2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52434

WIP: AI-Assisted Code Challenge Platform to Support Active Learning in Introductory Programming Courses

Juan David Yepes, Florida Atlantic University

Assistant Professor in Teaching

WIP: AI-Assisted Code Challenge Platform to Support Active Learning in Introductory Programming Courses

Juan D. Yepes
Department of Electrical Engineering and Computer Science
Florida Atlantic University, Boca Raton, FL 33431
Email: jyepes@fau.edu

Abstract

Introductory programming courses often face challenges related to student engagement, scalability, and the transition from passive lectures to active, problem-based learning. While many students can follow coding syntax and isolated examples, they frequently struggle to apply concepts cohesively when faced with open-ended problem-solving tasks. This Work-in-Progress (WIP) paper presents the development of an AI-assisted instructional tool designed to bridge this gap by enabling real-time, interactive code challenges during class sessions.

The platform allows instructors to launch coding challenges aligned with lecture topics and instantly distribute them to students' devices. Students can work individually or collaboratively in small groups, entering their code solutions directly into the system. The embedded AI assistant interacts with learners in natural language, providing conceptual explanations, contextual hints, and feedback on the logic of their approach, without directly revealing the correct answer. This conversational interaction simulates the presence of a personal teaching assistant, helping students diagnose misconceptions and reinforcing problem-solving autonomy.

Once challenges are completed, the system compiles responses and generates analytics that highlight common misconceptions, success rates, and patterns in problem-solving strategies. These analytics are then shared with both students and instructors, offering personalized and class-level feedback that can inform subsequent instruction. The design of this system aligns with current trends in computing education that emphasize formative assessment, metacognition, and the responsible integration of AI to enhance, rather than replace, human teaching.

This work also aims to prepare students for a future where AI systems increasingly assist in coding tasks. By engaging students in active problem-solving rather than passive code generation, the platform cultivates understanding of computational reasoning and helps learners internalize programming as a structured language for expressing logic, an essential skill for adapting to emerging AI-driven workflows.

A pilot implementation was conducted in a large introductory programming course composed of two sections, where data on usability, learning outcomes, and student perceptions were collected. The feedback from this trial has been instrumental in the plan to implement improvements to the platform's interface, AI tutoring behavior, and feedback mechanisms, with the long-term goal of enabling scalable adoption in diverse classroom contexts.

Keywords:

Introductory Programming; Code Challenges; Generative AI; Formative Feedback; Computer Science Education

1. Introduction

Introductory programming courses represent a foundational component of computer science and engineering education. These courses are often students' first sustained exposure to computational thinking, algorithmic reasoning, and formal problem decomposition. Despite their importance, such courses consistently face pedagogical challenges related to student engagement, cognitive overload, and the scalability of instructional support, particularly in large classroom settings.

A common instructional model in introductory programming relies heavily on live coding demonstrations, where instructors explain syntax and logic while students observe. While effective for introducing language constructs, this approach often promotes passive learning. Students may appear to follow the lecture but struggle to reproduce or adapt the underlying logic when faced with independent problem-solving tasks. This gap between recognizing code patterns and constructing original solutions is well documented in computer science education literature [9].

Active learning strategies, such as in-class exercises and peer instruction, have been shown to improve conceptual understanding and retention. However, their effectiveness depends on timely, individualized feedback. In a typical introductory programming class with more than 30 students, instructors are unable to provide real-time guidance to every learner who becomes stuck, misinterprets problem requirements, or encounters runtime errors. As a result, even well-designed active learning activities may fall short of their pedagogical potential when deployed at scale.

Recent advances in generative artificial intelligence (GenAI) offer new opportunities to address this scalability challenge by providing on-demand assistance during problem-solving. At the same time, their use in programming education raises significant concerns. Unrestricted AI code generation risks short-circuiting the learning process by replacing student reasoning with automated solutions. For AI to function as an effective educational aid, it must be carefully designed to support thinking rather than substitute for it.

This Work-in-Progress paper documents an instructional intervention that seeks to balance these opportunities and risks. The project introduces a simple AI-assisted Code Challenge platform that enables instructors to deploy short, focused programming problems during class and provide students with immediate, guided support through a Socratic-style AI assistant. Rather than generating complete solutions, the AI emphasizes problem understanding, requirement identification, and incremental reasoning.

An additional contribution of this work lies in its development model. The platform was conceived by the instructor and implemented by a senior engineering design team as a capstone project. Through weekly design meetings, pedagogical objectives were translated into concrete system features, resulting in a tool explicitly tailored to the needs and expectations of introductory programming students. This paper presents the system design, describes its

classroom deployment in an introductory Python course, and reports preliminary assessment results from student feedback and instructor observations.

2. Background and Related Work

This work is based on research in computer science education, active learning pedagogy, intelligent tutoring systems, and the emerging role of generative artificial intelligence (GenAI). Below we summarize foundational studies and situate the AI-assisted Code Challenge platform within these intersecting areas.

2.1 Active Learning in Introductory Programming

Active learning has been shown to improve student performance and retention in STEM education. Active learning strategies, including in-class exercises, peer instruction, and problem-based learning, engage students actively in constructing knowledge, fostering deeper understanding beyond passive listening. In computer science, such strategies have been associated with increased participation, motivation, and the development of problem-solving skills. A recent systematic review [1] found that active methodologies contribute to student engagement and meaningful learning in computing disciplines, leading to higher achievement and motivational outcomes compared to traditional instruction.

Evidence from broader STEM research supports the claim that active learning reduces failure rates and improves exam performance relative to lecture-only courses. Meta-analyses of thousands of studies show that students in active learning environments are significantly less likely to underperform than those in passive lectures. This body of work underpins calls to adopt student-centered instructional strategies in CS and engineering courses [2].

In introductory programming contexts specifically, active learning allows learners to apply concepts immediately, revealing misconceptions early and making classroom time more productive. However, the effectiveness of these activities depends heavily on timely feedback [10]. Without rapid guidance, learners who make early errors, such as incorrect loop logic, misuse of conditionals, or misunderstanding of input/output requirements, may disengage or wait passively for instructor help.

Despite the documented benefits of active learning, scaling these practices remains challenging; large classes often lack adequate instructor or teaching assistant support for real-time formative feedback. These constraints motivate the exploration of tools that provide scalable, immediate feedback during active learning activities.

2.2 Intelligent Tutoring Systems and Programming Education

Intelligent Tutoring Systems (ITSs) have a long history in educational research and have been applied to computing education to support personalized, adaptive feedback. ITSs leverage AI techniques to model student knowledge and tailor instructional support based on individual

learner interactions with educational tasks. These systems have been shown to improve learning by providing adaptive support, tracking learner behavior, and scaffolding problem-solving processes [3].

In programming education, ITS research has explored automated coding tutors that provide syntax feedback, hint scaffolding, and step-by-step guidance. Prior work on ITSs for programming demonstrates that when tutors focus on reasoning and scaffolded explanations, learners benefit more than when systems simply present correct answers [4].

Several systems have explored automated feedback for student code submissions, including static analysis tools and autograders. While these tools are effective for summative assessment and large-scale grading, they often lack the conversational adaptability needed for *in-the-moment* formative learning. Many ITS implementations are also designed for asynchronous use outside of class rather than integrated into live instruction.

The platform presented in this paper aligns with the formative tutoring tradition by emphasizing immediate, contextual feedback during problem solving. What sets it apart from traditional ITSs is the use of a conversational interface powered by a large language model, allowing students to request clarifications or hints in natural language rather than through pre-structured menus or rule sets.

2.3 Generative AI in Computer Science Education

The rapid adoption of generative AI has significantly impacted programming education. Large language models can produce syntactically correct code and explanations in response to simple prompts, creating both opportunities and challenges for instruction. Research highlights concerns around academic integrity and over-reliance on AI for solutions, which can undermine deeper engagement with computational logic. Educators are increasingly discussing how to integrate GenAI into curricula in ways that support, rather than replace, student reasoning [5].

At the same time, emerging work indicates that generative AI can play a constructive role when appropriately constrained. Studies show that AI systems designed to provide explanations, debugging guidance, or reflective prompts (rather than complete solutions) can support learners' understanding without undermining agency. Structured AI guidance that emphasizes conceptual support and incremental reasoning aligns with educational best practices and can complement active learning interventions when embedded thoughtfully [6].

The key challenge, therefore, lies in designing AI-assisted programming tools that reinforce learning objectives rather than replace them. Proposed strategies include limiting direct code generation, focusing on explanation and reasoning prompts, and integrating guideline structures that preserve students' cognitive engagement while scaling personalized support [7].

2.4 Research Gap and Contribution

While prior work has examined active learning in programming courses, intelligent tutoring systems, and the educational implications of generative artificial intelligence, relatively few studies have explored the integration of AI-assisted tutoring directly into live classroom activities. Much of the existing work focuses on asynchronous tools, post-submission feedback, or answer-oriented AI systems that operate outside the temporal and instructional dynamics of an in-class learning environment. As a result, these systems are often insufficiently integrated with instructor workflows and provide limited support for real-time formative intervention during active learning activities [8].

This Work-in-Progress addresses this gap by presenting an AI-assisted Code Challenge platform designed explicitly for in-class use. The system enables instructors to deploy short programming tasks in real time and provides students with Socratic, AI-based guidance that emphasizes problem understanding and incremental reasoning rather than solution generation. In parallel, instructor-facing analytics offer live visibility into student participation and progress, supporting timely instructional decisions during class.

An additional contribution of this work lies in its student-led development model. The platform was designed and implemented by senior engineering students as part of a capstone project, working closely with the instructor through weekly design meetings. Because the developers had recently completed introductory programming courses themselves, they were able to identify strongly with the target student population and incorporate peer-level perspectives into the system's design. During early classroom pilots, this proximity to the learner experience enabled rapid incorporation of student feedback, leading to iterative refinements that improved usability, clarity, and instructional effectiveness.

The following sections describe the system architecture, development process, and classroom implementation, followed by an analysis of preliminary assessment data collected during a pilot deployment in an introductory Python programming course.

3. System Design and Architecture

The AI-assisted Code Challenge platform was designed to support real-time, in-class programming activities while preserving pedagogical control and cognitive engagement. This section describes the student-led development process, the overall system architecture, and the key features of both the instructor and student interfaces.

3.1 Student-Led Development Process

The platform was developed as part of a senior engineering design capstone project. At the beginning of the semester, multiple project ideas were presented to the design cohort, from which students selected a concept to develop. The AI-assisted Code Challenge platform was one of these proposed ideas, as illustrated in Figure 1, which shows the original project description presented to the student teams.

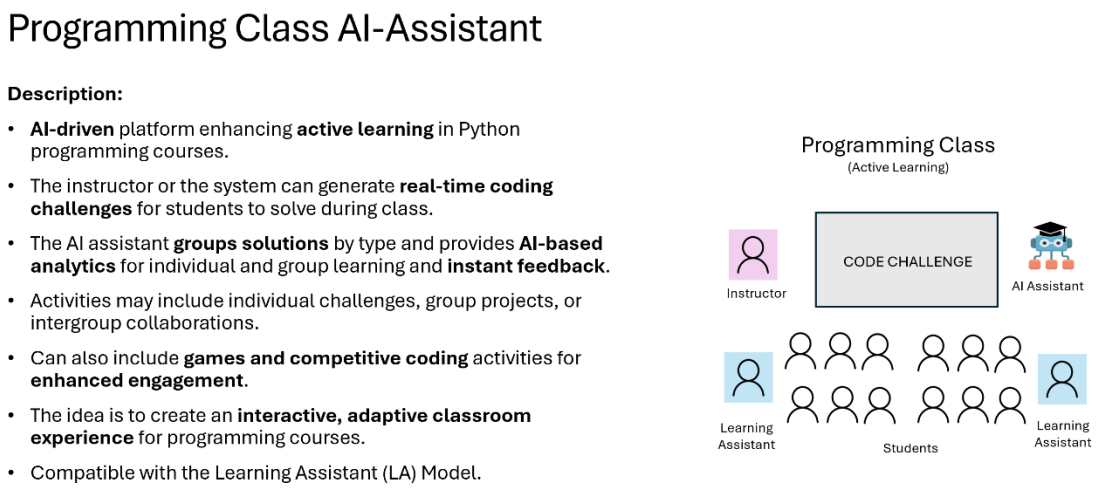

Figure 1. Initial project description slide presented to senior engineering design students, outlining the motivation and scope of the AI-assisted Code Challenge platform.

The team that selected this project consisted of five senior engineering students who worked under the guidance of the course instructor through weekly design meetings. Rather than providing a fully specified technical blueprint, the instructor articulated high-level pedagogical objectives, such as supporting active learning, minimizing cognitive shortcuts, and improving real-time classroom awareness, and allowed the student team to translate these goals into concrete system requirements and design decisions.

Figure 2 presents the suggested high-level project components provided to the students at the outset of development. These components outlined the expected system elements, including a student-facing interface, an instructor dashboard, backend services, and an AI-assisted feedback module, while leaving implementation details open to student interpretation and design choices.

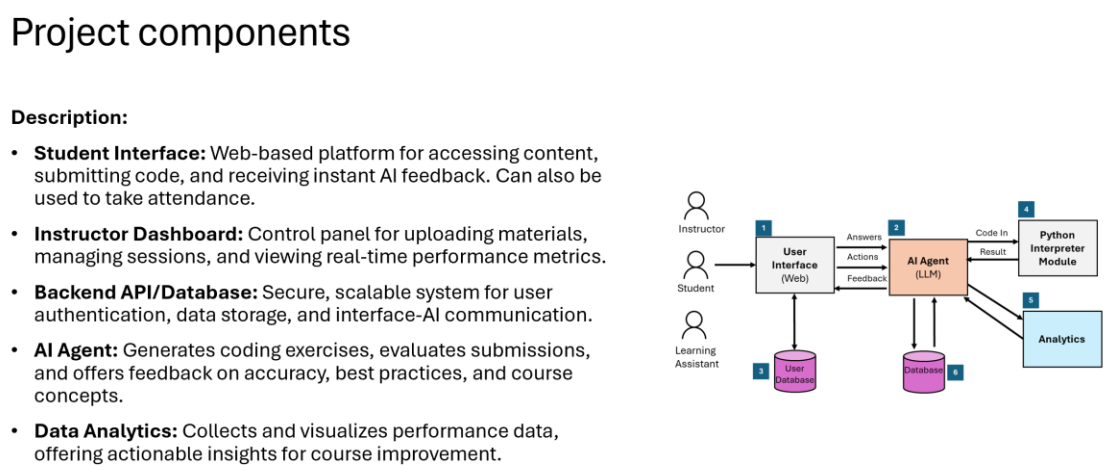

Figure 2. Suggested system components provided to the student development team, highlighting the modular architecture and pedagogical requirements of the platform.

This collaborative process positioned the students as both developers and representative end users. Their peer-level perspective influenced decisions related to interface simplicity, interaction flow, and terminology, helping ensure that the platform aligned with student expectations and classroom realities. Over the course of the second semester, the team iteratively refined the system based on instructor feedback, informal usability testing, and practical classroom constraints.

The platform was demonstrated during two live lecture sessions across two different sections of an introductory Python programming course, as shown in Figure 3. These demonstrations provided early formative feedback from students, which informed subsequent refinements. The final prototype was presented at the University Senior Design Showcase, where it received first-place recognition in its category. The project poster presented at the showcase is shown in Figure 4.

Figure 3. Live classroom demonstrations of the AI-assisted Code Challenge platform conducted in two sections of an introductory Python programming course.

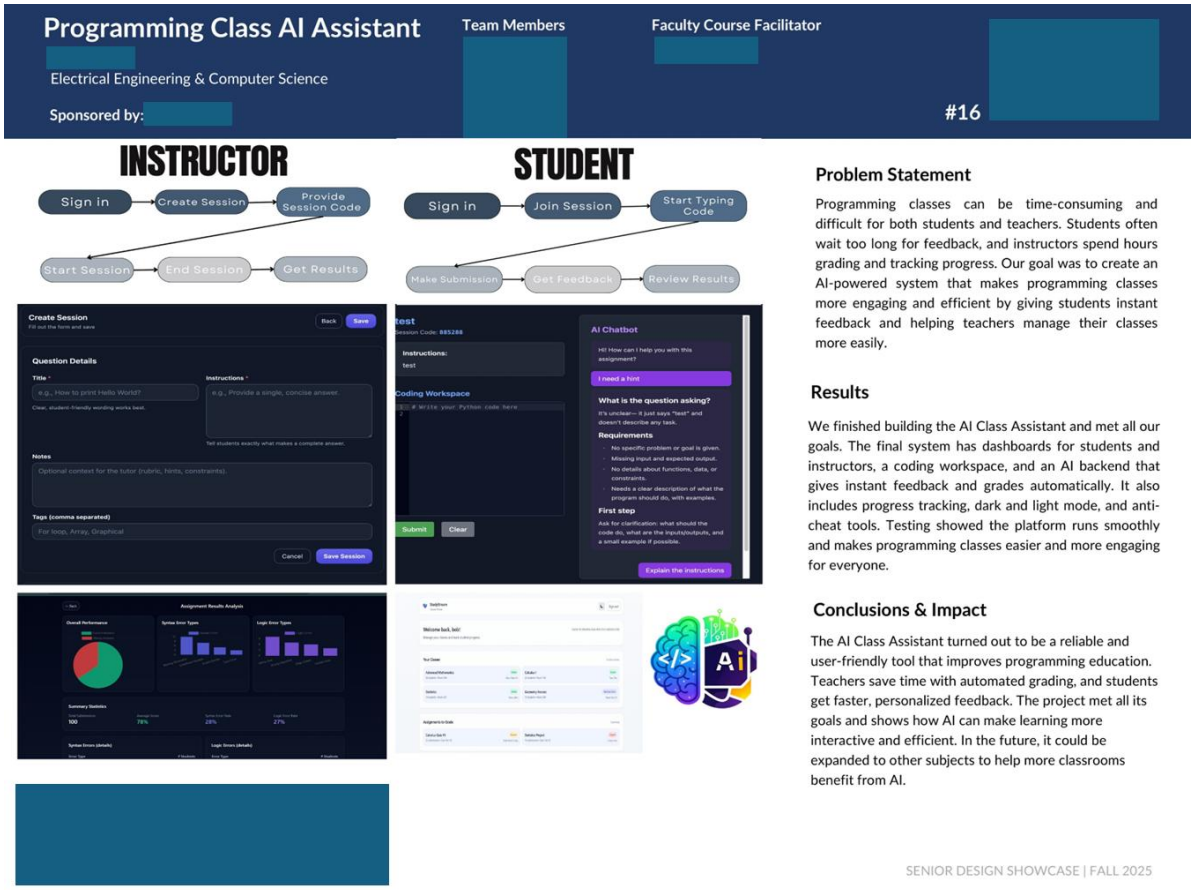

Figure 4. Project poster presented by the student development team at the University Senior Design Showcase.

3.2 Overall System Architecture

The AI-assisted Code Challenge platform follows a modular, web-based architecture composed of four primary components:

a) **Instructor Dashboard** – a real-time control and monitoring interface for managing in-class activities.
b) **Student Coding Workspace** – a browser-based environment in which students solve coding challenges and submit solutions.
c) **AI-Assisted Tutoring Module** – a large language model configured to provide Socratic-style guidance rather than solution generation.
d) **Backend Services and Analytics Layer** – responsible for session management, data storage, system coordination, and analytics generation.

At the start of a classroom session, the instructor initiates a code challenge through the instructor dashboard. The challenge prompt is distributed simultaneously to all connected student devices. Students then work within the coding workspace, where they write and submit their solutions

and may optionally interact with the AI assistant. Throughout the session, the backend services track connection status, submission progress, and interaction metadata, enabling real-time analytics for instructional use.

Figure 5 illustrates the instructor interface used to configure and launch a new code challenge, highlighting the simplicity of session setup and real-time deployment.

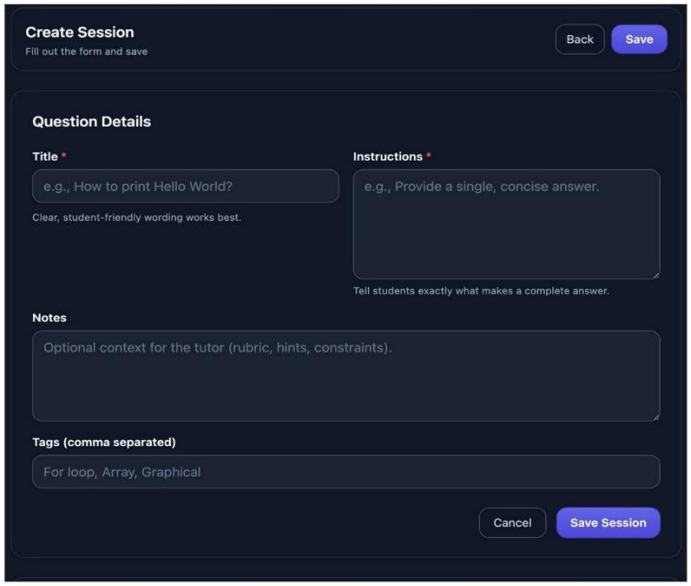

Figure 5. Instructor interface used to configure and launch a new in-class code challenge session.

3.3 Instructor Dashboard: Real-Time Classroom Awareness

The instructor dashboard functions as a centralized command interface for managing and monitoring in-class programming activities. Its design prioritizes situational awareness rather than detailed inspection of individual student code. Key features include:

- Live Session Control: Instructors can initiate, pause, and terminate code challenges during a lecture, enabling seamless transitions between explanation and practice.
- Connectivity Visualization: The dashboard displays the number of students currently connected to the session, providing immediate feedback on participation levels.
- Submission Progress Indicators: Visual representations (e.g., grouped or radial displays) indicate which students have submitted solutions and which remain active.
- At-a-Glance Analytics: Aggregated metrics allow instructors to quickly identify patterns, such as widespread difficulty or uneven progress, supporting timely instructional intervention.

Importantly, the dashboard is designed to support instructional decision-making rather than automated grading. Instructors retain full control over pacing, discussion, and follow-up explanation based on observed classroom dynamics.

Figure 6 shows the instructor dashboard design, emphasizing its role as a real-time awareness tool rather than an assessment interface.

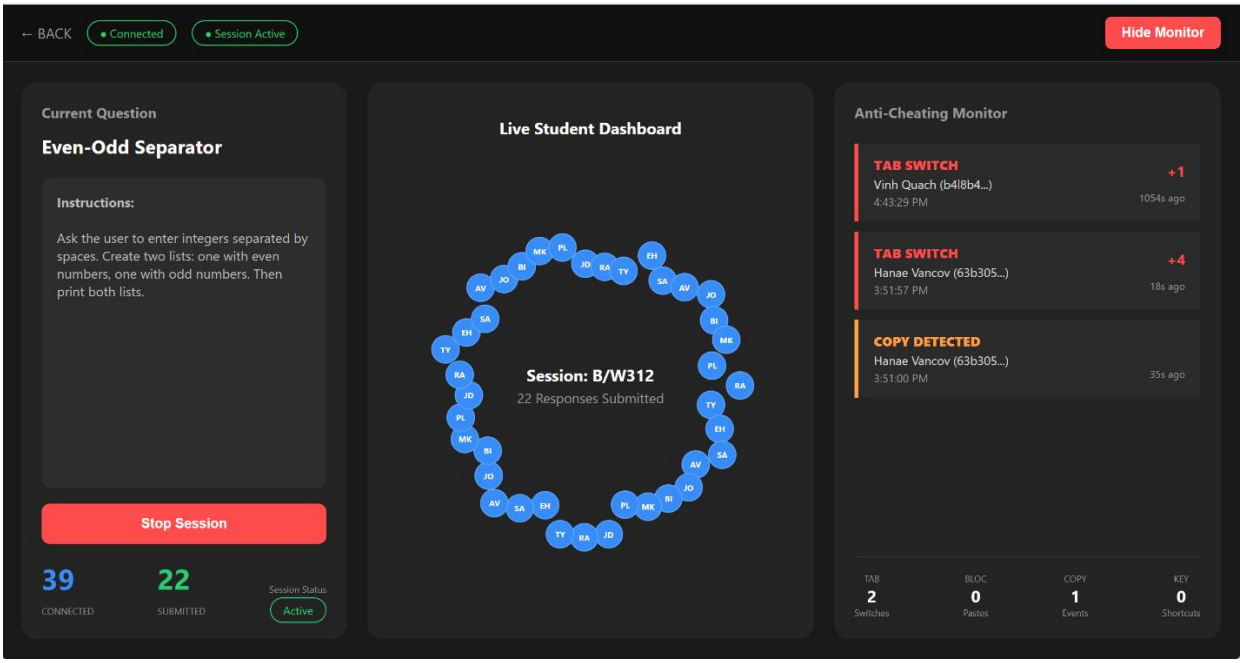

Figure 6. Instructor dashboard design emphasizing real-time participation tracking and submission progress.

3.4 Student Coding Workspace

The student-facing interface integrates problem presentation, code entry, and AI-assisted support within a single web-based workspace. The primary design goal was to minimize friction and cognitive overhead so that students could focus on problem-solving rather than tool navigation. Core features include:

- Challenge Prompt Display: A clearly formatted description of the task, including input/output requirements and constraints.
- Code Editor: A lightweight editor for writing and modifying code solutions.
- Submission Controls: Direct submission functionality, with the ability to revise code prior to final submission.
- AI Interaction Panel: A conversational interface through which students can request guidance or clarification.
- The workspace supports both individual work and informal collaboration, consistent with active learning practices commonly employed in introductory programming courses.

Figure 7 presents the student coding workspace, illustrating the integration of the code editor, prompt display, and AI interaction panel.

3.5 Socratic AI-Assisted Tutoring Strategy

A central design decision of the platform is the use of a Socratic AI assistant rather than a solution-generating agent. The AI module is explicitly constrained to avoid providing complete code solutions. Instead, it supports learning through structured, incremental guidance, including:

- Clarification of Requirements: Assisting students in restating or interpreting problem instructions.
- Identification of First Steps: Suggesting how to decompose the problem into manageable sub-tasks.
- Conceptual Reminders: Reinforcing relevant programming constructs (e.g., conditionals, loops, data structures) without prescribing specific implementations.
- Error Interpretation: Offering general explanations for common issues, such as logical misconceptions or incorrect assumptions, without directly correcting code.
- Students initiate AI interactions through prompts such as “I need a hint” or “Explain the instructions.” This opt-in interaction model preserves student agency and reduces the risk of over-reliance on automated assistance.

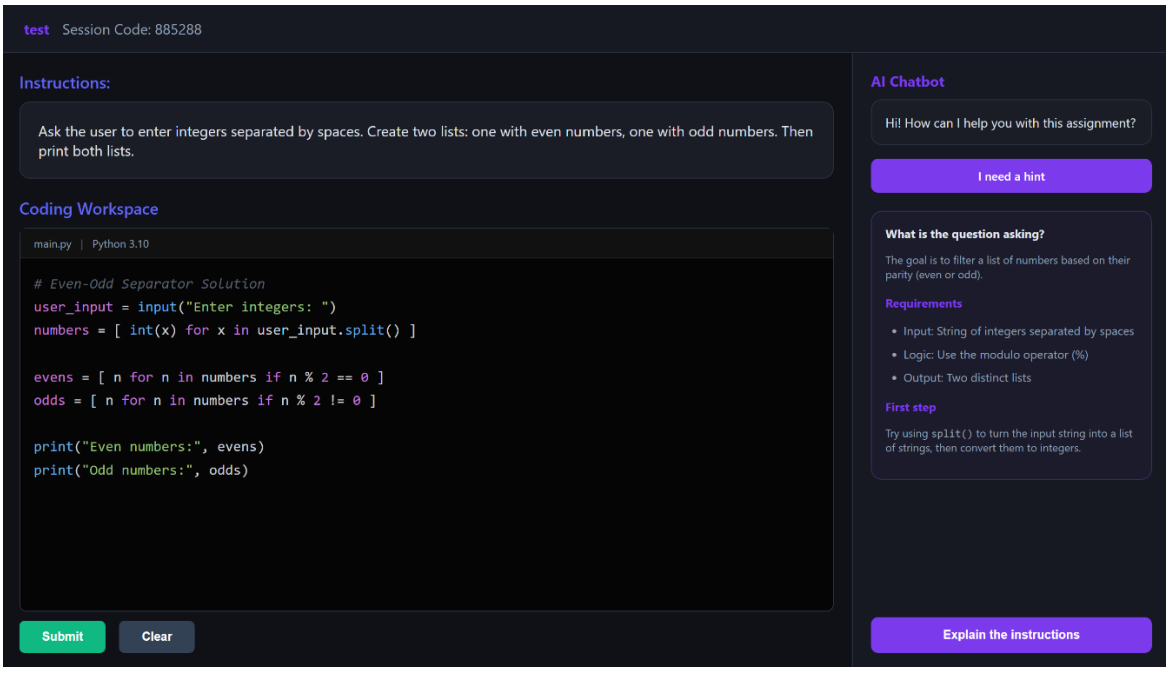

Figure 7. Student coding workspace integrating problem description, code editor, and AI-assisted guidance.

This approach aligns with productive failure frameworks, which suggest that guided struggle prior to solution exposure can enhance conceptual learning [11].

3.6. Backend Services and Analytics Layer

Figure 8 illustrates the analytics and results framework associated with AI interactions and student submissions, supporting both student reflection and instructor insight. Due to time constraints this feature was not fully implemented on the pilot phase.

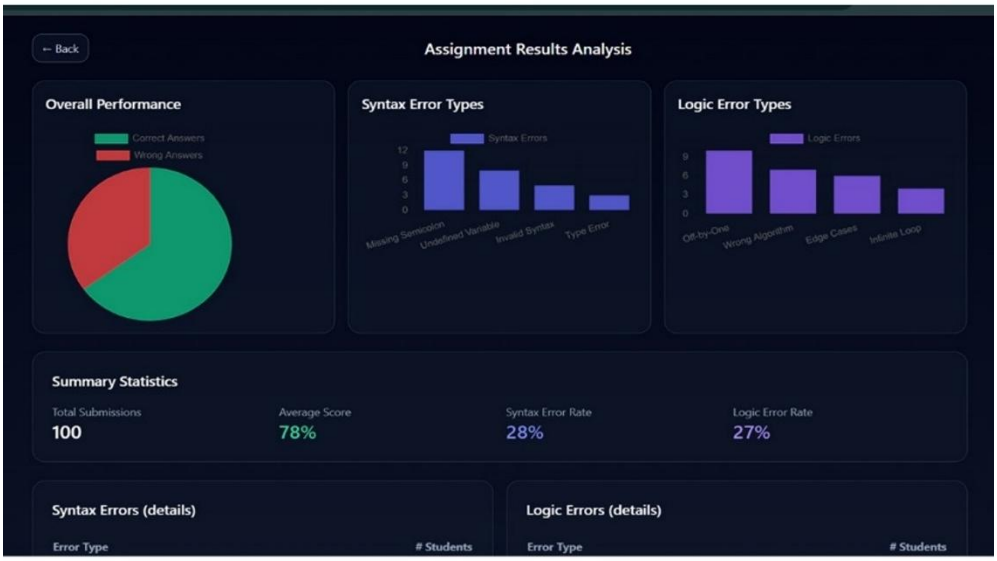

Figure 8. Analytics and results framework showing aggregated student responses and interaction data.

4. Implementation and Classroom Deployment

4.1 Deployment Context

The AI-assisted Code Challenge platform was piloted during live sessions of an introductory Python programming course at ___ University. The course enrolled approximately 100-150 undergraduate students, distributed across two sections (morning and afternoon), primarily from engineering and computing majors. The course followed a traditional lecture–lab structure.

The pilot was conducted during two regularly scheduled lecture sessions, one from each section, without modifying course credit, grading policies, or assessment weighting. This approach was intentionally selected to observe authentic student engagement and interaction with the platform under normal classroom conditions.

The instructor introduced the platform as a formative learning tool designed to support in-class problem-solving rather than graded assessment. Students accessed the system using personal laptops and joined the session through a shared session code displayed during class.

4.2 Case Study: In-Class Code Challenge

To evaluate the platform under realistic instructional conditions, a short programming challenge was deployed mid-lecture. The task required students to:

- Accept a sequence of integer inputs
- Separate the values into even and odd lists
- Output the resulting lists according to specified formatting rules

This challenge was selected because it integrates several foundational programming concepts commonly taught in introductory Python courses, including input handling, conditional logic, list operations, and iteration. The task is sufficiently constrained to be completed within a short in-class time window, yet open-ended enough to reveal common misconceptions and logical errors.

During the activity, students worked individually and were encouraged to interact with the AI assistant if they encountered difficulties interpreting the problem requirements or structuring their solution. This interaction model allowed students to seek guidance at their own pace while preserving autonomy and minimizing disruption to the overall classroom flow.

5. Results

Assessment data were collected through three complementary sources: (1) anonymous student surveys administered immediately after the demo, (2) qualitative feedback captured through open-ended responses, and (3) instructor observations supported by dashboard analytics.

5.1 Quantitative Student Feedback

An anonymous post-activity survey was completed by 76 students following the in-class demonstration across two course sections. Survey items focused on system usability, perceived instructional value of the AI assistant, student engagement, and perceptions of the tool's role relative to instructor teaching.

As summarized in Table 1, responses across all categories were predominantly positive. Items related to usability (Q1–Q2) received the strongest agreement, with approximately 77–80% of respondents indicating that the platform was easy to access and that code entry and submission were clear. These results suggest that the technical interface posed minimal barriers to participation during the activity.

Survey items addressing conceptual support (Q3–Q6) yielded more varied responses, with 59–65% of students agreeing or strongly agreeing that the AI assistant helped them understand concepts and guided problem-solving without revealing full solutions. A substantial proportion of neutral responses (28–38%) suggests that the perceived usefulness of AI assistance may depend on individual learning preferences or the specific nature of the programming task.

Measures of student engagement (Q7–Q8) showed strong positive trends, with 70–75% of respondents reporting that they felt actively engaged during the challenge and that the activity

was more engaging than a typical lecture. These findings support the use of short, in-class coding challenges as an effective active learning strategy.

Finally, responses related to future use and instructional trust (Q9–Q10) indicate broad acceptance of the platform's pedagogical role. Approximately 71–74% of students expressed interest in using the tool again in future classes and agreed that the system supported the instructor's teaching rather than replacing it. This result is particularly relevant given ongoing concerns about the role of generative AI in educational settings. Table 1 provides a condensed summary of student perceptions by category. Full response distributions for all survey items are provided in Appendix A.

Table 1. Summary of Student Perceptions of the AI-Assisted Code Challenge (N = 76)

Category	Survey Items	Agree + Strongly Agree	Neutral	Disagree + Strongly Disagree
Usability	Q1–Q2	77%–80%	12%–14%	8%–10%
Conceptual Support	Q3–Q6	59%–65%	28%–38%	7%–12%
Engagement	Q7–Q8	70%–75%	14%–28%	8%–12%
Future Use & Trust	Q9–Q10	71%–74%	21%–22%	5%–9%

5.2 Qualitative Feedback and Thematic Analysis

Open-ended survey responses (Q11) and informal post-demo feedback were analyzed using an inductive thematic approach. Student comments were coded and grouped into recurring themes related to instructional value, system limitations, AI interaction design, and academic integrity. Table 2 summarizes the primary themes and their instructional implications, which broadly corroborate the quantitative findings while identifying priorities for future development.

Table 2. Summary of Qualitative Themes from Student Feedback (Q11)

Theme	Description of Student Feedback	Instructional Implication
Perceived Benefits	Students described the platform as helpful and supportive during in-class problem solving. The AI assistant was frequently noted for clarifying problem requirements and suggesting next steps without revealing full solutions. The conversational interface lowered barriers to seeking help, particularly for students hesitant to ask questions publicly.	Supports AI-assisted formative guidance as a scalable complement to instructor feedback.

Need for Code Execution	The most prominent concern was the lack of built-in code execution. Students emphasized that the ability to run and test code is central to their programming workflow. Its absence increased friction and, in some cases, led to greater reliance on AI assistance.	Highlights the importance of integrating execution tools to preserve independent debugging practices.
AI Hint Design	Students reported mixed experiences with AI hints. While some found them helpful, others described them as overly indirect or repetitive, especially for specific logic or syntax errors. Requests for more adaptive or staged hinting were common.	Motivates refinement of AI scaffolding strategies and error-sensitive hinting.
Usability and Integrity Considerations	A smaller subset of students noted interface issues (cursor behavior, indentation, dark mode visibility) and expressed concerns about rigid or insufficiently flexible academic integrity mechanisms.	Indicates the need for interface polish and instructor-configurable integrity controls.

6. Discussion

The pilot deployment suggests that AI-assisted code challenges can enhance active learning in introductory programming courses when designed with pedagogical constraints in mind. Rather than replacing instructor interaction, the AI assistant functioned as a scalable extension of formative support, allowing students to receive immediate guidance while preserving the instructor's central role in framing, pacing, and consolidating learning, consistent with established perspectives on AI as an augmentative educational technology [12].

The student-led development model contributed to interface accessibility and relevance, as design decisions reflected authentic student workflows and expectations. This approach highlights the potential of capstone design projects not only as technical exercises, but also as mechanisms for translating pedagogical intent into classroom-ready tools. At the same time, the pilot revealed important design tensions, including the need for more precise AI feedback and integrated code execution capabilities to support independent debugging.

Overall, the findings indicate that carefully constrained generative AI can support in-class programming activities without undermining cognitive engagement, provided that system design emphasizes reasoning, transparency, and instructor oversight.

7. Limitations

As a Work-in-Progress study, this project has several limitations that inform both interpretation of the results and directions for future research. First, the pilot deployment was conducted in a single introductory Python course during two live classroom sessions. While the feedback collected provides valuable early insight, the results rely primarily on student perception and instructor observation rather than direct or longitudinal measures of learning outcomes.

As a Work-in-Progress study, the primary goal of this investigation is to establish feasibility, usability, and pedagogical alignment rather than to draw definitive conclusions about learning effectiveness.

Second, the system did not include an integrated code execution or testing environment at the time of deployment. As reflected in student feedback, the inability to run code prior to submission limited students' ability to debug independently and may have influenced perceptions of usability. Additionally, while the AI assistant was intentionally constrained to provide Socratic-style guidance, some students found the feedback insufficiently specific for diagnosing certain error types.

Finally, academic integrity mechanisms were limited in scope during the pilot. Although the platform included basic protections against direct copying, students identified potential workarounds. Additionally, as with many early classroom deployments, novelty effects may have influenced engagement and perceptions, underscoring the need for repeated use across semesters to assess sustained impact.

8. Future Work

Future development of the AI-assisted Code Challenge platform will focus on both technical enhancements and expanded pedagogical evaluation. Planned system improvements include:

- Integration of a secure code execution and testing environment to allow students to validate logic prior to submission.
- Refinement of AI-assisted hinting to better differentiate between syntax errors, logic errors, and conceptual misunderstandings.
- Enhanced instructor controls for managing submission revisions and regulating copying behavior during sessions.
- Improvements to user interface consistency across devices and display modes.

From a research perspective, future work will involve deploying the platform across multiple course sections and semesters to examine its impact on learning outcomes, persistence, and confidence in programming. Additional studies may explore the effectiveness of AI-assisted code challenges in other programming languages or engineering courses, as well as comparisons between AI-supported and non-AI active learning interventions.

Conclusion

This Work-in-Progress paper presented the design, development, and initial classroom deployment of an AI-assisted Code Challenge platform intended to support active learning in introductory programming courses. By enabling instructors to deploy real-time coding tasks and providing students with Socratic-style AI guidance, the platform addresses a persistent challenge in computer science education: scaling formative feedback without diminishing cognitive engagement.

A distinguishing feature of this project is its student-led development model, which leveraged a senior engineering design team to translate pedagogical goals into a functional classroom tool. Preliminary assessment results indicate positive student perceptions of AI-assisted support, increased engagement during in-class problem-solving, and improved instructor awareness of student progress.

While further evaluation is needed, these early findings suggest that carefully constrained generative AI can play a constructive role in programming education when embedded within active learning frameworks. This work contributes to ongoing discussions on responsible AI integration in education and highlights the potential of capstone design projects as vehicles for instructional innovation.

Acknowledgments

The author would like to acknowledge the senior engineering design students Tazwar Ibrahim, Aaroha Sapkota, Gianna Biondolillo, Aymerick Osse and Carlos Juarez who developed the AI-assisted Code Challenge platform as part of their capstone project. Their technical creativity, iterative design efforts, and responsiveness to pedagogical goals were essential to the success of this work.

References

[1] Córdova-Esparza, D. M., Romero-González, J. A., Córdova-Esparza, K. E., Terven, J., & López-Martínez, R. E. (2024). Active learning strategies in computer science education: A systematic review. *Multimodal Technologies and Interaction*, *8*(6), 50.

[2] Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the national academy of sciences*, *111*(23), 8410-8415.

[3] Guo, L., Wang, D., Gu, F., Li, Y., Wang, Y., & Zhou, R. (2021). Evolution and trends in intelligent tutoring systems research: a multidisciplinary and scientometric view. *Asia Pacific Education Review*, *22*(3), 441-461.

[4] Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018, January). Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian computing education conference* (pp. 53-62).

[5] Beale, R. (2025). Computer Science Education in the Age of Generative AI. *arXiv preprint arXiv:2507.02183*.

[6] Reihanian, I., Hou, Y., & Sun, Q. (2025). From Pilots to Practices: A Scoping Review of GenAI-Enabled Personalization in Computer Science Education. *AI*, *7*(1), 6.

[7] Nathaniel, J., Oyelere, S. S., Suhonen, J., & Tedre, M. (2025). Literature Review on the Integration of Generative AI in Programming Education. *International Journal of Artificial Intelligence in Education*, 1-32.

[8] Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., ... & Sarsa, S. (2024). Computing education in the era of generative AI. *Communications of the ACM*, *67*(2), 56-67.

[9] Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer science education, 13(2), 137-172.

[10] Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of educational research*, *77*(1), 81-112.

[11] Kapur, M. (2008). Productive failure. *Cognition and instruction*, *26*(3), 379-424.

[12] Luckin, R., & Holmes, W. (2016). Intelligence unleashed: An argument for AI in education.

Appendix A. Full Survey Response Distribution

AI Assistant Code Challenge

#	Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	No response	Total
Q1	The tool was easy to access and use on my device	5	1	11	21	36	2	76
Q2	It was clear how to enter and submit my code	5	1	9	23	38	0	76
Q3	The AI assistant's explanations helped me understand the concept	3	1	29	26	16	1	76
Q4	The AI hints guided me without giving away the full answer	6	6	20	23	20	1	76
Q5	Using an AI assistant during class felt helpful for my learning	7	4	21	25	19	0	76
Q6	This activity helped me understand *why* my code worked or didn't work	2	6	24	24	20	0	76
Q7	I felt actively engaged and thinking during the challenge	5	6	11	34	20	0	76
Q8	This activity was more engaging than a typical lecture	4	6	21	22	23	0	76
Q9	I would like to use this tool again in future classes	4	6	17	23	25	1	76
Q10	The tool supported the instructor's teaching instead of replacing it	4	3	16	30	23	0	76

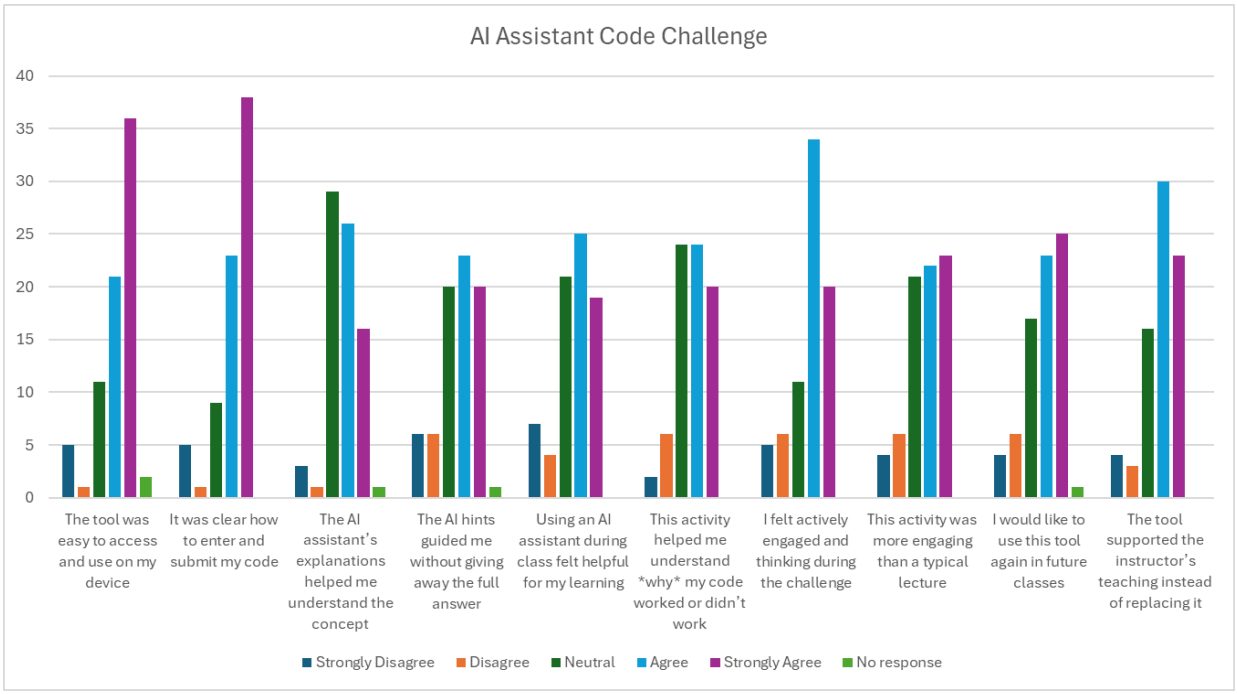

Figure 9. Distribution of student responses to post-activity survey items (Likert scale).