hassan et al. (2025) used an extensive prompt, including “*Avoid giving code-specific hints; instead, provide hints for independent understanding of the code.*" Similarly, Rivera et al. (2024), who designed a tool for the translation of students’ plans to code and test suites for evaluating them, used instructions for LLMs to not correct errors in students’ plans to allow students to find and resolve their mistakes.

4) Prompt Chaining, as the fourth common method (n = 20), incorporates tools where two or more sequential LLM calls are made and the output of one serves as the input to the next. As an example of this, Iris, developed by Bassner et al. (2024), first checks the student's query relevance, retrieves relevant exercise context and student code with build log from the Artemis learning management system, generates feedback, and finally self-reviews the response to ensure it follows the guidelines. Jacobs and Jaschke (2024) took a different approach with a two-stage RAG architecture. In the first stage, their system augments student code with compiler output and problem descriptions. Using this context, it generates internal questions about programming concepts missing from the student's code. The second stage retrieves answers to these generated questions from the course lectures transcripts and sends them to the LLM to generate the final response.

5) Retrieval Augmented Generation (RAG) (n = 12), which grounds LLM responses in an external corpus (such as lecture slides and lecture transcripts), was also a relatively common approach among the reviewed papers. For instance, Neumann et al. (2025) developed MoodleBot, which integrates with the Moodle learning management system and uses RAG to provide course-specific assistance to students. Another example is RAGMan, proposed by Ma et al. (2024), which incorporates a knowledge base from project descriptions and historical student discussion posts to provide course-specific guidance without revealing solutions.

6) Fact-Checker or Validator (n = 12) means a correctness check integrated into the tool's pipeline, where either a component was responsible for testing, or human vetting was enforced before delivery, or suggested through tool-embedded tests after delivery. For example, Birillo et al. (2024), in their tool for next-step hint generation, applied static analysis to LLM-generated responses to control hint size and code quality. As another example, Denny et al. (2024a) developed a tool that requires students to write prompts to generate code that matches given outputs. The system constrains the LLM to produce only code through prompt instructions, tests the generated code in a sandbox, and displays test success or failures directly to students for prompt refinement.

7) Few-Shot Prompting (n = 11), as a specific form of prompt engineering, integrates one or more examples of query and response into the prompt to teach LLM the desired behavior or pattern in response. For instance, in the tool that helps students with compiler errors, Pankiewicz and Baker (2024) included an example of an ideal response in the prompt, which includes an error explanation, a solution strategy, and a hint about the underlying concept.

8) Fine-Tuning (n = 6) was employed by a small number of studies to adapt LLM responses to educational contexts and desired interaction styles (Jin et al., 2024; Menezes et al., 2024; Gao et al., 2025; Liu et al., 2025a). For instance, Liu et al. (2025a) illustrated the potential of fine-tuning by aligning model responses with specific instructional strategies and tones. They assembled a diverse dataset of 50 student-tutor conversations covering code generation, debugging, and conceptual questions, then used this data to train an LLM. They found fine-tuning more effective than few-shot prompting in reflecting nuanced pedagogical requests. Menezes et al. (2024) created a chatbot for facilitating daily stand-up updates in project-based courses. They fine-tuned an LLM using a dataset of 566 human-rated standups and found that it outperformed zero-shot learning significantly in two tasks: determining whether a standup response was vague/specific, and whether it represented certain work hours.

Less common LLM techniques were: **(9) multi-modality** (n = 3), i.e., tools incorporating non-text content in their ingestion or generation process; **(10) in-context learning** (n = 1), which in their tool, Huo et al. (2024) translated it to feeding LLM with supplementary course materials without a complicated RAG architecture; and **(11) customized performer architecture** (n = 1), where Wang (2023) proposed an optimized version of Performer model, which improves feedback generation in programming education and targets space-time complexity and inference speed.

3.4. RQ3. Empirical Data Sources and Evaluation Metrics in Tools' Studies

To assess the effectiveness and overall impact of the developed tools, the studies employed various evaluation approaches, both in the data collected and the outcomes measured. Of the 52 papers, a substantial proportion used multiple data-collection methods (n = 38) or measured multiple types of student outcomes (n = 46), which reflects a tendency to mixed-method, multi-dimensional evaluations in the area. This section illustrates the common data collection methods and evaluation metrics used in the reviewed studies.

3.4.1. Empirical Data Sources

We identified eight categories of data sources: surveys, interaction logs, tool-response evaluations, LLM chat histories, student performance data, interviews, artifacts, and discussion board posts. The distribution of these sources is shown in Figure 7. Below, we describe the most frequently used data collection types.

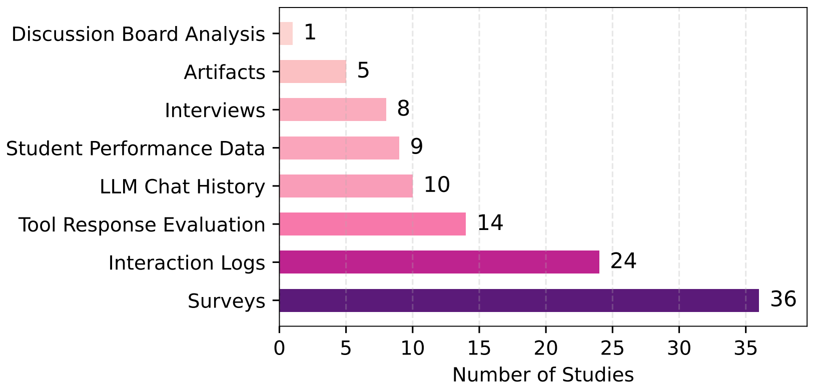

Figure 7: Distribution of Data Collection Methods Used across Studies

1) Surveys (n = 36) emerged as the most common data source, capturing students' perceptions and feedback on the implemented tools. These surveys varied in both design and administration methods. Some were deployed at regular intervals (e.g., weekly or after each activity), while others were embedded within the tool for real-time responses or administered only upon completion of the interaction. Survey designs ranged from numeric satisfaction and feedback ratings (Neyem et al., 2024a; Frankford et al., 2024; Tan et al., 2025a), to open-ended questions for qualitative feedback (Qadir, 2025).

2) Interaction Log Analytics (n = 24) involved analysis of tool usage data to understand user engagement with the tools. For example, Birillo et al. (2024) analyzed all internal system actions performed during task completion. Renzella et al. (2025) analyzed the relationship between different types of error assistance requests and reported that, as students progressed, their requests shifted from compile-time errors toward runtime errors.

3) Tool Response Evaluation (n = 14) addressed the uncertainty of LLM-generated responses through reviews of the tool outputs by the instructional team. For example, researchers assessed tool responses based on students' ratings of their correctness (accuracy) (Neyem et al., 2024a; Yang et al., 2024a; Jury et al., 2024; Pankiewicz and Baker, 2024; Denny et al., 2024b). With a different approach, Neyem et al. (2024b) conducted a linguistic analysis of the recommendations utilizing LLMs to gauge reading ease and comprehension complexity. Neumann et al. (2025) evaluated AI-generated responses using a combination of TA feedback on accuracy and quality, alongside an automated LLM-driven assessment pipeline.

4) LLM Chat Histories (n = 10) were collected to examine student-LLM dialogue patterns to understand the interaction dynamics and help-seeking behaviors. In *KOGI*, the authors observed that students in the Algorithms course primarily sought explanations for code errors, requesting fixes, whereas students in the Data Science course more frequently requested additional code examples, suggesting distinct patterns in how students sought AI assistance (Kuramitsu et al., 2023). The student-LLM chat analysis in Ma et al. (2024) revealed that AI support helped students refine and correctly formulate their queries; engaging in dialogue with an AI tutor enabled them to clarify their inquiries and explore them in greater depth.

5) **Additional methods** were employed less frequently but also provided valuable data. **Student performance data** (n = 9) enabled assessment of learning outcomes, such as analyzing the effectiveness of AI grading on project success and individual contributions (Menezes et al., 2024). **Interviews** (n = 8) captured qualitative insights into student experiences and preferences, with studies like Hou et al. (2024) finding that 88% of students preferred interactive AI tools over direct solution generation for collaborative problem-solving opportunities. **Artifact analysis** (n = 5) examined student work products to understand engagement patterns, such as investigating how students incorporated AI feedback and analyzing the nature of code edits following AI guidance (Woodrow et al., 2024). Discussion board data (n = 1) was used in one study to train LLMs on historical question-answer patterns and integrate the system into discussion platforms to efficiently generate responses for unanswered or unresolved questions (Huo et al., 2024).

3.4.2. Evaluation Metrics

To understand the effectiveness of LLM-based educational tools and educational impact, the collected data was analyzed across six primary categories: student perception and experience, tool usage patterns, technical performance, academic performance, learning behavior changes, and motivation and attitude. The distribution of these categories across studies is in Figure 8.

1) Student Perception and Experience (n = 43) represents the most frequently measured outcome, typically through administering surveys. This category encompasses perceived usefulness, satisfaction scores, willingness to continue using, and users' preferences compared to human tutors or alternative systems. For instance, Neyem et al. (2024a) gathered student ratings

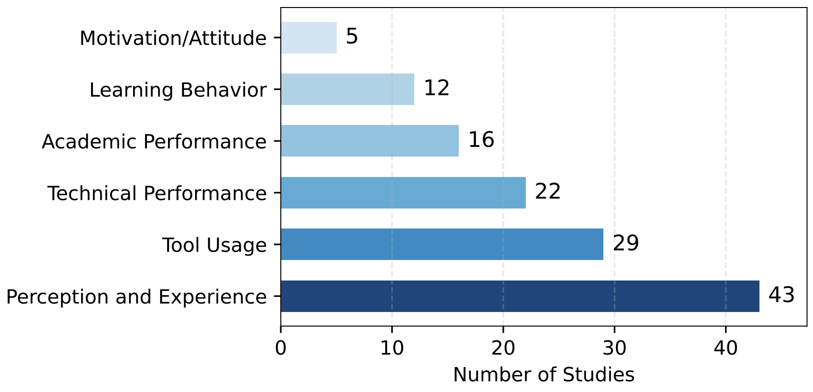

Figure 8: Distribution of Evaluation Metrics Identified in Studies.

of AI-generated responses on usefulness and impact of student affect, and Pankiewicz and Baker (2024) on ease of understanding, relevance, novelty, and serendipity (positive surprise). Some studies employed established theoretical frameworks such as the Technology Acceptance Model (TAM) (Frankford et al., 2024; Tan et al., 2025a). Other researchers examined specific aspects of the user experience, such as Hussein et al. (2024), who focused on students' usage experiences, and Hou et al. (2024), who collected self-reported engagement scores. This type of outcome also includes qualitative feedback collected in open-ended survey questions (Qadir, 2025).

2) Tool Usage (n = 29) metrics focus on behavioral patterns of student interaction with the systems. Researchers measured tool utilization frequency to understand adoption rates (Lui et al., 2024), time spent engaging with the tool (Yang et al., 2024a), query quality improvements in student question formulation (Denny et al., 2024a), and help-seeking behavior patterns identifying when and how students requested assistance (Taylor et al., 2024; Ma et al., 2024).

3) Technical Performance (n = 22) evaluates the system's capability and reliability. Key metrics include response correctness rates (Taylor et al., 2024), hallucination incidents (Liu et al., 2024a), response clarity ratings (Neyem et al., 2024b; Kuramitsu et al., 2023), and error identification accuracy, measuring the tool's ability to correctly detect student mistakes (Rivera et al., 2024; Pankiewicz and Baker, 2024).

4) Academic Performance Outcomes (n = 16) measure traditional educational success indicators. These include course grades or assessment performance on exams, assignments, and exercises (Pădurean et al., 2025), course completion rates (Huo et al., 2024; Smith et al., 2024), and pre/post test scores measuring performance changes before and after tool implementation (Yang et al., 2024a).

5) Learning Behavior (n = 12) captures how students' learning processes change with tool use. Metrics include independence development, measuring reduced need for external help (Liffiton et al., 2024), active participation and involvement levels (Yang et al., 2024a), students' ability to identify and fix mistakes (Gabbay and Cohen, 2024), dependency patterns, recognizing signs of over-reliance on the tool (Ahmed et al., 2025; Liu et al., 2025a), and critical thinking development (Hassan et al., 2025), providing evidence of analytical skill improvement.

6) Motivation/Attitude (n = 5) represents an important psychological dimension in CS education. This includes motivation and interest changes in learning computing subjects (Bassner et al., 2024), self-confidence changes through pre/post measures of student confidence (Kumar et al., 2024), and attitude toward computing (Lyu et al., 2024).

3.5. RQ4. Impacts on Students

To answer RQ4, we examined the empirical evidence on how LLM-driven tools impact student learning outcomes, behaviors, and perceptions in computing education. The reviewed empirical studies highlighted both positive and negative impacts of LLM-driven tools. Specifically, they reported that these tools improve student scores, reduce task completion times, and enhance confidence and engagement through continuous availability. However, the evidence also reveals persistent challenges, particularly regarding the provision of adequate scaffolding and the risk of student over-reliance. In this section, we detail these impacts across learning performance, personalized support, engagement, and implementation challenges.

3.5.1. Impacts on Learning Performance

Several studies have reported that LLM-augmented tools helped students with task completion and problem-solving, yielding positive impacts on learning (Fan et al., 2025; Kumar et al., 2024). For instance, Huo et al. (2024) reported that the chatbot contributed to higher completion rates for challenging programming exercises. In addition, Birillo et al. (2024) found that providing hints in multiple formats, such as textual explanations and code, supported students with diverse learning styles, offering particular benefits to novices and visual learners. In the context of group work, AI scoring of stand-up reports effectively improved project success by delivering targeted support to low-performing students (Menezes et al., 2024). Additionally, students appreciated the automated review tool (ART) for its professional and constructive feedback (Crandall et al., 2023).

Studies on the correlation between AI assistant use and student performance showed mixed results; some demonstrated a **positive impact of AI tutors on student scores and overall performance** (Lui et al., 2024; Pankiewicz and Baker, 2024; Liu et al., 2024b; Shin et al., 2024). In Lui et al. (2024), the authors observed that the higher use of GPTutor directly correlated with improved performance, attributing to GPTutor's precise, lecture-aligned answers, validated with references, and follow-up questions that promoted deeper exploration. Similarly, in Pankiewicz and Baker (2024), GPT hints significantly reduced students' confusion and frustration ("confrustion"), enabling greater focus on problem-solving and contributing to higher scores. Additionally, in Denny et al. (2024b), students value the feedback provided by AI teaching assistants, expressing a preference for features that preserve their autonomy, such as scaffolding (e.g., hints) that guide problem-solving rather than giving direct solutions.

3.5.2. Targeted Scaffolding and Personalized Learning Support

Many studies examined the quality of tool-generated exercises, emphasizing their potential to deliver learning support tailored to individual student needs (Pua et al., 2025; Jury et al., 2024; Shin et al., 2024). For example, the automated Parsons' Puzzle generator by del Carpio Gutierrez et al. (2024) increased student engagement and improved programming skills by allowing learners to select the concepts and contexts for their programs. This aligns with prior research suggesting that autonomy is a key feature for learners to choose to engage (Boud, 2012). On the other hand, Pădurean et al. (2025) observed that LLM-generated exercises matched expert difficulty and were perceived positively by teachers and students, creating debugging exercises that catered to learner needs and learning objectives. Moreover, the tool developed by Riazi and Rooshenas (2025) helped students iteratively improve the Entity Relationship Diagrams (ERD) in the Database Systems course, providing targeted guidance by effectively identifying and addressing common errors.

Our analysis indicated that a key objective for many researchers was the integration of targeted scaffolding into these tools (Gao et al., 2025). For instance, Hassan et al. (2025) found that their chatbot provided scaffolded support by assisting with debugging and code decomposition, targeting areas where learners required help to promote problem-solving. On the other hand, *CodeTailor* by changing the order of the correct solution, prompted students to think through the program flow, promoting deeper thinking compared to a direct code solution (Hou et al., 2024). The different formats of hints, i.e., detailed textual explanations including suggested fixes for errors or error location identification, were highly appreciated by students, leading to better performance in (Wang, 2023). Jacobs and Jaschke (2024) utilized a RAG-based approach to connect lecture materials with AI-generated responses, providing concise assistance that promoted problem-solving without directly providing solutions. Jin et al. (2024) demonstrated that their TeachYou system, through its targeted meta-cognitive prompts, guides students through "why" and "how" explanations, fostering deeper knowledge building with minimal cognitive load.

3.5.3. Impact on Learner Engagement

Another key aspect of these tools is their capacity to enhance student engagement thanks to their continuous availability. Researchers consistently noted that such availability encouraged learners to engage with course concepts during off-hours when instructional staff were less accessible. This underscores the vital role these tools play in sustaining students' learning engagement (Liffiton et al., 2024). For instance, Taylor et al. (2024) and Kazemitabaar et al. (2024) reported that nearly half of tool use happened during off-hours, and near exams. In addition, Nelson et al. (2025) reported that the continuous availability of the AI tool *SENSAI* significantly reduced problem-solving time. With a different take, Woodrow et al. (2024) argued that students who received timely feedback were five times more likely to engage with it than those who received delayed feedback, subsequently incorporating the suggestions in the later assignments as well.

Furthermore, Ma et al. (2024) reported that AI tutors provided a judgment-free environment conducive to learning. Similarly, students described AI responses as non-judgmental, which supported non-native English speakers by boosting their confidence (Qadir, 2025). An intriguing study on avatars as AI assistants found that students with more advanced computing skills preferred no avatar, while others trusted deepfake avatars for handling complex questions (Tan et al., 2025a).

3.5.4. Reported challenges

Despite documented benefits, challenges persist, including over-reliance on AI tools and insufficient scaffolding. While some studies showed promising targeted support, many highlighted difficulties in incorporating the human elements (scaffolding) essential for meta-cognition and problem-solving, pointing to the need for guidance beyond automated assistance (Ahmed et al., 2025; Neumann et al., 2025; Yang et al., 2024b; Sheese et al., 2024; Bassner et al., 2024). For instance, Renzella et al. (2025) and Hussein et al. (2024) noticed that students used the tool mostly to learn about compile-time or runtime errors and expressed a desire for better human scaffolding in skill learning. Frankford et al. (2024) reported that students found that the feedback lacked scaffolding and targeted support. Similarly, despite improved score gains for newcomers and majors outside computer science, Lyu et al. (2024) doubted CodeTutor's impact on promoting critical thinking.

Some studies reported little to no improvement or cited issues such as hallucinations, inconsistencies, and reduced reliability when handling advanced or complex problems (Zönnchen et al., 2025; Yang et al., 2024a; Liu et al., 2025a; Kuramitsu et al., 2023). Smith et al. (2024) raised concerns about over-reliance, although the tool helped students quickly complete tasks. On the other hand, Gabbay and Cohen (2024) reported inconsistencies in feedback for advanced coding tasks, despite LLM-generated guidance providing context-aware explanations of errors. Additionally, Rivera et al. (2024) reported that students appreciated the AI assistant for planning and testing edge cases, but found its responses to be inconsistent or insufficient.

4. Discussion

In this section, we first provide an overview of current trends in student-facing LLM tools for computing education by synthesizing findings from our four research questions and identifying key gaps in pedagogical approaches, technical implementation methods, evaluation practices, and documented student impacts. We then outline five future directions for researchers and practitioners. Lastly, we position these findings within the broader context of educational transformation as computing education navigates the emerging human-AI frontier.

4.1. Overview of Current Trends in Student-Facing LLM Tools for Computing Education

4.1.1. RQ1. Pedagogical Approaches Guiding Tools' Design

The predominance of scaffolding as the pedagogical strategy employed across studies reveals a logical extension of LLMs' capabilities to longstanding instructional goals in computing education. The flexibility of LLMs in modulating tone, depth, and structure enables some forms of support that were previously difficult to operationalize at scale (Raihan et al., 2024). The categories within scaffolding demonstrate the emphasis on feedback mechanisms, Socratic scaffolding, meta-cognitive support, and personalization, all of which illustrate the field's intent to preserve student agency in knowledge construction while contextualizing the support to meet student needs and align with the course materials (Taylor et al., 2024; Knezic et al., 2010; Yang et al., 2024a; Zimmerman, 2002). Particularly, the ability of LLMs to integrate multimodal inputs, such as student code, assignment instructions, or course materials, and contextualize the responses, makes them highly suited for personalized scaffolding, problem-based learning, and instructional intents (Neumann et al., 2025; Gabbay and Cohen, 2024; Hou et al., 2024; Yang et al., 2024a). However, like prior approaches to individualized instruction, some studies have shown that the LLM tool's scaffolding may not have been sufficient to adapt to all types of students. A systematic literature review of the past two decades of research on student-centered learning approaches reveals nuance in how these approaches work (Bernard et al., 2019). Their research suggests that teachers who adopt a more student-centered instructional role improve student achievement, while more student choice/autonomy in learning pace and contexts decreases learning. On the other hand, adaptive approaches have improved learning across different levels of student-centeredness, and more active learning approaches are more effective, regardless of students' levels of agency and autonomy (Bernard et al., 2019). The studies in our systematic review appear to reflect the awareness of LLM tool designers regarding the importance of active learner engagement, and they are purposefully adopting a student-centered, active learning approach that provides individualization.

Multiple studies' focus on direct instruction approaches demonstrates researchers' exploration of LLMs' potential as instructional intermediaries that can alleviate faculty workload pressures while being cost-effective and continuously available (Liffiton et al., 2024; Kazemitabaar et al., 2024). Within this category, the emergence of systems that provide complete solutions suggests that instructional contexts requiring explicit guidance, such as teaching programming syntax or step-by-step problem-solving, may particularly benefit from these tools (Hassan et al., 2025; Renzella et al., 2025). This aligns with prior work suggesting that fully worked examples, showing hints that solve a problem completely, can benefit learners (Shih et al., 2008; Salden et al., 2010). Furthermore, the presence of specialized approaches such as meta-cognitive support, personalization, and affective/motivational support suggests that LLMs are being customized in domains where teaching staff interactions are neither entirely feasible nor optimal (Nelson et al., 2025). Studies around learning-by-teaching approaches demonstrate LLMs' role-playing capabilities, which, despite the nuances of this scenario, have shown promising results in improving knowledge building during the problem-solving phase, while still being limited in fostering meta-cognitive skills (Jin et al., 2024). Prior work in learning by teaching, as seen in Betty's Brain by Biswas et al. (2016), suggests that this approach can be highly effective but requires extensive scaffolding to support the complex task of becoming an independent learner for complex problem-solving. The decades of work on this system and intelligent tutoring systems may help LLM tool designers, combined with the ability to quickly create new content using LLMs, along with strategies such as multi-armed bandits to deploy and test diverse adaptations at scale (Rafferty et al., 2019; Schmucker et al., 2025).

The course-specificity dimension reported in the literature, based on studies that explicitly describe this aspect of their tool architecture, reveals a tendency for LLM-based tools to operate on instructor-approved materials, whether through tools particularly built for a course or through general-purpose designs that can be constrained to selected content (Neumann et al., 2025; Pua et al., 2025; Alario-Hoyos et al., 2024; Taylor et al., 2024). This inclination has both advantages and drawbacks. On the positive side, in programming education (as a core component of computing curricula), conventions vary considerably across courses, ranging from stylistic norms to preferred libraries and functions. As a result, divergence from instructor-approved materials, especially in the learning process, can hinder student learning, create confusion, and increase the instructional burden. On the negative side, this choice makes tool performance heavily dependent on the quality and breadth of the curated content. Incorporating lecture transcripts, for example, is valuable in courses with content-intensive lectures, but can add noise in discussion-based or demonstration-focused instruction. Moreover, for tools not equipped to ingest multimodal inputs, constraining them to those materials reliably may bring more detriments than benefits. Applying the lenses of cognitive load theory (one of the most commonly used theories in computing education research (Berssanette and de Francisco, 2021)) and the aptitude-treatment interaction theory (Kanfer and Ackerman, 1989), we suggest that more novice learners may require more highly contextualized or affective scaffolds. In contrast, students with more expertise or prior experience can become more independent learners and leverage more expert-like and general supports with broader and less curated material (e.g., primary sources, online forums, Google search). These findings are similar to those on decades of research on learning by teaching (Biswas et al., 2016) and on becoming an expert (Sigmundsson, 2024); this is a long and complex process and each part of the process needs scaffolding appropriate for the learner's abilities, attitudes (including motivation and self-efficacy), prior knowledge, meta-cognitive skills, and must take into account the interactions between cognitive load and learner tasks (Skulmowski and Xu, 2022).

4.1.2. RQ2. Technical Approaches Employed in Tools' Development

The trend in technical approaches demonstrates a methodological maturity in the field, with researchers consistently selecting approaches that strike a balance between educational effectiveness and technical complexity. This is reflected in the studies' apparent preference for established, practical techniques, such as prompt engineering, prompt chaining, and RAG, which are all capable of addressing specific educational considerations. Through this lens, the popularity of prompt engineering (Gabbay and Cohen, 2024; Crandall et al., 2023; Rivera et al., 2024), a technique that can adjust the tone, depth, and structure of responses, is not surprising, as it is crucial for many educational uses, such as Socratic scaffolding (Jin et al., 2024; Zönnchen et al., 2025), and affective/motivational tone (Goddard et al., 2024).

Following that, the widespread use of multi-stage pipelines (Riazi and Rooshenas, 2025; Hassan et al., 2025; Woodrow et al., 2024), which is found in over half of the papers, demonstrates the field's growing confidence in positioning LLMs as components within larger systems or complex multi-agent frameworks. This trend is also supported by the popularity of prompt chaining (Jin et al., 2024; Alario-Hoyos et al., 2024), which allows LLMs to decompose tasks into smaller subtasks for sequential processing. Given the paramount importance of tools' accuracy and student learning outcomes in educational tool design, researchers have implemented both guardrails to mitigate misuse and inaccuracy, as well as post-processing validators or fact-checkers to verify response quality and reliability.

RAG, initially proposed by Lewis et al. (2020), is popular among researchers (Liu et al., 2025a; Lui et al., 2024; Yang et al., 2024b) since it grounds LLM responses in users' desired documents, which addresses curricular fidelity and, to a reasonable extent, the accuracy of responses. In addition to the advantages and disadvantages of limiting LLM tool responses to course-aligned materials discussed in Subsection 4.1.1, such a provision can decrease learner cognitive load and also aligns with Vygotsky's Zone of Proximal Development, which posits that students can more rapidly learn things that are closest to their existing knowledge (Vygotsky, 1978).

The limited adoption of fine-tuning among researchers can be attributed to the training data required for it, as well as the hardware and software infrastructure needed to support it at scale (Menezes et al., 2024). Moreover, as educational data typically requires Institutional Review Board (IRB) approval, acquiring appropriate training datasets is also logistically challenging. Similarly, multimodal capabilities received little attention in the literature, despite the growing prevalence of drag-and-drop document uploading capabilities in contemporary LLM tools. This suggests their integration into educational tools warrants greater attention.

4.1.3. RQ3. Empirical Data Sources and Evaluation Metrics in Tools' Studies

Our analysis reveals several gaps in how LLM-based educational tools are being evaluated. The widespread adoption of surveys (n = 36) as the primary data collection method is understandable due to their ease of collection, but this has led to the evaluation being heavily focused on student perceptions rather than learning effectiveness. Most studies measure their outcomes through perception and experience (n = 43), while only a small portion evaluates actual learning impacts, such as academic performance (n = 16) and changes in learning behavior (n = 12).

Most studies rely on short-term data collection, such as one-time surveys or week-long logs, rather than tracking long-term learning retention and behavioral changes across multiple semesters and multiple courses. Additionally, the majority of papers (n = 38) focused on early-stage undergraduate courses compared to only three (n = 3) on late-stage courses, with introductory programming being particularly overrepresented. This highlights a gap in equipping students with LLM skills and knowledge as they approach graduation, when they will need these skills for their jobs (Benhayoun and Lang, 2021).

The evaluation of current studies indicates a potential misalignment with broader computing education goals. While computing education research has long emphasized fostering interest in computing (Michaelis and Weintrop, 2022; Decker and McGill, 2019; Margulieux et al., 2019), building problem-solving and collaboration skills (Priemer et al., 2020), and developing career readiness (Scaffidi, 2018; Radermacher et al., 2014), we found that very few studies evaluate the impact of LLMs on these outcomes. Only five studies measured changes in motivation and attitude, suggesting that we may be missing important implications for understanding students' sustained engagement with computing.

These patterns suggest that, while we know students generally like these tools, we have a limited understanding of whether they actually improve learning or prepare students for careers in computing. Future research should focus on a comprehensive evaluation that includes long-term learning outcomes, impacts across different course levels, and alignment with fundamental computing education objectives.

4.1.4. RQ4. Impacts on Students

The assessment of impact can broadly be grouped into learning and performance, personalized support and scaffolding, and student engagement. Empirical evidence suggests that tools are capable of supporting students in completing challenging tasks and providing targeted assistance to low-performing learners. They also offer hints in various formats that help novices and visual learners, which supports contiguity, an important aspect in prior research on designing multimedia learning environments (Mayer and Moreno, 2002) and designing to reduce the cognitive load of learners (Mayer and Moreno, 2003).

However, the impact of these tools on student performance is mixed. Positive effects are reported when tools align with course content, provide context-aware hints, and include follow-up questions that deepen conceptual understanding to support problem-solving. These findings support theories of expertise development, allowing learners to apply domain principles through practice on critical aspects of the task (Sigmundsson, 2024). The quality of tool-generated exercises and the personalized support are another key aspect of impact. Studies show that such exercises effectively engage students, provide tailored guidance that aligns with expert-level difficulty. This supports the need for adaptive responses tailored to individual learners, recognizing that students at different levels require differentiated support (Liu et al., 2025b).

Support for targeted scaffolding and problem-solving remained another issue for researchers. Some studies achieved this through prompting students to reason through problems rather than providing direct solutions. Also offering multiple forms of hints allowed learners to select preferred support, fostering autonomy and agency. These designs help students focus on relevant details, reducing extraneous cognitive load and easing working memory demands during complex problem-solving (Haynes and Ericson, 2021).

Enhancing student engagement through the continuous availability of these tools was another key focus. Researchers observed that this accessibility encouraged learning beyond regular hours. Additionally, the judgment-free environment fostered active engagement and conducive learning conditions. This aligns with prior research demonstrating that learners with lower prior experience may require more motivational support, while those with higher proficiency benefit from targeted, detailed feedback without additional affective guidance. (Kanfer and Ackerman, 1989).

Despite the promises, several challenges remain. The support was mostly limited to foundational topics in a few subjects, and the effectiveness of problem-solving and scaffolding is largely underexplored, underscoring the need to extend validation across diverse courses, topics, and student populations. This highlights the need to develop new methods to address such limitations. Despite these challenges, findings show that LLM tools can effectively provide appropriate support; although there is a need to adapt interventions to individual student needs (Idowu, 2024).

4.2. Future Directions for Researchers and Practitioners in Computing Education

4.2.1. Direction 1: Expanding Pedagogical Approaches

Despite decades of empirical validation in educational research, numerous established pedagogical frameworks remain underexplored in LLM-based educational applications. Our analysis revealed an under-exploration of metacognitive support, personalization tools, effective motivational support systems, learning-by-teaching implementations, and methods that integrate course-specific resources. Other areas, such as peer tutoring (Markel and Guo, 2021) and collaborative learning tools (Rodríguez et al., 2017), were not explored, leaving vast potential for future LLM-integrated learning systems to leverage and support other effective learning supports.

4.2.2. Direction 2: More Advanced Technical LLM Adaptation Methods

The development of technically innovative LLM adaptation models and architectures presents substantial opportunities for researchers with machine learning expertise seeking to contribute to interdisciplinary educational uses. This direction includes both novel training methodologies and architectural innovations specifically designed to address the unique constraints and requirements of educational contexts. These include highly technical approaches to improving knowledge construction and reasoning abilities within LLMs (Zhu et al., 2024).

4.2.3. Direction 3: Integrate Established Educational Research

A promising direction involves systematically integrating decades of design and data-driven insights on adaptive, personalized, and affective learning supports into LLM systems. Our review has identified opportunities for personalization and adaptation that have had a positive impact on student learning, perceptions, and attitudes; however, not all LLM tools have achieved these goals to the same extent. Future systems could leverage established research from intelligent tutoring systems (Arroyo et al., 2009; Woolf et al., 2009), cognitive load theory (Sweller, 1988), and learning by teaching (Fiorella and Mayer, 2013) to accelerate innovation while maintaining high quality for learners with varied prior experience, abilities, knowledge, motivation, and attitudes. Student-centered approaches that value active learning and encourage increasing agency and independence over the trajectory of student learning are promising.

4.2.4. Direction 4: Supporting Instructor Control and Stakeholder Input

There is a critical need for developing low-cost methods that allow instructors without LLM technical expertise or computing resources to integrate their course materials into LLM-based educational tools. Furthermore, due to their drawbacks, including over-reliance and under-performance in specific domains and situations, as well as the sometimes high-stakes educational decisions, stakeholder control becomes essential for the responsible deployment. LLM tools need to be engineered to provide meaningful opportunities for human stakeholders, including instructors, caregivers, administrators, and students, to provide feedback, choose whether to use or not use them, and guide system improvements. The importance of these human oversight features increases with the stakes of educational decisions, particularly when learning interaction data influences high-stakes assessments or academic outcomes (Barnes et al., 2024).

4.2.5. Direction 5: Cross-course potential for learning systems

Given the ability to rapidly deploy and adapt LLM tools for student learning, there may now be potential for more transfer of techniques and personalized insights between systems. Although none of the research reviewed here attempted to build systems that can be ported between learning contexts, an important future direction for learning systems is to determine ways to share the important adaptive gains learned within one course or problem-solving context with future learning support systems. The unique ability of LLMs to translate data into narratives may pose a potential way for such findings to be exported from diverse systems that use divergent types of data and imported into promote for LLM tools. For example, at the end of a learning session, future LLM tools may be rapidly built to provide a textual and vectorized summary of the adaptive strategies and data features (such as time and learning metrics) that could be imported within prompts for future LLM-based tools. Such exports may also be tailored to explain a student's learning strengths, weaknesses, and needs, helping students learn more about their own learning to support metacognitive development, as with open learner models (Hooshyar et al., 2019).

4.3. A Window into Broader Educational Change

Examining the past, we can observe similar trends in the emergence of new technologies and their impact on learning systems as a whole. The emergence of the internet, with its ability to connect, share solutions, and compensate people for creating solutions, has had a profound impact on education. Search engines, such as Google, furthered this impact, as students no longer had to purchase each textbook; instead, they could look for answers online. However, students have always needed to contextualize the found information to their specific course and grade level (with or without help or guidance from their instructors). With LLMs, the story has changed. They offer a single, easily accessible tool that can solve student problems in context, with grades similar to or even higher than those of students themselves. While the studies in this review demonstrate that instructors and researchers still consider it essential to contextualize learning, no innovation to date has challenged the abilities of instructors to support students in undertaking independent, project-based work in the same manner as LLMs.

The field of computing education is unique because it is one of the few fields where students are already learning with continual computer-based feedback on complex, constructed problem solutions (i.e., computer programs designed to solve the posed problems). What is particular about secondary computing educators is that they have always had to adopt and integrate the latest technologies to address problem-based learning, provide feedback, and prepare students for the future of work at the human-computer frontier. The future of work at the human-computer frontier is now the human-AI frontier. All fields will require students to interact with AI to solve problems. Therefore, we believe that studying the current trends in LLM applications in computing education will provide insights into the future of learning systems that integrate generative AI. This research highlights the dimensions that both educators and learners can expect to change as AI becomes increasingly pervasive across various fields.

This systematic literature review highlights some of the challenges that learning systems will encounter. Furthermore, it illustrates how educators can utilize AI to motivate students and personalize learning. We suggest that the current innovations are building preliminary strategies that are needed to prepare learners for the human-AI frontier. We posit that more research and strategies are needed that explicitly prepare learners to work directly with AI for problem solving, including the identification and elaboration of the knowledge that students will need to develop to understand, critically evaluate, and modify the solutions that AI provides. This review emphasizes the importance of continuous reflection and adaptation of learning systems to integrate with AI, thereby supporting both educators and students in ways that facilitate students' development of expertise and the ability to solve complex problems. This means there is a critical need for proactive strategies to support educators in learning about and teaching with these technologies, while emphasizing the importance of human learning and the application of AI to address the larger problems that our world faces.

While this review focused on computing education, it highlights the need for supporting learners who need to interact with AI that can solve problems in their domain. It is becoming increasingly evident that LLM-based tools can solve problems across various domains. Still, their ability to provide reliable and valid responses remains inconsistent, particularly when issues become more complex. However, it is these specific scenarios that are already present in computing education, where we need the most work to determine when and how LLM-based tools require adaptation to continue supporting human learning despite inaccuracies, hallucinations, or lower system capabilities in complex problem-solving. As we tackle more complex global challenges, such as access to food and clean water, and the interactions of geopolitics with health, education, and human rights, it is imperative that we learn to leverage cutting-edge AI capabilities while remaining continually aware of its limitations and biases, and that humans be enabled to learn and contribute to new solutions in complex learning systems.

It is true that AI systems will perform beyond the cognitive capabilities of a large part of the population. Within the field of computing education, this is already the case. Similar to the emergence of automobiles, however, there is the potential to enable every person to leverage new technologies with proper design and consideration of what humans and machines need to know and do. We do not advocate that AI replace humans, but rather that it is essential to develop AI tools that support human learning, enabling individuals to tackle complex cognitive tasks within challenging domains. As the need for human problem-solving continues to grow, this approach becomes increasingly important. Learning systems must continue to strive to promote human learning, enabling human learners to contribute to problem-solving for ever-more complex problems. This requires learning systems that engage learners in learning about AI and the problem-solving domains in which they will operate, and help focus learner tasks by appropriately adapting to the specific learner needs within these tasks, thereby reducing cognitive load in ways that enable and promote human learning, engagement, and motivation.

Generative AI tools, similar to LLMs, will continue to evolve to solve a broader range of problems at higher complexity levels. Our societies and learning systems need to evolve more rapidly to engage all humans in complex problem-solving processes in roles and ways tailored to their unique abilities, thereby contributing to novel and transformative solutions. This will require more interconnected systems that support human-human interactions (such as teacher-student and student-student) and human-AI interactions (such as LLM tool-student and beyond) to promote learning. Learning systems must become more accessible, accept more human modifications by teachers, caregivers, and students as appropriate to specific contexts and needs, and they must continually adapt to changing needs. While we have not focused on this as a particular area, it is also imperative that these evolving learning systems promote AI in education in ethical ways, particularly by centering ethics and impacts on learners as primary design goals, not afterthoughts. A critical need is the development of new frameworks that would have enabled, for example, the authors of the papers in this review direction in LLM tool design to elaborate and adhere to specifically targeted ethics principles and goals (Barnes et al., 2024).

5. Threats to validity

Internal Validity. The subjective nature of inclusion/exclusion criteria, quality assessment, and thematic analysis creates potential for bias and inconsistent decision-making and interpretation. To mitigate these risks, we adhered to a well-defined protocol from the very first phase of finding papers to the thematic analysis, with regular team discussions. Specifically, the two researchers involved in the screening phases followed standardized procedures and independently conducted quality reviews of all selected papers before proceeding to subsequent steps. Additionally, during the analysis phase, all researchers held regular meetings to discuss findings and maintain consistency.

External Validity. Our focus on student-facing tools in computing education limits the applicability of results to other academic disciplines and tool types, such as instructor-facing or institutional platforms. Additionally, most reviewed studies were conducted in the United States, which may restrict the relevance of our findings to regions with different pedagogical approaches and infrastructures. Furthermore, we excluded non-peer-reviewed papers to ensure quality standards; however, we acknowledge that this decision may have excluded some innovative works published on platforms such as arXiv. Finally, our analysis did not differentiate findings by specific LLM models, which may potentially limit the generalizability of the results to different AI systems.

Construct Validity. The constructs examined in our analysis (e.g., engagement or performance) were not operationalized identically across the reviewed studies. Although we maintained fidelity to each paper's original terminology, we acknowledge that potential discrepancies in those papers may have impacted our synthesis.

Conclusion Validity. The reliability of our conclusions depends fundamentally on the methodological quality and rigor of the studies included in our review. In other words, limitations or biases in the original research can propagate into the aggregated findings. To mitigate this, we implemented stringent quality assessment criteria and restricted our analysis to peer-reviewed publications, but we acknowledge that quality assessment cannot eliminate all issues in the primary literature.

6. Conclusion

This review synthesized 52 articles published in journals and conferences indexed in ACM, IEEE, and ScienceDirect between 2021 and mid-June 2025 to examine practices and trends in student-facing, LLM-driven tools for undergraduate computing education. The reviewed studies demonstrated that these tools are increasingly utilizing pedagogical frameworks, employ diverse assessments of student-LLM interaction, and leverage multi-modal data to foster engagement and support learning. However, some concerns remain, particularly regarding the limited scaffolding, potential over-reliance on AI-generated solutions, and insufficient development of students' meta-cognitive and problem-solving skills.

Overall, while LLM-driven tools show promise in enhancing engagement and supporting instruction, the field requires further research to establish their effectiveness and inform effective implementation. Specifically, there is a critical need for empirical evidence from large-scale deployments, longitudinal studies examining long-term impacts on teaching and learning, and the development or adoption of pedagogical and technical frameworks suitable for this new context. Furthermore, the field must prioritize creating comprehensive design frameworks that guide the development of LLM tools for specific domains, learners, and educational contexts. These frameworks should particularly emphasize the creation of safe, reliable, and ethical tools that genuinely prioritize student learning outcomes over technological capabilities alone.

CRediT authorship contribution statement

Author 1: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Data curation; Visualization; Writing – Original Draft; Writing – Review & Editing; Project administration. **Author 2:** Investigation; Data curation; Validation; Formal analysis; Writing – Original Draft; Writing – Review & Editing. **Author 3:** Conceptualization; Methodology; Investigation; Data curation. Writing – Review & Editing. **Author 4:**

Methodology; Data curation; Validation; Visualization; Supervision; Formal analysis; Writing – Original Draft; Writing – Review & Editing. **Author 5:** Investigation; Data curation; Validation; Writing – Original Draft. **Author 6:** Investigation; Data curation; Validation. **Author 7:** Writing – Review & Editing. **Author 8:** Supervision; Funding acquisition; Conceptualization; Project administration; Writing – Original Draft; Writing – Review & Editing.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used LLMs by OpenAI, Google, Anthropic, Ollama, and Adobe in order to improve language, readability, and organization/flow of this article. After using these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Complete List of Papers

Table A.3: Data extraction categories

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
1	(Ahmed et al., 2025)	Feasibility Study of Augmenting Teaching Assistants with AI for CS1 Programming Feedback	Early-stage	CS1	India	Explore AI-generated feedback for human teaching assistants in CS1 programming exercises and student perceptions of AI-augmented TA responses	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Fact-Checker/Validator	Survey, Tool Response Evaluation	Academic Performance, Perception and Experience, Learning Behavior
2	(Alario-Hoyos et al., 2024)	Tailoring Your Code Companion: Leveraging LLMs and RAG to Develop a Chatbot to Support Students in a Programming Course	Early-stage	Systems Programming	Spain	Support first-year engineering students in a Java programming course with context-specific answers	Web-based UI	RAG, Prompt Engineering , Multi-Stage Pipeline, Guardrails, Prompt Chaining, Multimodal	Interaction logs , LLM Chat History	Tool Usage, Technical Performance, Perception and Experience
3	(Birillo et al., 2024)	One Step at a Time: Combining LLMs and Static Analysis to Generate Next-Step Hints for Programming Tasks	Early-stage	introduction to kotlin	Unclear	Combine LLMs with static analysis to provide textual and code hints for programming tasks	IDE-integrated Agent	Prompt Engineering, Multi-Stage Pipeline, Prompt Chaining, Fact-Checker/Validator	Interaction logs	Perception and Experience, Tool Usage, Technical Performance
4	(Crandall et al., 2023)	Generative Pre-Trained Transformer (GPT) Models as a Code Review Feedback Tool in Computer Science Programs	Early-stage	2nd-semester CS course	USA	Provide timely, automated code reviews for undergraduate CS students	GitHub Actions Interface	Prompt Engineering, Multi-Stage Pipeline, Guardrails	Survey	Perception and Experience
5	(del Carpio Gutierrez et al., 2024)	Automating Personalized Parsons Problems with Customized Contexts and Concepts	Early-stage	CS1	New Zealand	Generate personalized Parsons problems with customizable contexts for novice programmers.	Web-based UI	Prompt Engineering, Multi-Stage Pipeline	Interaction logs , Survey	Tool Usage, Perception and Experience
6	(Denny et al., 2024b)	Desirable Characteristics for AI Teaching Assistants in Programming Education	Early-stage	Introductory programming	New Zealand	Investigate student views on digital TAs and compare their preferences for automated assistance and human tutors	Web-based UI	Prompt Engineering, Guardrails, Multi-Stage Pipeline, Prompt Chaining	Interaction logs , Tool Response Evaluation , Survey	Perception and Experience, Tool Usage

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
7	(Denny et al., 2024a)	Prompt Problems: A New Programming Exercise for the Generative AI Era	Early-stage	Python-based courses CS1 and CS2	New Zealand	Help students learn how to write effective prompts for AI code generators	Web-based UI	Prompt Engineering	Survey, Interaction logs	Tool Usage, Perception and Experience
8	(Fan et al., 2025)	Software Engineering Educational Experience in Building an Intelligent Tutoring System	Early-stage	CS1	Singapore	Provide automated, real-time feedback for novice students	Learning platform-integrated Agent	Prompt Engineering, Multi-Stage Pipeline, Guardrails	Survey, Student Performance Metrics	Academic Performance, Perception and Experience
9	(Gabbay and Cohen, 2024)	Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming	Early-stage	Introductory python	Israel	Integrate LLM-generated feedback into Automated Test-based Feedback (ATF) in a MOOC to provide tailored support for learners in code assignments	Web-based UI	Prompt Engineering, Fact-Checker/Validator	Interaction logs , Survey, Tool Response Evaluation	Technical Performance, Perception and Experience
10	(Hassan et al., 2025)	On Teaching Novices Computational Thinking by Utilizing Large Language Models Within Assessments	Early-stage	Introductory programming	USA	Enhance novice programmers' computational thinking skills through scalable feedback and integrated assessments	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Guardrails, Multimodal	Interviews	Perception and Experience, Learning Behavior
11	(Huo et al., 2024)	Accelerating Learning with AI: Improving Students' Capability to Receive and Build Automated Feedback for Programming Courses	Early-stage	Programming 2	Australia	Explore a personalized programming feedback system using a fine-tuned LLM and develop a visualized interaction tool to help students interpret code and optimize prompts for debugging	Learning platform-integrated Agent	Prompt Engineering , In-context Learning, Fine tuning, Prompt Chaining	LLM Chat History, Survey, Discussion Board Analysis, Student Performance Metrics	Tool Usage, Academic Performance, Technical Performance, Perception and Experience
12	(Hussein et al., 2024)	Integrating Personalized AI-Assisted Instruction Into Remote Laboratories: Enhancing Engineering Education with OpenAI's GPT Models	Late-stage	Advanced digital design course	USA	Provide personalized AI-assisted instruction in remote laboratories, promoting independent learning and critical thinking	Web-based UI, IDE-integrated Agent	Prompt Engineering, Guardrails	Survey	Perception and Experience

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
13	(Jury et al., 2024)	Evaluating LLM-generated Worked Examples in an Introductory Programming Course	Early-stage	Introductory CS Class	New Zealand	Generate interactive worked examples	Web-based UI	Prompt Engineering, Prompt Chaining, Multi-Stage Pipeline, Few Shot Prompting, Guardrails	Interaction logs , Survey, Tool Response Evaluation	Tool Usage, Perception and Experience, Technical Performance
14	(Kazemitabaar et al., 2024)	CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs	Early-stage	C programming	Canada	Meet the needs of both students and educators by being helpful and technically correct, while not directly revealing code solutions.	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Few Shot Prompting, Guardrails, Prompt Chaining	Interaction logs , Survey, Interviews	Tool Usage, Technical Performance, Perception and Experience
15	(Kumar et al., 2024)	Supporting Self-Reflection at Scale with Large Language Models: Insights from Randomized Field Experiments in Classrooms	Mixed	Introduction to Databases and Intro to Computer Programming	Canada	Facilitate students' self-reflection with the goal of enabling interactive reflection at scale	Web-based UI	Prompt Engineering	Survey, LLM Chat History, Student Performance Metrics	Academic Performance, Perception and Experience, Motivation/Attitude, Tool Usage
16	(Kuramitsu et al., 2023)	KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education	Mixed	Algorithm and Data Science	Japan	Provide AI-based learning assistance to students who encounter programming challenges, such as unresolved errors, unknown terms, unexplained code, and code modification	IDE-integrated Agent	Prompt Engineering, Guardrails, Prompt Chaining	Survey, Interaction logs , LLM Chat History	Tool Usage, Learning Behavior, Technical Performance
17	(Liffiton et al., 2024)	CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes	Early-stage	a first-year computer and data-science course	USA	Provide on-demand assistance to programming students without directly revealing solutions, mitigating the over-reliance trap	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Guardrails, Prompt Chaining	Interaction logs, Survey	Perception and Experience, Tool Usage
18	(Liu et al., 2025a)	Improving AI in CS50: Leveraging Human Feedback for Better Learning	Early-stage	Introductory Python	Israel	Propose a continuous improvement process for educational AI systems, combining human evaluations, automated assessments, few-shot prompting and fine-tuning techniques to ensure pedagogically sound help	Web-based UI	Prompt Engineering, Fact-Checker/Validator	Interaction logs , Survey, Tool Response Evaluation	Technical Performance, Perception and Experience

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
19	(Liu et al., 2024b)	Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education	Early-stage	Intro to CS (CS50)	USA	Provide students with an AI tutor that acts as a personal expert by avoiding direct answers and adapting to course updates.	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Prompt Chaining, Fine tuning, Guardrails, Fact-Checker/Validator, Few Shot Prompting	Survey, Interaction logs , Tool Response Evaluation	Perception and Experience, Tool Usage, Technical Performance
20	(Lui et al., 2024)	GPTutor: A Generative AI-powered Intelligent Tutoring System to Support Interactive Learning with Knowledge-Grounded Question Answering	Mixed	Two software engineering courses	Hong Kong	Enhance interactive learning, and student engagement, through course material-grounded question-answering	Web-based UI	RAG, Prompt Engineering, Multi-Stage Pipeline, Guardrails	Survey, Interaction logs	Academic Performance, Perception and Experience, Tool Usage
21	(Lyu et al., 2024)	Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study	Early-stage	Programming in Python	USA	Explore the effectiveness of an LLM-powered tool in enhancing learning outcomes and engagement in introductory CS courses	Web-based UI	Prompt Engineering	Interaction logs , Student Performance Metrics , LLM Chat History	Academic Performance, Motivation/Attitude, Tool Usage
22	(Ma et al., 2024)	Integrating AI Tutors in a Programming Course	Early-stage	Introductory programming (Python)	USA	Provide programming students with indirect support and nudge them towards possible next steps relative to their questions	Web-based UI	RAG, Prompt Engineering, Few Shot Prompting, Guardrails	LLM Chat History	Tool Usage, Perception and Experience, Technical Performance, Academic Performance
23	(Menezes et al., 2024)	AI-Grading Standup Updates to Improve Project-Based Learning Outcomes	Unspecified	Code-Day Labs (College-level program for software development)	USA	Collect and share student standup updates in team Slack channels, propose a grading rubric, and train an AI to automate grading.	Chat App Integration (Slack, Discord)	Prompt Engineering, Fine tuning	Survey, Tool Response Evaluation , Student Performance Metrics	Technical Performance, Academic Performance, Perception and Experience

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
24	(Nelson et al., 2025)	SENSAI: Large Language Models as Applied Cybersecurity Tutors	Unspecified	Cybersecurity	USA	Provide personalized learning support in cybersecurity education by analyzing a student's working context, including their active terminals and edited files	Web-based UI	Prompt Engineering, Fine tuning, Multi-Stage Pipeline	Survey, Interaction logs	Perception and Experience, Learning Behavior
25	(Neumann et al., 2025)	An LLM-Driven Chatbot in Higher Education for Databases and Information Systems	Unspecified	Databases and Information Systems	Germany	Integrate LLMs into LMSs like Moodle to support self-regulated learning (SRL) and help-seeking behavior	Learning platform-integrated Agent	RAG, Prompt Engineering, Multi-Stage Pipeline, Fact-Checker/Validator, Prompt Chaining	Survey, LLM Chat History, Tool Response Evaluation	Perception and Experience, Motivation/Attitude, Technical Performance
26	(Neyem et al., 2024b)	Exploring the Impact of Generative AI for StandUp Report Recommendations in Software Capstone Project Development	Late-stage	Capstone software engineering course	Chile	Explore the impact of generative AI on improving quality and effectiveness of StandUp Reports in software engineering capstone courses	Chat App Integration (Slack, Discord)	Prompt Engineering, Multi-Stage Pipeline	Submission artifacts , Survey, Tool Response Evaluation	Perception and Experience, Technical Performance
27	(Neyem et al., 2024a)	Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development	Late-stage	Software development	Chile	Enhance LLM's quality and relevance through integration with a capstone course database	Web-based UI	RAG, Prompt Engineering, Multi-Stage Pipeline	Survey, Tool Response Evaluation , Interaction logs	Perception and Experience, Technical Performance
28	(Pankiewicz and Baker, 2024)	Navigating Compiler Errors with AI Assistance - A Study of GPT Hints in an Introductory Programming Course	Early-stage	Introductry programming	Poland	Explore the effectiveness of AI-generated hints in helping introductory programming students navigate compiler errors	Learning platform-integrated Agent, Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Few Shot Prompting	Survey, Tool Response Evaluation, Submission artifacts	Perception and Experience, Learning Behavior, Academic Performance
29	(P?durean et al., 2025)	BugSpotter: Automated Generation of Code Debugging Exercises	Early-stage	Introductory C programming	New Zealand	Produce buggy code based on a problem description and validate the added bugs using a test suite	Web-based UI	Prompt Engineering, Multi-Stage Pipeline	Student Performance Metrics	Technical Performance, Academic Performance

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
30	(Pua et al., 2025)	CustomAIzEd: Bridging Interdisciplinary Gaps in AI Education with Customized Content Using LLMs	Early-stage	Introductory AI	Singapore	Create personalized assignments and quizzes focused on AI concepts	Web-based UI	Prompt Engineering, Few Shot Prompting, Prompt Chaining, Guardrails, Fact-Checker/Validator	Interviews	Perception and Experience
31	(Qadir, 2025)	Generative AI in Undergraduate Classrooms: Lessons from Implementing a Customized GPT Chatbot for Learning Enhancement	Mixed	Data and Computer Communications Networks I (DCCN-I) Course and Computer Ethics Course	Qatar	Study the integration of a customized AI chatbot into undergraduate courses with majorly non-native English speakers	Web-based UI	RAG, Prompt Engineering, Guardrails	Survey	Perception and Experience
32	(Renzella et al., 2025)	Compiler-Integrated, Conversational AI for Debugging CS1 Programs	Early-stage	CS1	Australia	Present a compiler-integrated AI tool designed to enhance debugging support for CS1 programming students	Web-based UI, Command-line Interface (CLI)	Prompt Engineering, Guardrails, Multi-Stage Pipeline, Prompt Chaining	Interaction logs	Tool Usage, Perception and Experience
33	(Riazi and Rooshenas, 2025)	LLM-Driven Feedback for Enhancing Conceptual Design Learning in Database Systems Courses	Unspecified	Database Systems course	USA	Provide targeted feedback on conceptual design tasks in Database Systems courses	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Prompt Chaining	Survey, Tool Response Evaluation	Perception and Experience, Technical Performance
34	(Rivera et al., 2024)	Iterative Student Program Planning using Transformer-Driven Feedback	Early-stage	CS1	Austria	Utilize LLMs to generate code based on students' plans, and evaluate the code against expert-defined test suites.	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Guardrails, Fact-Checker/Validator	Survey	Perception and Experience, Tool Usage, Learning Behavior, Technical Performance
35	(Rogers et al., 2025)	Playing Dumb to Get Smart: Creating and Evaluating an LLM-based Teachable Agent within University Computer Science Classes	Early-stage	Introductory MATLAB	USA	Present the iterative design and evaluation of an LLM-based teachable agent that facilitates learning-by-teaching in CS courses	Web-based UI	Prompt Engineering, Guardrails	Survey, Interviews, LLM Chat History	Perception and Experience, Motivation/Attitude, Academic Performance

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
36	(Sheese et al., 2024)	Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant	Early-stage	introductory computer- and data-science course	USA	Study students' use of LLMs in practice and their help-seeking patterns	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Guardrails, Prompt Chaining	Interaction logs , Student Performance Metrics	Learning Behavior, Tool Usage, Academic Performance
37	(Shin et al., 2024)	Understanding Optimal Interactions Between Students and A Chatbot During A Programming Task	Early-stage	CS2	USA	Study students' effective help-seeking behaviors and the correlation between problem-solving strategies and learning outcomes.	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Guardrails, Prompt Chaining	Submission artifacts , Interaction logs	Tool Usage, Learning Behavior
38	(Smith et al., 2024)	Prompting for Comprehension: Exploring the Intersection of Explain in Plain English Questions and Prompt Writing	Early-stage	Introductory programming	New Zealand	Combine "Explain in Plain English" questions and LLMs to enhance student learning and feedback	Web-based UI	Prompt Engineering, Guardrails, Multi-Stage Pipeline, Fact-Checker/Validator	Student Performance Metrics	Perception and Experience, Academic Performance, Learning Behavior
39	(Tan et al., 2025a)	Explore the Impact of Avatar Representations in AI Chatbot Tutors on Learning Experiences	Unspecified	Design thinking course	Singapore	Explore how different AI chatbot avatar representations, e.g., visual and auditory elements, impact student engagement, prompting behaviors, and perceived response quality	Web-based UI	RAG, Prompt Engineering, Multimodal, Multi-Stage Pipeline	Survey, Interviews	Perception and Experience
40	(Taylor et al., 2024)	dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models	Early-stage	CS1 and CS2	Australia	Generate context-aware, novice-focused error explanations in programming environment (terminal)	Command-line Interface (CLI)	Prompt Engineering, Guardrails	Interaction logs , Tool Response Evaluation	Tool Usage, Technical Performance
41	(Yang et al., 2024b)	Debugging with an AI Tutor: Investigating Novice Help-seeking Behaviors and Perceived Learning	Early-stage	Introductory programming	USA	Explore how a pedagogically-designed chatbot supports students' debugging in an introductory programming course	Web-based UI	RAG, Prompt Engineering, Multi-Stage Pipeline, Guardrails	Interviews	Learning Behavior, Perception and Experience, Tool Usage
42	(Yang et al., 2024a)	Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education	Early-stage	Python programming	Taiwan	Support novice Python programmers	Web-based UI	RAG, Prompt Engineering, Guardrails, Fact-Checker/Validator	Student Performance Metrics , Interaction logs	Tool Usage, Academic Performance

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
43	(Zönnchen et al., 2025)	Exploring the Role of Large Language Models as Artificial Tutors	Early-stage	Computational Thinking course	Germany	Investigate students' perceptions, engagement, and use of the CS50 Duck in Computational Thinking course	Web-based UI	Prompt Engineering	Survey, Submission artifacts	Perception and Experience, Learning Behavior, Tool Usage, Motivation/Attitude
Supplementary Papers										
44	(Bassner et al., 2024)	Iris: An AI-Driven Virtual Tutor for Computer Science Education	Early-stage	Programming fundamentals (CS1)	Germany	Integrate the LLM into the interactive learning platform to guide students with hints and counter-questions, without revealing complete solutions	Learning platform-integrated Agent	Prompt Engineering, Multi-Stage Pipeline, Few Shot Prompting, Prompt Chaining, Guardrails	Survey	Perception and Experience
45	(Frankford et al., 2024)	AI-Tutoring in Software Engineering Education	Early-stage	Introduction to programming	Austria	Evaluate LLMs' integration in Automated Programming Assessment Systems (APASs)	Learning platform-integrated Agent	Prompt Engineering	Interaction logs , Survey	Perception and Experience, Tool Usage
46	(Gao et al., 2025)	Fine-Tuned Large Language Model for Visualization System: A Study on Self-Regulated Learning in Education	Unspecified	Unspecified	China	Create an intelligent visualization system to support beginners' self-regulated learning	Web-based UI	Fine tuning, Multi-Stage Pipeline, Prompt Engineering , Prompt Chaining	Survey, Interaction logs	Perception and Experience, Tool Usage, Technical Performance
47	(Goddard et al., 2024)	A Chatbot Won't Judge Me: An Exploratory Study of Self-disclosing Chatbots in Introductory Computer Science Classes	Early-stage	Introductory CS	Canada	Foster a more inclusive CS culture and normalize challenges in learning through self-disclosing chatbot's imaginary struggles with course materials	Chat App Integration (Slack, Discord)	Multi-Stage Pipeline, Prompt Engineering	Survey, Interviews , LLM Chat History	Perception and Experience, Tool Usage
48	(Hou et al., 2024)	CodeTailor: LLM-Powered Personalized Parsons Puzzles for Engaging Support While Learning Programming	Early-stage	Introductory python	USA	Generate correct code solutions and encourage student engagement and active learning	Web-based UI	RAG, Prompt Engineering , Few Shot Prompting, Multi-Stage Pipeline, Guardrails, Fact-Checker/Validator, Prompt Chaining	Survey, Interaction logs , Interviews	Perception and Experience, Academic Performance

Continued on next page

Continued from previous page

#	Author(s)	Title	Grade	Subject	Location	Research Purposes	UI Modality	Technical Approaches	Data Collection Methods	Evaluation Focus
49	(Jacobs and Jaschke, 2024)	Leveraging Lecture Content for Improved Feedback: Explorations with GPT-4 and Retrieval Augmented Generation	Early-stage	Introductory programming	Germany	Encourage students' independent problem solving without revealing answers	Web-based UI	RAG, Prompt Engineering, Multi-Stage Pipeline, Few Shot Prompting, Guardrails, Prompt Chaining	Survey	Perception and Experience, Tool Usage, Technical Performance
50	(Jin et al., 2024)	Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education	Early-stage	Introductory algorithm learning	Korea	Investigate LLMs as teachable agents for "learning by teaching (LBT)" by proposing a pipeline that restrains LLMs' knowledge and makes them initiate 'why' and 'how' questions for effective knowledge-building	Web-based UI	Prompt Engineering, Multi-Stage Pipeline, Few Shot Prompting, Guardrails, Prompt Chaining, Fact-Checker/Validator, Fine tuning	Survey, LLM Chat History	Perception and Experience, Tool Usage, Technical Performance
51	(Wang, 2023)	Efficient generation of text feedback in object-oriented programming education using cached performer revision	Unspecified	Object-oriented programming (OOP)	Taiwan	Propose a cached Performer model to address Transformer architecture's speed and scalability issues when generating text feedback for object-oriented programming education	Web-based UI	Cache augmented prompt pipeline, Customized Performer (Transformer)	Survey, Tool Response Evaluation	Perception and Experience, Technical Performance
52	(Woodrow et al., 2024)	AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course	Early-stage	CS1	Worldwide	Provide "style feedback" to help introductory programming students write elegant, reusable, and comprehensible code	IDE-integrated Agent	Multi-Stage Pipeline, Prompt Engineering, Guardrails	Interaction logs , Submission artifacts	Academic Performance, Learning Behavior, Tool Usage

Appendix B. Pedagogical Approaches Utilized in the Papers

Table B.4: Papers with Pedagogical Approaches

#	Author(s)	Title	Scaff. General	Scaff. Feedback	Scaff. Socratic	Scaff. Metacognitive	Scaff. Personalization	Problem-Based Learning	Dir. Teach Concept	Dir. Write Code	Affective/Motivational	Learn by Teach	Course-Specific	Unrestricted
1	(Ahmed et al., 2025)	Feasibility Study of Augmenting Teaching Assistants with AI for CS1 Programming Feedback	√	√				√	√					
2	(Alario-Hoyos et al., 2024)	Tailoring Your Code Companion: Leveraging LLMs and RAG to Develop a Chatbot to Support Students in a Programming Course											√	
3	(Birillo et al., 2024)	One Step at a Time: Combining LLMs and Static Analysis to Generate Next-Step Hints for Programming Tasks	√	√			√	√		√				
4	(Crandall et al., 2023)	Generative Pre-Trained Transformer (GPT) Models as a Code Review Feedback Tool in Computer Science Programs	√	√	√									
5	(del Carpio Gutierrez et al., 2024)	Automating Personalized Parsons Problems with Customized Contexts and Concepts	√	√	√			√						
6	(Denny et al., 2024b)	Desirable Characteristics for AI Teaching Assistants in Programming Education	√		√						√			
7	(Denny et al., 2024a)	Prompt Problems: A New Programming Exercise for the Generative AI Era								√				
8	(Fan et al., 2025)	Software Engineering Educational Experience in Building an Intelligent Tutoring System	√	√										
9	(Gabbay and Cohen, 2024)	Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming	√	√					√					
10	(Hassan et al., 2025)	On Teaching Novices Computational Thinking by Utilizing Large Language Models Within Assessments	√			√			√	√				
11	(Huo et al., 2024)	Accelerating Learning with AI: Improving Students' Capability to Receive and Build Automated Feedback for Programming Courses	√						√				√	

Continued on next page

Continued from previous page

#	Author(s)	Title	Scaff. Gen.	Scaff. Feed.	Scaff. Socr.	Scaff. Meta.	Scaff. Pers.	Prob.-Based	Dir. Teach	Dir. Write	Affect./Motiv.	Learn by Teach	Course-Spec.	Unrestricted
12	(Hussein et al., 2024)	Integrating Personalized AI-Assisted Instruction Into Remote Laboratories: Enhancing Engineering Education with OpenAI's GPT Models	√		√								√	
13	(Jury et al., 2024)	Evaluating LLM-generated Worked Examples in an Introductory Programming Course	√											
14	(Kazemitabaar et al., 2024)	CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs	√	√				√	√	√				
15	(Kumar et al., 2024)	Supporting Self-Reflection at Scale with Large Language Models: Insights from Randomized Field Experiments in Classrooms				√								
16	(Kuramitsu et al., 2023)	KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education	√						√	√				
17	(Liffiton et al., 2024)	CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes	√		√								√	
18	(Liu et al., 2025a)	Improving AI in CS50: Leveraging Human Feedback for Better Learning	√		√						√			
19	(Liu et al., 2024b)	Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education	√		√			√					√	
20	(Lui et al., 2024)	GPTutor: A Generative AI-powered Intelligent Tutoring System to Support Interactive Learning with Knowledge-Grounded Question Answering	√										√	
21	(Lyu et al., 2024)	Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study												√
22	(Ma et al., 2024)	Integrating AI Tutors in a Programming Course	√	√	√		√						√	
23	(Menezes et al., 2024)	AI-Grading Standup Updates to Improve Project-Based Learning Outcomes	√	√		√		√						
24	(Nelson et al., 2025)	SENSAI: Large Language Models as Applied Cybersecurity Tutors	√		√								√	

Continued on next page

Continued from previous page

#	Author(s)	Title	Scaff. Gen.	Scaff. Feed.	Scaff. Socr.	Scaff. Meta.	Scaff. Pers.	Prob.-Based	Dir. Teach	Dir. Write	Affect./Motiv.	Learn by Teach	Course-Spec.	Unrestricted
25	(Neumann et al., 2025)	An LLM-Driven Chatbot in Higher Education for Databases and Information Systems							√				√	
26	(Neyem et al., 2024b)	Exploring the Impact of Generative AI for StandUp Report Recommendations in Software Capstone Project Development	√	√				√						
27	(Neyem et al., 2024a)	Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development	√					√					√	
28	(Pankiewicz and Baker, 2024)	Navigating Compiler Errors with AI Assistance – A Study of GPT Hints in an Introductory Programming Course	√											
29	(P?durean et al., 2025)	BugSpotter: Automated Generation of Code Debugging Exercises	√											
30	(Pua et al., 2025)	CustomAIzEd: Bridging Interdisciplinary Gaps in AI Education with Customized Content Using LLMs	√											
31	(Qadir, 2025)	Generative AI in Undergraduate Classrooms: Lessons from Implementing a Customized GPT Chatbot for Learning Enhancement	√										√	
32	(Renzella et al., 2025)	Compiler-Integrated, Conversational AI for Debugging CS1 Programs	√	√					√					
33	(Riazi and Rooshenas, 2025)	LLM-Driven Feedback for Enhancing Conceptual Design Learning in Database Systems Courses	√	√				√	√				√	
34	(Rivera et al., 2024)	Iterative Student Program Planning using Transformer-Driven Feedback	√			√		√		√				
35	(Rogers et al., 2025)	Playing Dumb to Get Smart: Creating and Evaluating an LLM-based Teachable Agent within University Computer Science Classes										√		
36	(Sheese et al., 2024)	Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant	√	√				√						
37	(Shin et al., 2024)	Understanding Optimal Interactions Between Students and a Chatbot During a Programming Task	√		√									√

Continued on next page

Continued from previous page

#	Author(s)	Title	Scaff. Gen.	Scaff. Feed.	Scaff. Socr.	Scaff. Meta.	Scaff. Pers.	Prob.-Based	Dir. Teach	Dir. Write	Affect./Motiv.	Learn by Teach	Course-Spec.	Unrestricted
38	(Smith et al., 2024)	Prompting for Comprehension: Exploring the Intersection of "Explain in Plain English" Questions and Prompt Writing	✓	✓								✓		
39	(Tan et al., 2025a)	Explore the Impact of Avatar Representations in AI Chatbot Tutors on Learning Experiences						✓						✓
40	(Taylor et al., 2024)	dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models	✓	✓										
41	(Yang et al., 2024b)	Debugging with an AI Tutor: Investigating Novice Help-Seeking Behaviors and Perceived Learning	✓		✓			✓						
42	(Yang et al., 2024a)	Enhancing Python Learning with PyTutor: Efficacy of a ChatGPT-Based Intelligent Tutoring System in Programming Education	✓			✓	✓		✓					
43	(Zönnchen et al., 2025)	Exploring the Role of Large Language Models as Artificial Tutors	✓		✓			✓						
Supplementary Papers														
44	(Bassner et al., 2024)	Iris: An AI-Driven Virtual Tutor for Computer Science Education	✓	✓	✓			✓					✓	
45	(Frankford et al., 2024)	AI-Tutoring in Software Engineering Education	✓	✓	✓			✓						
46	(Gao et al., 2025)	Fine-Tuned Large Language Model for Visualization System: A Study on Self-Regulated Learning in Education	✓			✓	✓		✓				✓	
47	(Goddard et al., 2024)	A Chatbot Won't Judge Me: An Exploratory Study of Self-Disclosing Chatbots in Introductory Computer Science Classes									✓			
48	(Hou et al., 2024)	CodeTailor: LLM-Powered Personalized Parsons Puzzles for Engaging Support While Learning Programming	✓				✓	✓						
49	(Jacobs and Jaschke, 2024)	Leveraging Lecture Content for Improved Feedback: Explorations with GPT-4 and Retrieval-Augmented Generation	✓	✓			✓						✓	
50	(Jin et al., 2024)	Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education	✓			✓						✓		

Continued on next page

Continued from previous page

#	Author(s)	Title	Scaff. Gen.	Scaff. Feed.	Scaff. Socr.	Scaff. Meta.	Scaff. Pers.	Prob.-Based	Dir. Teach	Dir. Write	Affect./Motiv.	Learn by Teach	Course-Spec.	Unrestricted
51	(Wang, 2023)	Efficient Generation of Text Feedback in Object-Oriented Programming Education Using Cached Performer Revision						√	√					
52	(Woodrow et al., 2024)	AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course	√											

References

, 2025. Qualitative Studies Checklist - CASP. URL: `https://casp-uk.net/casp-tools-checklists/qualitative-studies-checklist/`.

Ahmed, U.Z., Sahai, S., Leong, B., Karkare, A., 2025. Feasibility Study of Augmenting Teaching Assistants with AI for CS1 Programming Feedback, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 11–17. URL: `https://doi.org/10.1145/3641554.3701972`, doi:10.1145/3641554.3701972.

Alanazi, M., Soh, B., Samra, H., Li, A., 2025. The Influence of Artificial Intelligence Tools on Learning Outcomes in Computer Programming: A Systematic Review and Meta-Analysis. Computers 14, 185. URL: `https://www.mdpi.com/2073-431X/14/5/185`, doi:10.3390/computers14050185. number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

Alario-Hoyos, C., Kemcha, R., Kloos, C.D., Callejo, P., Estévez-Ayres, I., Santín-Cristóbal, D., Cruz-Argudo, F., López-Sánchez, J.L., 2024. Tailoring Your Code Companion: Leveraging LLMs and RAG to Develop a Chatbot to Support Students in a Programming Course, in: 2024 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), pp. 1–8. URL: `https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10834365`, doi:10.1109/TALE62452.2024.10834365.

Allen, V.L., Feldman, R.S., 1973. Learning through tutoring: Low-achieving children as tutors. The Journal of Experimental Education 42, 1–5.

Álvarez Ariza, J., Benitez Restrepo, M., Hernández Hernández, C., 2025. Generative AI in Engineering and Computing Education: A Scoping Review of Empirical Studies and Educational Practices. IEEE Access 13, 30789–30810. URL: `https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10883995`, doi:10.1109/ACCESS.2025.3541424.

Arroyo, I., Cooper, D.G., Burleson, W., Woolf, B.P., Muldner, K., Christopherson, R., 2009. Emotion sensors go to school, in: Artificial intelligence in education, Ios Press. pp. 17–24.

Barnes, T., Burriss, S., Danish, J., Finkelstein, S., Humburg, M., Limke, A., Molvig, O., Reichert, H., 2024. Toward ethical and just ai in education research. Community for Advancing Discovery Research in Education (CADRE) .

Bassner, P., Frankford, E., Krusche, S., 2024. Iris: An AI-Driven Virtual Tutor for Computer Science Education, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 394–400. URL: `https://dl.acm.org/doi/10.1145/3649217.3653543`, doi:10.1145/3649217.3653543.

Benhayoun, L., Lang, D., 2021. Does higher education properly prepare graduates for the growing artificial intelligence market? gaps' identification using text mining. Human Systems Management 40, 639–651.

Bernabei, M., Colabianchi, S., Falegnami, A., Costantino, F., 2023. Students' use of large language models in engineering education: A case study on technology acceptance, perceptions, efficacy, and detection chances. Computers and Education: Artificial Intelligence 5, 100172. URL: `https://www.sciencedirect.com/science/article/pii/S2666920X23000516`, doi:10.1016/j.caeai.2023.100172.

Bernard, R.M., Borokhovski, E., Schmid, R.F., Waddington, D.I., Pickup, D.I., 2019. Twenty-first century adaptive teaching and individualized learning operationalized as specific blends of student-centered instructional events: A systematic review and meta-analysis. Campbell Systematic Reviews 15, e1017.

Berssanette, J.H., de Francisco, A.C., 2021. Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review. IEEE Transactions on Education 65, 440–449.

Birillo, A., Artser, E., Potriasaeva, A., Vlasov, I., Dzialets, K., Golubev, Y., Gerasimov, I., Keuning, H., Bryksin, T., 2024. One Step at a Time: Combining LLMs and Static Analysis to Generate Next-Step Hints for Programming Tasks, in: Proceedings of the 24th Koli Calling International Conference on Computing Education Research, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3699538.3699556, doi:10.1145/3699538.3699556.

Biswas, G., Segedy, J.R., Bunchongchit, K., 2016. From design to implementation to practice a learning by teaching system: Betty's brain. International Journal of Artificial Intelligence in Education 26, 350–364.

Boud, D., 2012. Developing student autonomy in learning. Routledge.

Braun, V., , Clarke, V., 2019. Reflecting on reflexive thematic analysis. Qualitative Research in Sport, Exercise and Health 11, 589–597. URL: https://doi.org/10.1080/2159676X.2019.1628806, doi:10.1080/2159676X.2019.1628806. publisher: Routledge _eprint: https://doi.org/10.1080/2159676X.2019.1628806.

Braun, V., Clarke, V., 2024. Supporting best practice in reflexive thematic analysis reporting in Palliative Medicine: A review of published research and introduction to the Reflexive Thematic Analysis Reporting Guidelines (RTARG). Palliative Medicine 38, 608. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11157981/, doi:10.1177/02692163241234800.

Cambaz, D., Zhang, X., 2024. Use of AI-driven Code Generation Models in Teaching and Learning Programming: a Systematic Literature Review, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 172–178. URL: https://doi.org/10.1145/3626252.3630958, doi:10.1145/3626252.3630958.

del Carpio Gutierrez, A., Denny, P., Luxton-Reilly, A., 2024. Automating Personalized Parsons Problems with Customized Contexts and Concepts, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 688–694. URL: https://doi.org/10.1145/3649217.3653568, doi:10.1145/3649217.3653568.

Chi, M.T., Wylie, R., 2014. The icap framework: Linking cognitive engagement to active learning outcomes. Educational psychologist 49, 219–243.

Covidence, . Covidence - Better systematic review management. URL: https://www.covidence.org/.

Crandall, A.S., Sprint, G., Fischer, B., 2023. Generative Pre-Trained Transformer (GPT) Models as a Code Review Feedback Tool in Computer Science Programs. J. Comput. Sci. Coll. 39, 38–47.

Decker, A., McGill, M.M., 2019. A topical review of evaluation instruments for computing education, in: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp. 558–564.

Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B.A., Reeves, B.N., 2024a. Prompt Problems: A New Programming Exercise for the Generative AI Era, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 296–302. URL: https://doi.org/10.1145/3626252.3630909, doi:10.1145/3626252.3630909.

Denny, P., MacNeil, S., Savelka, J., Porter, L., Luxton-Reilly, A., 2024b. Desirable Characteristics for AI Teaching Assistants in Programming Education, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 408–414. URL: https://doi.org/10.1145/3649217.3653574, doi:10.1145/3649217.3653574.

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., 2024c. Computing Education in the Era of Generative AI. Commun. ACM 67, 56–67. URL: https://dl.acm.org/doi/10.1145/3624720, doi:10.1145/3624720.

Fan, Z., Noller, Y., Dandekar, A., Roychoudhury, A., 2025. Software Engineering Educational Experience in Building an Intelligent Tutoring System, in: 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T), pp. 75–86. URL: https://ieeexplore.ieee.org/document/11024295, doi:10.1109/CSEET66350.2025.00015. iSSN: 2832-7578.

Feng, T.H., Luxton-Reilly, A., Wünsche, B.C., Denny, P., 2025. From Automation to Cognition: Redefining the Roles of Educators and Generative AI in Computing Education, in: Proceedings of the 27th Australasian Computing Education Conference, Association for Computing Machinery, New York, NY, USA. pp. 164–171. URL: https://doi.org/10.1145/3716640.3716658, doi:10.1145/3716640.3716658.

Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E.A., Prather, J., Becker, B.A., 2023. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises, in: Proceedings of the 25th Australasian Computing Education Conference, Association for Computing Machinery, New York, NY, USA. pp. 97–104. URL: https://doi.org/10.1145/3576123.3576134, doi:10.1145/3576123.3576134.

Fiorella, L., Mayer, R.E., 2013. The relative benefits of learning by teaching and teaching expectancy. Contemporary Educational Psychology 38, 281–288. doi:10.1016/j.cedpsych.2013.06.001.

Frankford, E., Sauerwein, C., Bassner, P., Krusche, S., Breu, R., 2024. AI-Tutoring in Software Engineering Education, in: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, Association for Computing Machinery, New York, NY, USA. pp. 309–319. URL: https://dl.acm.org/doi/10.1145/3639474.3640061, doi:10.1145/3639474.3640061.

Gabbay, H., Cohen, A., 2024. Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, Association for Computing Machinery, New York, NY, USA. pp. 177–187. URL: https://doi.org/10.1145/3657604.3662040, doi:10.1145/3657604.3662040.

Gao, L., Lu, J., Shao, Z., Lin, Z., Yue, S., Leong, C., Sun, Y., Zauner, R.J., Wei, Z., Chen, S., 2025. Fine-Tuned Large Language Model for Visualization System: A Study on Self-Regulated Learning in Education. IEEE Transactions on Visualization and Computer Graphics 31, 514–524. URL: https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10670435, doi:10.1109/TVCG.2024.3456145.

Goddard, Q., Moton, N., Hudson, J., He, H.A., 2024. A Chatbot Won't Judge Me: An Exploratory Study of Self-disclosing Chatbots in Introductory Computer Science Classes, in: Proceedings of the 26th Western Canadian Conference on Computing Education, Association for Computing Machinery, New York, NY, USA. pp. 1–7. URL: https://dl.acm.org/doi/10.1145/3660650.3660662, doi:10.1145/3660650.3660662.

Hassan, M., Chen, Y., Denny, P., Zilles, C., 2025. On Teaching Novices Computational Thinking by Utilizing Large Language Models Within Assessments, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 471–477. URL: https://doi.org/10.1145/3641554.3701906, doi:10.1145/3641554.3701906.

Haynes, C.C., Ericson, B.J., 2021. Problem-solving efficiency and cognitive load for adaptive parsons problems vs. writing the equivalent code, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–15.

Hooshyar, D., Kori, K., Pedaste, M., Bardone, E., 2019. The potential of open learner models to promote active thinking by enhancing self-regulated learning in online higher education learning environments. British Journal of Educational Technology 50, 2365–2386.

Hou, X., Wu, Z., Wang, X., Ericson, B.J., 2024. CodeTailor: LLM-Powered Personalized Parsons Puzzles for Engaging Support While Learning Programming, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, Association for Computing Machinery, New York, NY, USA. pp. 51–62. URL: https://dl.acm.org/doi/10.1145/3657604.3662032, doi:10.1145/3657604.3662032.

Huo, H., Ding, X., Guo, Z., Shen, S., Ye, D., Pham, O., Milne, D., Mathieson, L., Gardner, A., 2024. Accelerating Learning with AI: Improving Students' Capability to Receive and Build Automated Feedback for Programming Courses, in: 2024 World Engineering Education Forum - Global Engineering Deans Council (WEEF-GEDC), pp. 1–9. URL: https://ieeexplore.ieee.org/document/10854928, doi:10.1109/WEEF-GEDC63419.2024.10854928. iSSN: 2837-5025.

Hussein, R., Zhang, Z., Amarante, P., Hancock, N., Orduna, P., Rodriguez-Gil, L., 2024. Integrating Personalized AI-Assisted Instruction Into Remote Laboratories: Enhancing Engineering Education with OpenAI's GPT Models, in: 2024 IEEE Frontiers in Education Conference (FIE), pp. 1–7. URL: https://ieeexplore.ieee.org/document/10892918, doi:10.1109/FIE61694.2024.10892918. iSSN: 2377-634X.

Idowu, E., 2024. Personalized learning: Tailoring instruction to individual student needs .

Jacobs, S., Jaschke, S., 2024. Leveraging Lecture Content for Improved Feedback: Explorations with GPT-4 and Retrieval Augmented Generation, in: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T), pp. 1–5. URL: https://ieeexplore.ieee.org/document/10663001, doi:10.1109/CSEET62301.2024.10663001. iSSN: 2377-570X.

Jin, H., Lee, S., Shin, H., Kim, J., 2024. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education, in: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. pp. 1–28. URL: https://dl.acm.org/doi/10.1145/3613904.3642349, doi:10.1145/3613904.3642349.

Jury, B., Lorusso, A., Leinonen, J., Denny, P., Luxton-Reilly, A., 2024. Evaluating LLM-generated Worked Examples in an Introductory Programming Course, in: Proceedings of the 26th Australasian Computing Education Conference, Association for Computing Machinery, New York, NY, USA. pp. 77–86. URL: https://doi.org/10.1145/3636243.3636252, doi:10.1145/3636243.3636252.

Kanfer, R., Ackerman, P.L., 1989. Motivation and cognitive abilities: An integrative/aptitude-treatment interaction approach to skill acquisition. Journal of applied psychology 74, 657.

Kazemitabaar, M., Huang, O., Suh, S., Henley, A.Z., Grossman, T., 2025. Exploring the Design Space of Cognitive Engagement Techniques with AI-Generated Code for Enhanced Learning, in: Proceedings of the 30th International Conference on Intelligent User Interfaces, Association for Computing Machinery, New York, NY, USA. pp. 695–714. URL: https://dl.acm.org/doi/10.1145/3708359.3712104, doi:10.1145/3708359.3712104.

Kazemitabaar, M., Ye, R., Wang, X., Henley, A.Z., Denny, P., Craig, M., Grossman, T., 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs, in: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3613904.3642773, doi:10.1145/3613904.3642773.

Knezic, D., Wubbels, T., Elbers, E., Hajer, M., 2010. The socratic dialogue and teacher education. Teaching and teacher education 26, 1104–1111.

Koutcheme, C., Dainese, N., Sarsa, S., Hellas, A., Leinonen, J., Denny, P., 2024. Open Source Language Models Can Provide Feedback: Evaluating LLMs' Ability to Help Students Using GPT-4-As-A-Judge, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 52–58. URL: https://dl.acm.org/doi/10.1145/3649217.3653612, doi:10.1145/3649217.3653612.

Kumar, H., Xiao, R., Lawson, B., Musabirov, I., Shi, J., Wang, X., Luo, H., Williams, J.J., Rafferty, A.N., Stamper, J., Liut, M., 2024. Supporting Self-Reflection at Scale with Large Language Models: Insights from Randomized Field Experiments in Classrooms, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, Association for Computing Machinery, New York, NY, USA. pp. 86–97. URL: https://doi.org/10.1145/3657604.3662042, doi:10.1145/3657604.3662042.

Kuramitsu, K., Obara, Y., Sato, M., Obara, M., 2023. KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education, in: Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E, Association for Computing Machinery, New York, NY, USA. pp. 50–59. URL: https://doi.org/10.1145/3622780.3623648, doi:10.1145/3622780.3623648.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. pp. 9459–9474.

Liffiton, M., Sheese, B.E., Savelka, J., Denny, P., 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes, in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3631802.3631830, doi:10.1145/3631802.3631830.

Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., Malan, D.J., 2024a. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2, Association for Computing Machinery, New York, NY, USA. p. 1927. URL: https://doi.org/10.1145/3626253.3635427, doi:10.1145/3626253.3635427.

Liu, R., Zenke, C., Lloyd, D., Malan, D.J., 2024b. Teaching with AI (GPT), in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2, Association for Computing Machinery, New York, NY, USA. p. 1902. URL: https://doi.org/10.1145/3626253.3633433, doi:10.1145/3626253.3633433.

Liu, R., Zhao, J., Xu, B., Perez, C., Zhukovets, Y., Malan, D.J., 2025a. Improving AI in CS50: Leveraging Human Feedback for Better Learning, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 715–721. URL: https://doi.org/10.1145/3641554.3701945, doi:10.1145/3641554.3701945.

Liu, Y., Jiang, H., Zoghi, B., 2025b. Leveraging ai and machine learning to improve student engagement, learning outcomes, and trust in education. INTED2025 Proceedings , 2768–2774.

Lorås, M., Sindre, G., Trætteberg, H., Aalberg, T., 2021. Study Behavior in Computing Education—A Systematic Literature Review. ACM Trans. Comput. Educ. 22, 9:1–9:40. URL: https://dl.acm.org/doi/10.1145/3469129, doi:10.1145/3469129.

Lui, R.W.C., Bai, H., Zhang, A.W.Y., Chu, E.T.H., 2024. GPTutor: A Generative AI-powered Intelligent Tutoring System to Support Interactive Learning with Knowledge-Grounded Question Answering, in: 2024 International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp. 702–707. URL: https://ieeexplore.ieee.org/document/10898626, doi:10.1109/AEECA62331.2024.00124.

Lyu, W., Wang, Y., Chung, T.R., Sun, Y., Zhang, Y., 2024. Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, Association for Computing Machinery, New York, NY, USA. pp. 63–74. URL: https://doi.org/10.1145/3657604.3662036, doi:10.1145/3657604.3662036.

Ma, I., Krone-Martins, A., Videira Lopes, C., 2024. Integrating AI Tutors in a Programming Course, in: Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1, Association for Computing Machinery, New York, NY, USA. pp. 130–136. URL: https://doi.org/10.1145/3649165.3690094, doi:10.1145/3649165.3690094.

Mahon, J., Mac Namee, B., Becker, B.A., 2024. Guidelines for the Evolving Role of Generative AI in Introductory Programming Based on Emerging Practice, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 10–16. URL: https://doi.org/10.1145/3649217.3653602, doi:10.1145/3649217.3653602.

Manorat, P., Tuarob, S., Pongpaichet, S., 2025. Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence 8, 100403. URL: https://www.sciencedirect.com/science/article/pii/S2666920X25000438, doi:10.1016/j.caeai.2025.100403.

Margulieux, L., Ketenci, T.A., Decker, A., 2019. Review of measurements used in computing education research and suggestions for increasing standardization. Computer Science Education 29, 49–78.

Markel, J.M., Guo, P.J., 2021. Inside the mind of a cs undergraduate ta: A firsthand account of undergraduate peer tutoring in computer labs, in: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, pp. 502–508.

Maurya, K.K., Srivatsa, K.A., Petukhova, K., Kochmar, E., 2025. Unifying AI tutor evaluation: An evaluation taxonomy for pedagogical ability assessment of LLM-powered AI tutors, in: Chiruzzo, L., Ritter, A., Wang, L. (Eds.), Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Albuquerque, New Mexico. pp. 1234–1251. URL: https://aclanthology.org/2025.naacl-long.57/, doi:10.18653/v1/2025.naacl-long.57.

Mayer, R.E., Moreno, R., 2002. Aids to computer-based multimedia learning. Learning and instruction 12, 107–119.

Mayer, R.E., Moreno, R., 2003. Nine ways to reduce cognitive load in multimedia learning. Educational psychologist 38, 43–52.

Menezes, T., Egherman, L., Garg, N., 2024. AI-Grading Standup Updates to Improve Project-Based Learning Outcomes, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 17–23. URL: https://doi.org/10.1145/3649217.3653541, doi:10.1145/3649217.3653541.

Mercer, N., Howe, C., 2012. Explaining the dialogic processes of teaching and learning: The value and potential of sociocultural theory. Learning, culture and social interaction 1, 12–21.

Michaelis, J.E., Weintrop, D., 2022. Interest development theory in computing education: A framework and toolkit for researchers and designers. ACM Transactions on Computing Education 22, 1–27.

Nelson, C., Doupé, A., Shoshitaishvili, Y., 2025. SENSAI: Large Language Models as Applied Cybersecurity Tutors, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 833–839. URL: https://doi.org/10.1145/3641554.3701801, doi:10.1145/3641554.3701801.

Neumann, A.T., Yin, Y., Sowe, S., Decker, S., Jarke, M., 2025. An LLM-Driven Chatbot in Higher Education for Databases and Information Systems. IEEE Transactions on Education 68, 103–116. URL: https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10706931, doi:10.1109/TE.2024.3467912.

Neyem, A., González, L.A., Mendoza, M., Alcocer, J.P.S., Centellas, L., Paredes, C., 2024a. Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development. IEEE Transactions on Learning Technologies 17, 1599–1614. URL: https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10518103, doi:10.1109/TLT.2024.3396735.

Neyem, A., Sandoval Alcocer, J.P., Mendoza, M., Centellas-Claros, L., Gonzalez, L.A., Paredes-Robles, C., 2024b. Exploring the Impact of Generative AI for StandUp Report Recommendations in Software Capstone Project Development, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 951–957. URL: https://doi.org/10.1145/3626252.3630854, doi:10.1145/3626252.3630854.

Page, M.J., McKenzie, J.E., Bossuyt, P.M., Boutron, I., Hoffmann, T.C., Mulrow, C.D., Shamseer, L., Tetzlaff, J.M., Akl, E.A., Brennan, S.E., Chou, R., Glanville, J., Grimshaw, J.M., Hróbjartsson, A., Lalu, M.M., Li, T., Loder, E.W., Mayo-Wilson, E., McDonald, S., McGuinness, L.A., Stewart, L.A., Thomas, J., Tricco, A.C., Welch, V.A., Whiting, P., Moher, D., 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ 372, n71. URL: https://www.bmj.com/content/372/bmj.n71, doi:10.1136/bmj.n71. publisher: British Medical Journal Publishing Group Section: Research Methods &amp; Reporting.

Pang, A., Vahid, F., 2024. ChatGPT and Cheat Detection in CS1 Using a Program Autograding System, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 367–373. URL: https://doi.org/10.1145/3649217.3653558, doi:10.1145/3649217.3653558.

Pankiewicz, M., Baker, R.S., 2024. Navigating Compiler Errors with AI Assistance - A Study of GPT Hints in an Introductory Programming Course, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 94–100. URL: https://doi.org/10.1145/3649217.3653608, doi:10.1145/3649217.3653608.

P?durean, V.A., Denny, P., Singla, A., 2025. BugSpotter: Automated Generation of Code Debugging Exercises, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 896–902. URL: https://doi.org/10.1145/3641554.3701974, doi:10.1145/3641554.3701974.

Pirzado, F.A., Ahmed, A., Mendoza-Urdiales, R.A., Terashima-Marin, H., 2024. Navigating the Pitfalls: Analyzing the Behavior of LLMs as a Coding Assistant for Computer Science Students—A Systematic Review of the Literature. IEEE Access 12, 112605–112625. URL: https://ieeexplore.ieee.org/abstract/document/10636140, doi:10.1109/ACCESS.2024.3443621. conference Name: IEEE Access.

Prather, J., Denny, P., Leinonen, J., Becker, B.A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B.N., Savelka, J., 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education, in: Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education, Association for Computing Machinery, New York, NY, USA. pp. 108–159. URL: https://doi.org/10.1145/3623762.3633499, doi:10.1145/3623762.3633499.

Prather, J., Leinonen, J., Kiesler, N., Gorson Benario, J., Lau, S., MacNeil, S., Norouzi, N., Opel, S., Pettit, V., Porter, L., Reeves, B.N., Savelka, J., Smith, D.H., Strickroth, S., Zingaro, D., 2025. Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools, in: 2024 Working Group Reports on Innovation and Technology in Computer Science Education, Association for Computing Machinery, New York, NY, USA. pp. 300–338. URL: https://doi.org/10.1145/3689187.3709614, doi:10.1145/3689187.3709614.

Priemer, B., Eilerts, K., Filler, A., Pinkwart, N., Rösken-Winter, B., Tiemann, R., Zu Belzen, A.U., 2020. A framework to foster problem-solving in stem and computing education. Research in Science & Technological Education 38, 105–130.

Pua, L.X., Ramesh, R., Natarajan, P., Iyer, G.N., 2025. CustomAIzEd: Bridging Interdisciplinary Gaps in AI Education with Customized Content Using LLMs, in: 2025 IEEE Global Engineering Education Conference (EDUCON), pp. 1–10. URL: https://ieeexplore.ieee.org/document/11016642, doi:10.1109/EDUCON62633.2025.11016642. iSSN: 2165-9567.

Pădurean, V.A., Denny, P., Singla, A., 2025. BugSpotter: Automated Generation of Code Debugging Exercises, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 896–902. URL: https://dl.acm.org/doi/10.1145/3641554.3701974, doi:10.1145/3641554.3701974.

Qadir, J., 2025. Generative AI in Undergraduate Classrooms: Lessons from Implementing a Customized GPT Chatbot for Learning Enhancement, in: 2025 IEEE Global Engineering Education Conference (EDUCON), pp. 1–10. URL: https://ieeexplore.ieee.org/document/11016479, doi:10.1109/EDUCON62633.2025.11016479. iSSN: 2165-9567.

Qu, X., Sherwood, J., Liu, P., Aleisa, N., 2025. Generative AI Tools in Higher Education: A Meta-Analysis of Cognitive Impact, in: Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3706599.3719841, doi:10.1145/3706599.3719841.

Radermacher, A., Walia, G., Knudson, D., 2014. Investigating the skill gap between graduating students and industry expectations, in: Companion Proceedings of the 36th international conference on software engineering, pp. 291–300.

Rafferty, A., Ying, H., Williams, J., et al., 2019. Statistical consequences of using multi-armed bandits to conduct adaptive educational experiments. Journal of Educational Data Mining 11, 47–79.

Raihan, N., Siddiq, M.L., Santos, J.C.S., Zampieri, M., 2024. Large Language Models in Computer Science Education: A Systematic Literature Review. URL: https://arxiv.org/abs/2410.16349v1.

Ralph, P., Ali, N.b., Baltes, S., Bianculli, D., Diaz, J., Dittrich, Y., Ernst, N., Felderer, M., Feldt, R., Filieri, A., França, B.B.N.d., Furia, C.A., Gay, G., Gold, N., Graziotin, D., He, P., Hoda, R., Juristo, N., Kitchenham, B., Lenarduzzi, V., Martínez, J., Melegati, J., Mendez, D., Menzies, T., Molleri, J., Pfahl, D., Robbes, R., Russo, D., Saarimäki, N., Sarro, F., Taibi, D., Siegmund, J., Spinellis, D., Staron, M., Stol, K., Storey, M.A., Taibi, D., Tamburri, D., Torchiano, M., Treude, C., Turhan, B., Wang, X., Vegas, S., 2021. Empirical Standards for Software Engineering Research. URL: http://arxiv.org/abs/2010.03525, doi:10.48550/arXiv.2010.03525. arXiv:2010.03525 [cs].

Renzella, J., Vassar, A., Lee Solano, L., Taylor, A., 2025. Compiler-Integrated, Conversational AI for Debugging CS1 Programs, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 994–1000. URL: https://doi.org/10.1145/3641554.3701827, doi:10.1145/3641554.3701827.

Riazi, S., Rooshenas, P., 2025. LLM-Driven Feedback for Enhancing Conceptual Design Learning in Database Systems Courses, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 1001–1007. URL: https://doi.org/10.1145/3641554.3701940, doi:10.1145/3641554.3701940.

Richards, M., Waugh, K., Slaymaker, M., Petre, M., Woodthorpe, J., Gooch, D., 2024. Bob or Bot: Exploring ChatGPT's Answers to University Computer Science Assessment. ACM Trans. Comput. Educ. 24. URL: https://doi.org/10.1145/3633287, doi:10.1145/3633287.

Rivera, E., Steinmaurer, A., Fisler, K., Krishnamurthi, S., 2024. Iterative Student Program Planning using Transformer-Driven Feedback, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 45–51. URL: https://doi.org/10.1145/3649217.3653607, doi:10.1145/3649217.3653607.

Rodrigues, T., Teixeira Lopes, C., 2025. Harnessing Large Language Models for Clinical Information Extraction: A Systematic Literature Review. ACM Trans. Comput. Healthcare URL: https://dl.acm.org/doi/10.1145/3744660, doi:10.1145/3744660. just Accepted.

Rodríguez, F.J., Price, K.M., Boyer, K.E., 2017. Exploring the pair programming process: Characteristics of effective collaboration, in: Proceedings of the 2017 acm sigcse technical symposium on computer science education, pp. 507–512.

Rogers, K., Davis, M., Maharana, M., Etheredge, P., Chernova, S., 2025. Playing Dumb to Get Smart: Creating and Evaluating an LLM-based Teachable Agent within University Computer Science Classes, in: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3706598.3713644, doi:10.1145/3706598.3713644.

Sadat Shanto, S., Ahmed, Z., Jony, A.I., 2025. Generative AI for Programming Education: Can ChatGPT Facilitate the Acquisition of Fundamental Programming Skills for Novices?, in: Proceedings of the 3rd International Conference on Computing Advancements, Association for Computing Machinery, New York, NY, USA. pp. 685–692. URL: https://doi.org/10.1145/3723178.3723268, doi:10.1145/3723178.3723268.

Salden, R.J., Koedinger, K.R., Renkl, A., Aleven, V., McLaren, B.M., 2010. Accounting for beneficial effects of worked examples in tutored problem solving. Educational Psychology Review 22, 379–392.

Scaffidi, C., 2018. Employers' needs for computer science, information technology and software engineering skills among new graduates. International Journal of Computer Science, Engineering and Information Technology 8, 1–12.

Schmucker, R., Pachapurkar, N., Bala, S., Shah, M., Mitchell, T., 2025. Learning to optimize feedback for one million students: Insights from multi-armed and contextual bandits in large-scale online tutoring. arXiv preprint arXiv:2508.00270 .

Sheese, B., Liffiton, M., Savelka, J., Denny, P., 2024. Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant, in: Proceedings of the 26th Australasian Computing Education Conference, Association for Computing Machinery, New York, NY, USA. pp. 49–57. URL: https://doi.org/10.1145/3636243.3636249, doi:10.1145/3636243.3636249.

Shih, B., Koedinger, K.R., Scheines, R., 2008. A response time model for bottom-out hints as worked examples. EDM 2008, 117–126.

Shin, J., Cruz-Castro, L., Yang, Z., Castelblanco, G., Aggarwal, A., Leite, W.L., Carroll, B.F., 2024. Understanding Optimal Interactions Between Students and A Chatbot During A Programming Task, in: 2024 Winter Simulation Conference (WSC), pp. 3106–3117. URL: https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/10838799, doi:10.1109/WSC63780.2024.10838799. iSSN: 1558-4305.

Shuvo, U.A., Dip, S.A., Vaskar, N.R., Al Islam, A.B.M.A., 2025. Assessing ChatGPT's Code Generation Capabilities with Short vs Long Context Programming Problems, in: Proceedings of the 11th International Conference on Networking, Systems, and Security, Association for Computing Machinery, New York, NY, USA. pp. 32–40. URL: https://dl.acm.org/doi/10.1145/3704522.3704535, doi:10.1145/3704522.3704535.

Sigmundsson, H., 2024. How we learn and become experts. Igniting the spark. Cham: Springer .

Skulmowski, A., Xu, K.M., 2022. Understanding cognitive load in digital and online learning: A new perspective on extraneous cognitive load. Educational psychology review 34, 171–196.

Smith, D.H., Denny, P., Fowler, M., 2024. Prompting for Comprehension: Exploring the Intersection of Explain in Plain English Questions and Prompt Writing, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, Association for Computing Machinery, New York, NY, USA. pp. 39–50. URL: https://doi.org/10.1145/3657604.3662039, doi:10.1145/3657604.3662039.

Sweetser, P., 2024. Large Language Models and Video Games: A Preliminary Scoping Review, in: Proceedings of the 6th ACM Conference on Conversational User Interfaces, Association for Computing Machinery, New York, NY, USA. pp. 1–8. URL: https://dl.acm.org/doi/10.1145/3640794.3665582, doi:10.1145/3640794.3665582.

Sweller, J., 1988. Cognitive load during problem solving: Effects on learning. Cognitive science 12, 257–285.

Sætra, H.S., 2023. Generative AI: Here to stay, but for good? Technology in Society 75, 102372. URL: https://www.sciencedirect.com/science/article/pii/S0160791X2300177X, doi:10.1016/j.techsoc.2023.102372.

Tan, C.T., Atmosukarto, I., Tandianus, B., Shen, S., Wong, S., 2025a. Exploring the Impact of Avatar Representations in AI Chatbot Tutors on Learning Experiences, in: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3706598.3713456, doi:10.1145/3706598.3713456.

Tan, L.Y., Hu, S., Yeo, D.J., Cheong, K.H., 2025b. Artificial intelligence-enabled adaptive learning platforms: A review. Computers and Education: Artificial Intelligence 9, 100429. URL: https://www.sciencedirect.com/science/article/pii/S2666920X25000694, doi:10.1016/j.caeai.2025.100429.

Taylor, A., Vassar, A., Renzella, J., Pearce, H., 2024. dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 1314–1320. URL: https://doi.org/10.1145/3626252.3630822, doi:10.1145/3626252.3630822.

Vygotsky, L.S., 1978. Mind in society: The development of higher psychological processes. Harvard University Press, Cambridge, MA.

Walter, Y., 2024. Embracing the future of Artificial Intelligence in the classroom: the relevance of AI literacy, prompt engineering, and critical thinking in modern education. International Journal of Educational Technology in Higher Education 21, 15. URL: https://doi.org/10.1186/s41239-024-00448-3, doi:10.1186/s41239-024-00448-3.

Wang, F.H., 2023. Efficient generation of text feedback in object-oriented programming education using cached performer revision. Machine Learning with Applications 13, 100481. URL: https://www.sciencedirect.com/science/article/pii/S2666827023000348, doi:10.1016/j.mlwa.2023.100481.

Weinstein, Y., Madan, C.R., Sumeracki, M.A., 2018. Teaching the science of learning. Cognitive research: principles and implications 3, 1–17.

Woodrow, J., Malik, A., Piech, C., 2024. AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 1442–1448. URL: https://dl.acm.org/doi/10.1145/3626252.3630773, doi:10.1145/3626252.3630773.

Woolf, B., Burleson, W., Arroyo, I., Dragon, T., Cooper, D., Picard, R., 2009. Affect-aware tutors: recognising and responding to student affect. International Journal of Learning Technology 4, 129–164.

Yang, A.C.M., Lin, J.Y., Lin, C.Y., Ogata, H., 2024a. Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education. Computers and Education: Artificial Intelligence 7, 100309. URL: https://www.sciencedirect.com/science/article/pii/S2666920X24001127, doi:10.1016/j.caeai.2024.100309.

Yang, S., Zhao, H., Xu, Y., Brennan, K., Schneider, B., 2024b. Debugging with an AI Tutor: Investigating Novice Help-seeking Behaviors and Perceived Learning, in: Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1, Association for Computing Machinery, New York, NY, USA. pp. 84–94. URL: https://doi.org/10.1145/3632620.3671092, doi:10.1145/3632620.3671092.

Zamfirescu-Pereira, J., Qi, L., Hartmann, B., DeNero, J., Norouzi, N., 2025. 61A Bot Report: AI Assistants in CS1 Save Students Homework Time and Reduce Demands on Staff. (Now What?), in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, Association for Computing Machinery, New York, NY, USA. pp. 1309–1315. URL: https://dl.acm.org/doi/10.1145/3641554.3701864, doi:10.1145/3641554.3701864.

Zhang, R., Zou, D., Cheng, G., 2024. A review of chatbot-assisted learning: pedagogical approaches, implementations, factors leading to effectiveness, theories, and future directions. Interactive Learning Environments 32, 4529–4557.

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., Zhang, N., 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. World Wide Web 27, 58.

Zimmerman, B.J., 2002. Becoming a self-regulated learner: An overview. Theory into practice 41, 64–70.

Zönnchen, B., Hobelsberger, M., Socher, G., Thurner, V., Ottinger, S., 2025. Exploring the Role of Large Language Models as Artificial Tutors, in: 2025 IEEE Global Engineering Education Conference (EDUCON), pp. 1–10. URL: https://ieeexplore.ieee.org/document/11016571, doi:10.1109/EDUCON62633.2025.11016571. iSSN: 2165-9567.