Human-Human-AI Triadic Programming: Uncovering the Role of AI Agent and the Value of Human Partner in Collaborative Learning

Taufiq Daryanto
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
taufiqhd@vt.edu

Xiaohan Ding
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
xiaohan@vt.edu

Kaike Ping
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
pkk@vt.edu

Lance T Wilhelm
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
lancewilhelm@vt.edu

Yan Chen
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
ych@vt.edu

Chris Brown
Computer Science
Virginia Tech
Blacksburg, Virginia, United States
Virginia, USA
dcbrown@vt.edu

Eugenia H Rho
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
eugenia@vt.edu

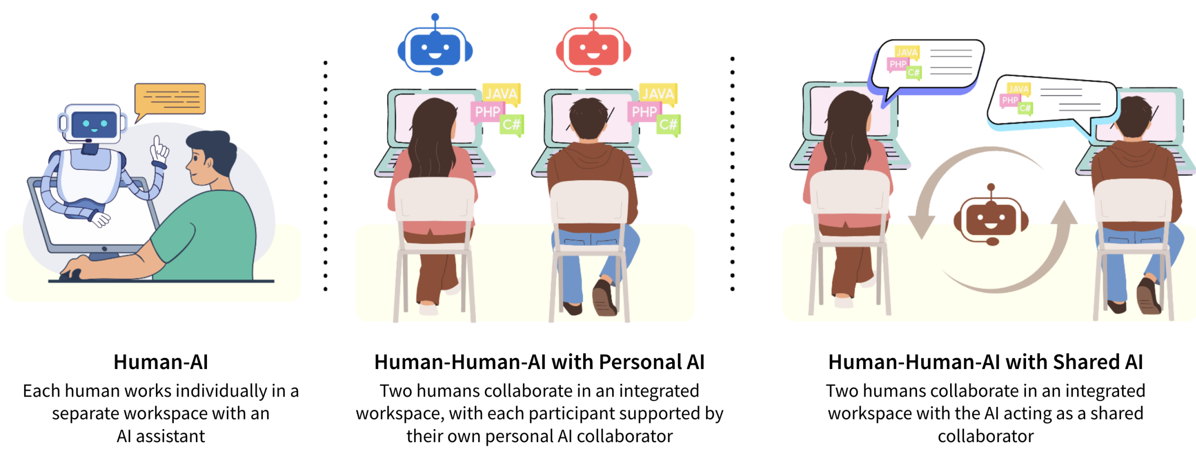

Figure 1: Rethinking AI for Programming Learning. Rather than replacing human partners, can AI augment a human–human pair programming while preserving the social and pedagogical benefits of collaboration? To answer that, we explore "human–human–AI triadic programming", where two humans work together with an AI agent. Our study compares this approach to human–AI pairs and examines design considerations in this triadic interaction, including whether AI should act as a shared collaborator or personal support.

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3791773

Abstract

As AI assistance becomes embedded in programming practice, researchers have increasingly examined how these systems help learners generate code and work more efficiently. However, these studies often position AI as a replacement for human collaboration and overlook the social and learning-oriented aspects that emerge in collaborative programming. Our work introduces human-human-AI (HHAI) triadic programming, where an AI agent serves as an additional collaborator rather than a substitute for a human partner. Through a within-subjects study with 20 participants, we show that triadic collaboration enhances collaborative learning and social presence compared to the dyadic human–AI (HAI) baseline. In the triadic HHAI conditions, participants relied significantly less on AI generated code in their work. This effect was strongest in the HHAI-shared condition, where participants had an increased sense of responsibility to understand AI suggestions before applying them. These findings demonstrate how triadic settings activate socially shared regulation of learning by making AI use visible and accountable to a human peer, suggesting that AI systems that augment rather than automate peer collaboration can better preserve the learning processes that collaborative programming relies on.

CCS Concepts

• Human-centered computing → Collaborative interaction; **Empirical studies in HCI**; **User interface programming**.

Keywords

human-computer interaction, collaborative programming, proactive agent, AI agent, learning

ACM Reference Format:
Taufiq Daryanto, Xiaohan Ding, Kaike Ping, Lance T Wilhelm, Yan Chen, Chris Brown, and Eugenia H Rho. 2026. Human-Human-AI Triadic Programming: Uncovering the Role of AI Agent and the Value of Human Partner in Collaborative Learning. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3772318.3791773

1 Introduction

AI assistance has become increasingly prevalent in programming, with tools such as ChatGPT [84] and GitHub Copilot [41] now common in everyday development practices [109]. Prior work shows that these tools can accelerate productivity and improve task completion [109], and this trend has extended into educational settings where AI is frequently used for programming activities [108]. Many studies adopt a human–AI pair programming setup in which a learner works with an AI model on the same task [36, 74]. These studies often position the AI as a replacement for a human partner [74], and are frequently compared to collaborative programming practices such as human–human pair programming [36, 74]. Although substituting a peer with an AI can speed up progress and improve coding outcomes [36, 74], this approach can overlook the social and collaborative dimensions that support learning in peer-based programming between two human partners [36]. This raises a central question for computing education about whether AI can augment human collaboration in ways that preserve the social and pedagogical benefits of peer learning.

Existing studies largely examine the benefits of dyadic human–AI programming, with most work emphasizing speed and code quality as key evaluation metrics [36, 74]. However, these metrics tend to overlook the social and collaborative experiences in learning that shape how students learn [81]. In collaborative programming, learners work together by articulating their thought processes and sharing knowledge [6] as they jointly contribute to a shared task [82]. Even in the era of AI, these collaborative skills remain valuable, particularly as modern work places increasingly adopt human-AI teaming practices [49, 80] where teams of humans and autonomous agents can work together toward shared goals [9]. Students similarly rely on peers for shared reasoning, clarification, and support during programming activities [112], yet they also recognize that AI can provide distinct forms of assistance that are difficult for peers to offer [24]. To better understand the combined strengths of both human and AI partners, we investigate a **human–human–AI triadic programming** approach where two human programmers collaborate with an AI agent as a third teammate.

Moving from human-AI dyads to a triadic human–human–AI setting introduces distinct design considerations [22]. First, should the AI act as a *shared* collaborator for the human-human pair, or as *personal* support for each human partner [34, 55, 71, 119]? Second, beyond providing on-demand answers, what types of AI interventions are useful, for example, should the AI intervene proactively when the pair is stuck [18, 55, 87]? Following recent work that defines proactive support as system-initiated assistance in the programming environment [87], we refer to a proactive AI agent as one that has the capability to initiate suggestions without always being explicitly prompted by the user. Prior work shows that such proactive AI behaviors in programming settings can be perceived as helpful partners and improve collaboration experiences, but may also be disruptive [18, 87]. However, such work has focused exclusively on dyadic human–AI interactions. We know little about how a proactive AI agent should participate in synchronous programming with two humans, where conversational flow, role coordination, and mutual accountability add additional layers of complexity [14, 51, 113]. Clarifying when and how an AI should intervene, and how its positioning reshapes collaboration between two human partners, remains an open challenge. In this work, we address this gap through the design and evaluation of a human–human–AI triadic programming system that allows us to compare shared versus personal AI positioning and proactive versus on-demand support.

To investigate this, we built an integrated workspace that connects two humans and an AI agent in real time, combining a collaborative code editor with a conversational interface that supports text and voice (Figure 2). The system is designed to support collaborative programming practice between the two humans, allowing them to actively engage in continuous dialogue to articulate their thought processes and coding. The AI agent complements this interaction and can assist on demand through hints and code analysis, as well as through lightweight proactive interventions (e.g., providing hints during idle periods), while avoiding premature solution reveal.

We then conducted a within-subjects study of 20 participants (10 pairs) to examine key design considerations for human-human-AI triadic programming, such as the scope of AI interactions (shared collaborator versus personal support), the usefulness of different intervention types, and user perceptions towards human-human-AI triadic programming in general. Specifically, we compared three conditions: (1) **Shared AI**, where the AI addresses the pair as a third collaborator; (2) **Personal AI**, where each human has an individual AI partner; and (3) **Human–AI**, a baseline condition where each human works alone with AI.

Finally, we analyzed the usage of AI and participant perceptions across conditions.

In this study, we investigate:

- **RQ1:** What are the benefits and drawbacks of human-human-AI triadic programming compared to human-AI pair programming?
- **RQ2:** How does AI positioning (shared vs. personal) affect learning experiences in human-human-AI triadic programming?
- **RQ3:** What types of AI support are useful for human-human-AI triadic programming?

Our study shows that human-human-AI (HHAI) triadic programming provides a stronger collaborative learning experience and greater social presence compared to the human–AI (HAI) condition. Participants noted that having both a human partner and an AI enabled multiple ways of learning, such as observing the interaction between their peer and the AI, or learning by explaining concepts to their partner. Interestingly, we also found that working with a peer (especially in the shared HHAI condition) increased participants' sense of responsibility in how they engaged with the AI and corresponded with lower reliance on AI-generated code. Participants mentioned that knowing their partner was observing their interaction with the AI encouraged them to use the AI more responsibly.

We make the following contributions to HCI research on human–AI collaboration and learning:

(1) **Empirical evidence on triadic human–human–AI collaboration**. Through a within-subjects study with 20 computer science students, we provide the first systematic comparison of human–human–AI triadic programming against human–AI baselines. Our findings show that adding a peer alongside AI restores collaborative learning and social presence, fosters accountable AI use, and reshapes how proactive AI suggestions are integrated.
(2) **A conceptual framing of augmentation versus automation in collaborative programming**. We show how replacing a peer with AI removes the interactional conditions that sustain dialogue, accountability, and explanation, whereas augmenting peer collaboration with AI preserves and reinforces these mechanisms. By grounding this distinction in empirical evidence from collaborative programming, we extend long-standing HCI discussion on automation versus augmentation with a concrete account of their divergent effects on collaborative learning.
(3) **Design principles for AI-supported collaborative programming**. We derive implications for future AI tools for collaborative programming: (a) making AI outputs visible to peers to scaffold accountability, (b) tuning AI proactivity to conversational flow, and (c) augmenting rather than displacing the pedagogical benefits of peer collaboration.

2 Related Work

2.1 AI for Programming Education

The integration of AI into education has rapidly evolved in recent years [44], enabling more personalized learning experiences and supporting a wide range of applications [2]. These advancements have led to its adoption in various domains, including programming education [86]. For example, studies have examined how AI tools can assist students in coding tasks such as generating solutions, debugging, and learning programming concepts [39, 108]. AI has also been applied to create programming education resources [45], such as generating coding exercises [7, 93]. Additionally, researchers have developed interactive AI systems that tailor the learning process to specific computing skills [60, 72, 73]. Furthermore, in higher education, several universities have begun integrating AI into programming courses by allowing the usage of AI coding assistants [7].

Among these applications, AI integration has recently received increasing attention for its role in collaborative programming [101, 107, 116]. Collaborative programming refers to multiple programmers working jointly on the same coding task [82, 107]. One widely used form of this practice is pair programming [36, 71, 74] where two programmers work together in a single workspace, in which one user assumes the *driver* role of writing the code, while the other takes the *navigator* role of reviewing and suggesting improvements [51]. This practice has been shown to improve problem-solving skills [111], enhance code quality [57], and increase learning gains and satisfaction [15]. In educational contexts, pair programming can also be done remotely [10], where two students collaborate using a collaborative online IDE (Integrated Development Environment), while only one programmer writes code at a time, thus still reinforcing the driver-navigator framework. Studies also mention that remote pair programming can be similarly effective as in-person pair programming [94].

Recent work in this area has leveraged AI to support programming practice, primarily by substituting one human partner with an AI partner [74]. For example, studies have examined the use of tools such as GitHub Copilot or ChatGPT to assist with code writing, generate documentation, and produce comments [71]. Similarly, other research has explored AI for programming learning, where the AI provides hints or step-by-step guidance to support learners as they work through coding problems [89]. More recent studies, such as Pu et al. [87] and Chen et al. [18], have further advanced this approach by developing AI assistants capable of proactively providing help during the coding process.

While those AI-assisted programming approaches have shown benefits by increasing programming performance [36], they often lack the social presence, mutual engagement, and peer support found in human–human collaborative programming [36]. Moreover, studies have shown that many students still prefer learning with a peer because they find it more enjoyable and experience a stronger human connection [24]. In collaborative programming in particular, as students collaborate through continuous dialogue, they develop their ability to articulate thought processes and problem-solving strategies [6]. Yet, despite these advantages, prior work has also noted the limitations of learning solely with another human partner, such as the lack of targeted, expert feedback [24]. Limiting collaboration to only human partners can therefore miss opportunities to leverage AI strengths to provide detailed support and feedback [24].

To combine the benefits of both human and AI partners while addressing their respective limitations, we explore human–human–AI triadic programming, where the AI acts as an additional collaborator rather than replacing a human partner. Although prior work has examined AI tools such as ChatGPT and GitHub Copilot for programming support [71], little is known about how to effectively integrate AI into human–human–AI settings. Our study addresses this gap by investigating AI roles, types of beneficial AI interventions, and user perceptions in such collaborations.

2.2 Multi-Human and AI Collaboration

In recent years, human–AI collaboration has become an important research topic in HCI and CSCW [16, 27, 35, 110, 114]. Much of the existing work has focused on one-on-one interactions between humans and AI [23, 106, 110, 114], where AI tools are designed to help individual users, such as writing assistance [97, 114], code completion [27, 100], learning [23, 105, 110], and decision support [16, 35]. These studies highlight the role of AI in supporting individuals and have driven the development of intelligent assistant applications.

In practice, however, many collaboration scenarios involve multiple people working together, such as group learning [21, 59], team brainstorming [42, 79], and decision-making [19, 26]. This has led researchers to explore multi-human–AI collaboration, examining the roles and functions of AI within group settings [19, 53, 55, 59, 115]. For example, Houde et al. (2025) studied how conversational AI participates in small-group discussions [55]. They found that AI could inspire and support groups but might also cause frustration when it dominated the conversation [55].

We observe that group collaboration with AI support often falls into two paradigms. The first is the **Personal AI** paradigm, where AI provides individualized feedback or suggestions to each group member, helping them contribute to the collective task [34, 119]. For example, this occurs when individuals each use ChatGPT while still collaborating with one another [34, 119]. The second is the **Shared AI Teammate** paradigm, where AI is designed to act as a group member that directly participates in collaboration alongside humans [55, 79]. Houde et al.'s study illustrates this approach, showing that proactive AI participation can foster creativity and sustain conversational flow [55].

These paradigms have been studied in various domains such as education [59], design [53], and group decision-making [19, 53]. However, they remain underexplored in collaborative programming [71, 74]. Drawing upon typical collaborative programming settings [8, 10], this form of collaboration is distinctive because it requires high synchronicity, with participants simultaneously engaging in coding and verbal communication while navigating different roles within the team [14, 36, 51, 74]. This multimodal and multi-role nature sets collaborative programming apart from other collaboration contexts and raises open questions about how AI should participate. To address this gap, our work investigates how Personal AI and Shared AI paradigms shape multi-human–AI collaboration in the context of triadic programming.

3 System Design for Human–Human–AI Triadic Programming

We developed a system to support human–human–AI triadic programming. The goal is to facilitate collaborative programming between two humans and an AI agent in an integrated environment where all participants can contribute simultaneously. The system is designed to enable collaborative learning between the two humans, enabling them to actively engage in continuous dialogue through which they articulate their thought processes and problem-solving strategies. The AI agent complements this interaction by proactively joining the discussion, providing guidance, offering hints, and suggesting potential solutions when needed.

This system builds upon prior work on programming assistance [18, 71, 74, 87, 89, 117] and AI-assisted learning [24, 72, 73], which emphasizes integration [71], multimodality [46], and proactivity [18, 87, 117]. Our implementation enables two human programmers and an AI agent to collaborate in the same integrated workspace, supporting both human–human coordination and direct human–AI interaction to support collaboration and learning. The system supports multimodal interaction, allowing collaboration through voice, text, code typing, and interactive controls. The AI agent is designed to be proactive, meaning it can autonomously participate in the collaboration and provide assistance [18, 87], while also remaining available on demand when explicitly queried. By using this system, we aim to explore human–human–AI triadic programming for learning, enabling both humans to learn together with the support of a proactive AI agent.

To clarify the boundary of our system's capabilities, we designed our AI agent to be proactive but not fully agentic [92] because its interventions are based on predefined, task-specific rules, such as providing support after a period of user inactivity, rather than on autonomous decision making. Our design follows prior work on proactive AI agents that offer system-initiated assistance during idle moments in programming activity [18, 87], as we describe in §3.2.1. Although the AI in our system is not fully agentic, prior work shows that users often perceived proactive AI agents as partners or collaborators rather than as simple tools [55, 87].

3.1 Integrated Interface for Triadic Programming

The interface integrates a collaborative code editor with a conversational interface, and connects two humans with an AI agent simultaneously through WebSocket [75]. This design supports three-way communication, allowing both humans and the AI to interact within the same environment. Interaction with the AI is multimodal: users can engage not only through text-based chat but also via voice, code editor, and interactive buttons. By keeping all interactions within a single workspace, we aim to reduce the need for context switching, addressing limitations noted in prior work where students had to move between the coding environment and external tools like ChatGPT during collaborative programming [71].

Our tool is primarily designed to support remote collaboration. We focused on remote settings because they provide greater flexibility for participants to work together from different locations [11]. Remote collaborative programming has become increasingly

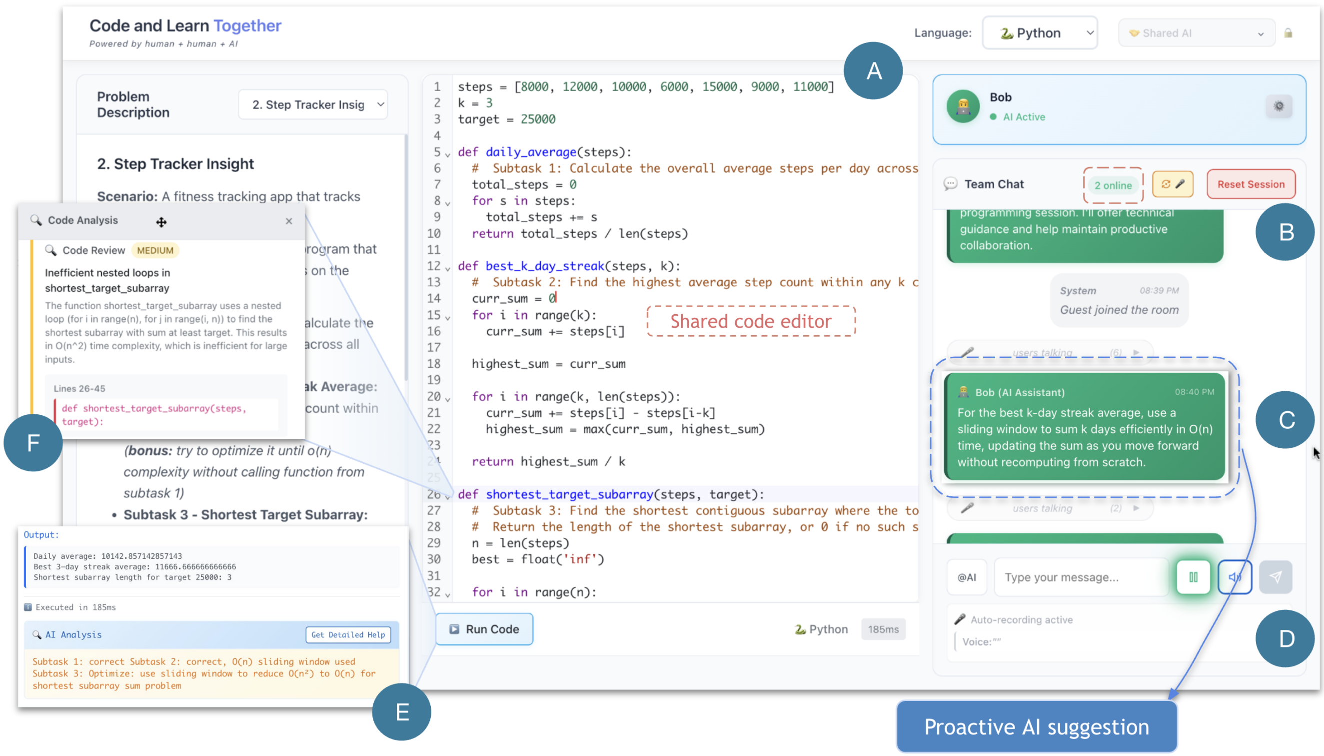

Figure 2: Integrated Interface to Facilitate Human–Human–AI Triadic Programming. A) Collaborative Code Editor: A live-shared editor where all participants can work together in real time. B) Conversational Interface: Supports dialogue between the two humans and the AI. The AI can also be queried directly through spoken or typed questions (Direct Request). C) Proactive Intervention: The AI agent can proactively intervene in the conversation with contextually relevant suggestions. D) Live Transcription: All spoken input is automatically transcribed using speech-to-text, allowing AI suggestions to be grounded in the ongoing conversation, the typed code, and the given problem. E) Code Run Feedback: Each time the code is executed, the AI analyzes the output and provides debugging tips or improvement suggestions when necessary. F) Code Block Analysis: By right-clicking a code block (e.g., a loop or function), users can request AI feedback that evaluates correctness and suggests improvements. ***The enlarged view of this figure is provided in Appendix E.***

common in both educational contexts [10, 56] and professional environments [11], offering benefits comparable to in-person settings [94].

3.1.1 Collaborative Code Editor. The code editor (Figure 2. A) design was based on LeetCode [67], a widely used platform for practicing programming. The interface displays the programming problem on the side and provides a code editor implemented using CodeMirror [52]. The editor supports multiple programming languages (C++, Java, and Python) and allows users to write and run code directly within the environment. We implemented real-time collaboration features that allow both users to edit code in the same online code editor, with support for viewing each other's cursors and text highlights, similar to the Live Share feature in Visual Studio Code [76]. This synchronization was built using WebSocket [75], which enables two-way communication so that editing events such as keystrokes and cursor movements are shared instantly between participants with minimal delay.

3.1.2 Conversational Interface. We implemented a conversational interface into the workspace (Figure 2. B), positioned on the right side of the coding environment. The interface supports both text and voice interaction, enabling communication between the two humans as well as with the AI agent. All spoken input is automatically transcribed using live speech-to-text transcription, allowing the AI to process dialogue in real time. Based on these transcripts, the AI can respond proactively or reactively according to several triggers, which we describe in the following subsection on AI interventions.

To support interaction, we also provided an option for users to enable AI voice output in addition to text [24]. Furthermore, every interaction with the AI is contextually grounded as the prompts include the current programming problem, the live state of the shared code editor, and the ongoing conversation. This design aims to provide relevant suggestions without requiring users to repeatedly explain their coding context [71].

3.2 AI Interventions

We implemented the AI agent based on the GPT-4.1-mini large language model (All the prompts are provided in Appendix A). The AI agent is designed to participate in the collaboration through a set of intervention mechanisms informed by prior work on programming assistance [18, 78, 87, 117] and AI-supported learning in programming [24, 89]. These interventions are intended to provide support while maintaining opportunities for collaborative problem-solving among both users. We designed the AI agent to be able to intervene proactively in the conversation [55], or respond reactively when manually prompted or triggered by users [89]. In addition, we prompted the LLM to support learning [4] by offering guidance and hints without revealing full solutions too early [89], which aims to encourage peer discussion [102].

(1) Proactive Intervention during Idle Period: The AI provides guidance or hints to re-engage users when both conversation and coding activity remain idle, without being explicitly prompted by users (Figure 2. C). Productive collaboration relies on participants actively sharing their thoughts and engaging with one another [25], so extended silence is often unproductive [31]. To address this, the AI intervenes by offering guidance that can help learners overcome moments of being stuck and continue making progress [50]. This design is informed by prior work on proactive programming assistance, which demonstrates the benefits of offering support during idle periods [18, 87]. Following prior implementations [18], the AI triggers interventions after 5 seconds of silence or inactivity (i.e., neither typing nor talking), but restricts them to once every 20 seconds since the last AI message to reduce disruptiveness.
(2) Direct Request: Users can directly engage the AI by addressing it explicitly (e.g., saying "Hey Bob ..." (Bob is the name for the AI agent) or typing "@AI ...") to request clarification, hints, or additional explanations. They can also ask follow-up questions based on prior suggestions. To support the learning process, the AI is prompted not to reveal full solutions, but instead to provide just enough hints or guidance to help users make progress [89].
(3) Task Scaffolding: We implemented a code editor feature that generates inline suggestions from user comments, similar to interactions in existing AI coding assistants (e.g., GitHub Copilot [41]). However, when users write inline instructions as comments in the code editor (e.g., "## implement ..."), the AI generates task scaffolding or step-by-step todos instead of directly providing the full code [69]. This approach is designed to support the learning process by helping users get started without immediately revealing the solution. If users remain stuck, they can right-click on a todo line to reveal the corresponding code implementation for that specific line.
(4) Code Run Feedback: Each time the code is executed, the AI analyzes the output and provides suggestions or debugging tips when necessary (Figure 2. E) [87]. If users are still uncertain about the feedback, they can click "Get Detailed Help" to bring the AI's explanation into the conversation for more detailed guidance.
(5) Code Block Analysis: By right-clicking a code block (e.g., a loop or a function), users can request an AI analysis that evaluates whether the implementation is correct and suggests possible improvements, such as enhancing code efficiency (Figure 2. F). One potential use case of this feature is when a user finishes writing a block of code and wants to check its quality. Another potential use case is when the navigator (the other user who is not actively coding) wants to verify the implementation while reviewing the driver's code, using the AI as additional support [14].

3.3 Human-Human-AI Interaction: Shared Versus Personal

Drawing on prior work that examined different ways of integrating AI into collaborative settings [34, 53, 55, 71, 119], we aim to explore how an AI agent can be positioned in human–human–AI triadic programming: either as a shared collaborator supporting both participants collectively [53, 55, 115], or as a personal collaborator assisting each participant individually [34, 119].

3.3.1 ***Shared AI*.** In this mode, the AI acts as a third collaborator, interacting with both humans simultaneously. The AI is positioned as a shared collaborator that can proactively engage with the group. All interventions are directed to the pair as a whole, enabling the AI to participate directly in the joint dialogue between the two human collaborators. This design builds on a growing interest in non-dyadic human–AI interaction [22] and recent studies showing how proactive AI can support group discussions by contributing ideas and facilitating conversation [55]. It also parallels findings in collaborative programming where third-party interventions, such as those from an instructor, have been shown to assist the learning process [113].

3.3.2 ***Personal AI*.** In this mode, each human is paired with their own AI collaborator. This design reflects common practices where individuals use AI tools (e.g., ChatGPT) while still collaborating with others [34, 119]. It also aligns with current programming conditions in which collaborators may each rely on tools such as ChatGPT or GitHub Copilot [71]. In our implementation, the AI can still provide proactive interventions, but it interacts only with its paired user rather than addressing the group. Unlike in the shared mode, the AI does not use voice output in this configuration because its interventions are directed only to each individual. Instead, the AI provides text-based support, making its assistance more individualized while the two humans continue to collaborate with each other.

4 Empirical Study of Human-Human-AI Triadic Programming

4.1 Participants

We recruited 20 computer science students (10 pairs) from the first author's university. Participants registered together as pairs who already knew each other. We acknowledge that this choice limits the generalizability of our findings, as working with unfamiliar partners may lead to different experiences, which we leave for future work to explore.

All participants had experience using AI tools for coding (e.g., ChatGPT [84]). The majority (18/20) had prior experience with collaborative programming, either through class projects or personal projects. Most of our participants (16/20), who were third-year undergraduates or above, had taken a data structures and algorithms course, while the remaining four participants (second-year students) had taken an intermediate data structures course. Participants were recruited through the university mailing list and class announcements. All participants were compensated $25 for a 90-minute user study. Detailed demographic information is provided in Table 1.

4.2 Study Design

To examine the role of AI agent and the value of a human partner in human–human–AI triadic programming, we conducted a within-subject study in which participants engaged with our system across three different conditions (Figure 1). The order of conditions was counterbalanced to mitigate ordering and learning effects. All study sessions were conducted remotely using Zoom, as collaborative programming can also be done in a remote setting [30, 56]. In our study, although our tool allows for simultaneous code editing, we instructed participants that only one person should edit the code at a time, which grounds our work in the driver–navigator model of pair programming [10]. However, participants could switch roles at any point by coordinating verbally, since both participants had access to the same editor. We draw this practice from prior literature on remote pair programming, which suggests the benefit of maintaining the driver–navigator model [1].

The three study conditions were as follows:

(1) **Human–Human–AI with Shared AI**: Two humans collaborate in real time in an integrated workspace with the AI acting as a shared collaborator (§3.3.1). In this mode, all AI interventions are directed to the pair collectively, positioning the AI as a third collaborator in the collaboration.
(2) **Human–Human–AI with Personal AI**: Two humans collaborate in real time in an integrated workspace, with each participant supported by their own personal AI collaborator (§3.3.2). Here, the AI provides individual support to each human, enabling personalized guidance while still working within the shared environment. Comparing the first and second conditions enables us to understand how to position AI in human-human-AI triadic programming, either as a personal collaborator or a shared collaborator.
(3) **Human–AI**: Each human works individually in a separate room with an AI assistant, using the same implementation described in §3.3.1 but without a human partner. This ablated condition allows us to compare collaboration with and without a human partner, helping uncover the value and role of a human partner in the process, as well as whether users perceive the AI differently in human–AI versus human–human–AI collaboration.

Our study focuses on conditions that involve AI because contemporary programming practices increasingly rely on AI-assistance. As a result, our baseline is a human–AI rather than a human–human collaboration setup. Although human–human collaboration in pair programming, has long been an established practice in computing education [90], recent research has significantly shifted toward AI-assisted programming [36, 71, 74]. This shift reflects the extent to which AI tools have become integrated into both educational [70] and professional programming contexts [66], making their use inseparable from modern learning and development practices in programming contexts [70, 108].

By exploring three AI-supported collaboration conditions, we aim to examine how people collaborate with AI under different configurations and understand how these configurations shape programming practice. Moreover, by comparing the human–human–AI and human–AI conditions, we seek to uncover the added value of having a human partner alongside the AI, highlighting how AI can complement human collaboration in the learning contexts. This can potentially reinforce the importance of maintaining human involvement rather than positioning AI as a replacement for human partners.

4.3 Programming Tasks

The programming tasks were adapted from LeetCode problems [67], as their scope and style are well-suited for studying programming practice in line with prior work [24, 87]. Each task was modified to include multiple subproblems that participants could work on. We designed the tasks to support multiple solution approaches with varying levels of optimization, enabling participants to explore different strategies and reflect on trade-offs [24]. This flexibility encouraged exploration and discussion among both humans and the AI. We selected tasks that emphasized algorithmic thinking rather than heavy syntax, since developing strong problem-solving skills is critical for students' reasoning abilities and highly relevant for career preparation [24]. To ensure fairness across conditions, all tasks were constructed to be of comparable difficulty. Additionally, the coding tasks are distributed equally across all 3 conditions to avoid confounding task difficulty. All tasks are provided in Appendix B.

4.4 Procedure

The study was conducted online via Zoom, where two participants joined the session from different locations. We began by asking participants to complete a consent form, as the study had been approved by the Institutional Review Board (IRB) at the first author's institution. Next, we introduced the tool and provided a live demonstration. Participants were then instructed to work on three conditions (HHAI Shared, HHAI Personal, and HAI as shown in Figure 1). The order of conditions was counterbalanced across participants. Each condition involved a separate problem. Before each task, participants received an explanation of the tool and the condition. They were given 17 minutes to complete each task. After finishing a task, participants completed a questionnaire. At the end of the study, we conducted a semi-structured interview to gather insights into their experiences and perceptions of each condition. In total, the user study lasted 90 minutes. Each participant was compensated with a $25 Amazon gift card for the study.

4.5 Data Collection and Analysis

Post-task questionnaire: After finishing each task, participants completed a questionnaire. This included the Collaborative Learning Scale [36, 99], the Social Presence Questionnaire [36, 65], and

Table 1: Participant Demographics (Pairs)

ID	Gender	Ethnicity	Education	ID	Gender	Ethnicity	Education
P1	Male	Asian	3rd Year Undergrad	P2	Male	Asian	3rd Year Undergrad
P3	Male	Asian	Master	P4	Male	White	Master
P5	Male	Asian	Master	P6	Male	Asian	1st Year PhD
P7	Male	Asian	2nd Year Undergrad	P8	Male	Asian	2nd Year Undergrad
P9	Male	White	3rd Year Undergrad	P10	Male	White	3rd Year Undergrad
P11	Male	White	2nd Year Undergrad	P12	Male	MENA	2nd Year Undergrad
P13	Male	MENA	Master	P14	Female	White	Master
P15	Male	Asian	Master	P16	Male	Asian	Master
P17	Female	Asian	3rd Year Undergrad	P18	Female	White	3rd Year Undergrad
P19	Male	Black	3rd Year Undergrad	P20	Female	Black	3rd Year Undergrad

several individual questions about their perceptions of AI in the collaboration. All questions are provided in Appendix C.

Qualitative interview: All study sessions were recorded on Zoom and automatically transcribed. At the end of the study, we conducted semi-structured interviews to gather participant insights. We then applied thematic analysis [12]. The first author conducted inductive coding to generate codes grounded in the data [13]. Then, these codes were discussed with another author to refine the coding, and we constructed broader themes and connected them to the quantitative findings [29]. We used the coding mainly to identify representative quotes that illustrate and contextualize the quantitative findings and to help inform the design implications by interpreting how participants experienced the AI during collaboration.

AI usage: We captured all interaction data in our database, including conversation logs, code typed line by line, and AI interventions. We also recorded the sessions on Zoom while both participants shared their screens. Because Zoom only records the screen of the most recent participant to share, we additionally recorded the first participant's screen using a screen recording app on our laptop (This was possible by toggling between shared screens and manually recording the desired view). To analyze AI usage, two researchers manually annotated each AI intervention. For each AI suggestion, we marked whether it was used or not. We defined "AI suggestion is used" as cases where participants (i) incorporated it into code (e.g., syntax fixes, code changes based on hints, or copy-pasted AI-generated code), (ii) discussed it with their partner, or (iii) asked follow-up questions to the AI. Annotations were made based on the interaction data in our database and aligned with Zoom-recorded events.

AI-generated code usage: In addition to tracking AI interventions, we also counted how many lines of code were fully generated by the AI versus written by participants. Syntax fixes were excluded since they involved partial human input. We counted a line as AI-generated if participants directly copy-pasted or manually wrote code from the AI suggestion into their solution fully.

Number of sub-tasks completed: To measure participants' programming performance, we followed prior approaches [18] by counting how many sub-tasks they completed. Each problem contained three sub-tasks. Sub-task 1 had a single target solution, while sub-tasks 2 and 3 could be solved either with a brute-force solution or with an optimized solution.

4.5.1 Statistical Analysis. All statistical analyses were conducted in R (version 4.5.1). Scores for the Collaborative Learning Scale (CLS) [36, 99] and the Social Presence Questionnaire (SPQ) [36, 65] were calculated by averaging across items within each scale to yield a final composite score. For continuous outcome variables from the survey measures, we used linear mixed-effects models [83]. Mixed-effects models accounted for the repeated-measures structure of the study, where each participant completed all three conditions and participants worked in pairs. The models included fixed effects for condition and random intercepts for participant ID and group ID to capture individual- and pair-level variability. The order of conditions was included as a fixed effect to account for potential ordering effects. For proportional outcome variables, such as the proportion of AI-generated code and the proportion of AI suggestions used, we used the same models with a binomial error distribution. These models incorporated the same random and fixed effects structure described above. For individual survey items with 7-point Likert scale responses, cumulative link mixed models [20] were used to appropriately account for the ordinal nature of the response variables. Post-hoc pairwise comparisons between experimental conditions were conducted using estimated marginal means [95]. Across all analyses, the threshold for statistical significance was set at $\alpha = 0.05$

5 Findings

5.1 HHAI Restores Collaborative Learning and Social Presence

Collaborating with a human partner restored the collaborative learning and social presence that participants felt were missing in human–AI dyads, with the strongest gains in the Shared AI condition. As shown in Figure 3, both collaborative learning and social presence ratings were higher in HHAI than in HAI, with Shared AI producing the largest effects.

5.1.1 Supporting Collaborative Learning. Participants emphasized that working with another human provided opportunities to learn in ways absent in HAI. P3 described how simply *"observing what my partner is doing"* helped them pick up new techniques. Others highlighted the value of explanation; P18 reflected that *"when I talked to my partner... she would catch my mistakes that I didn't even think twice about... just talking out loud [helps]."* Such accounts

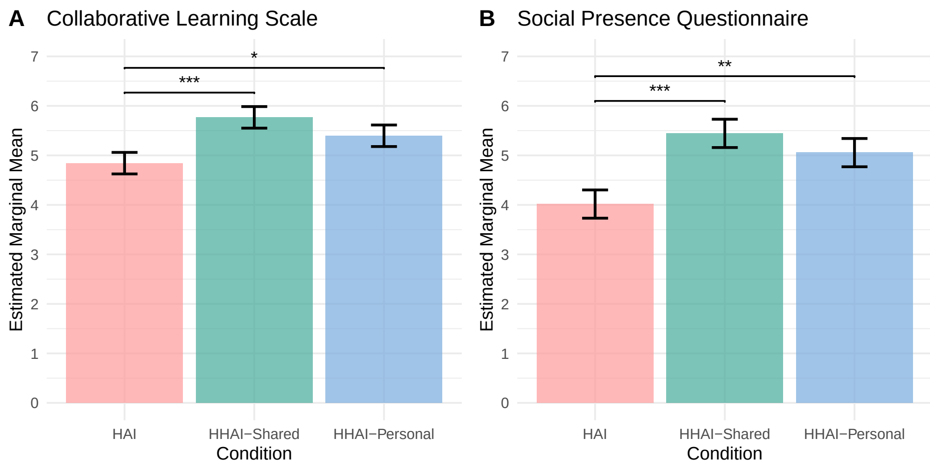

Figure 3: Collaborative Learning Scale (CLS) and Social Presence Questionnaire (SPQ): Both HHAI conditions showed significant differences compared to the HAI condition.

contrast with HAI sessions, where participants often stayed silent and engaged mainly with the AI's outputs.

Our analysis of the CLS questionnaire results revealed significant main effects of experimental condition on collaborative learning ratings ($F(2, 37) = 7.31$, $p < .01$). Compared to the human–AI baseline (HAI; $M = 4.63$), participants reported significantly higher collaborative learning in both the HHAI–Shared condition ($\beta = 0.93$, $SE = 0.24$, $t(37) = 3.81$, $p < .001$) and in the HHAI–Personal condition ($\beta = 0.55$, $SE = 0.24$, $t(37) = 2.28$, $p = .028$) with no significance in the experimental order ($p = .381$). Participants noted that the benefit of both HHAI conditions came less from efficiency and more from being pushed to explain and refine their thinking in dialogue with a peer.

5.1.2 Strengthening Social Presence. Participants consistently emphasized that collaborating with a peer felt more natural and socially engaging than interacting with AI alone. One participant explained:

> "I would say you can explain your thoughts to your partner in a more easier way. I think he can, like, try to understand what I'm trying to do, but conversing that to AI is harder for me. So, like, for me, I would just ask AI to, again, just solve it. So, sharing my thoughts loudly is easier to share with your partner, rather than AI." – P5

Other participants echoed this sentiment, describing AI-only work as *"awkward"* (P7) or *"one-sided"* (P12) compared to the back-and-forth with a peer. Social presence ratings showed a similar pattern of differences across conditions ($F(2, 37) = 9.45$, $p < .001$). Relative to HAI ($M = 4.13$), participants in both HHAI conditions reported significantly higher social presence (HHAI–Shared: $\beta = 1.43$, $SE = 0.37$, $t(37) = 3.90$, $p < .001$; HHAI–Personal: $\beta = 1.04$, $SE = 0.37$, $t(37) = 2.84$, $p = .007$) with no significance in the experimental order ($p = .764$). Post hoc comparisons (Figure 3.B) further indicated that HHAI–Shared did not differ from HHAI–Personal ($p > .05$).

Overall, participants contrasted HHAI with HAI to emphasize how a human peer restored the pedagogical and social value of collaboration. Quantitative results confirmed that both HHAI conditions improved collaborative learning and social presence, and that Shared AI magnified these effects.

5.2 HHAI Encourages More Accountable Use of AI

Working with a peer, especially in the Shared AI condition, increased participants' sense of responsibility for how they engaged with AI and corresponded with lower reliance on AI-generated code. As shown in Figure 4, responsibility ratings were highest in Shared AI, and the proportion of AI-generated code was lowest in HHAI compared to HAI.

5.2.1 Heightened Responsibility to Understand Before Applying AI Suggestions. Participants explained that the presence of a human partner made them more careful about how they applied AI output. For instance, P6 explained that simply knowing a partner could see her prompts made her feel more accountable and careful in how she engaged with the AI:

> "If I see that someone is watching my AI prompts, I would feel that I need to be more mature and try to understand the code first, rather than just copy-pasting." – P6

Similarly, P13 also emphasized how this visibility created a sense of social pressure that encouraged them to be more deliberate about how they used AI:

> "There's a little bit of pressure… I don't want to be looked bad or looked down upon by being reliant on AI when coding with another human." – P13

Our questionnaire results also reflected this pattern. Analysis using cumulative link mixed models revealed that the influence of a human partner on participants' sense of responsibility was most pronounced in the HHAI–Shared condition (Wald $\chi^2 = 24.4$, $p <$

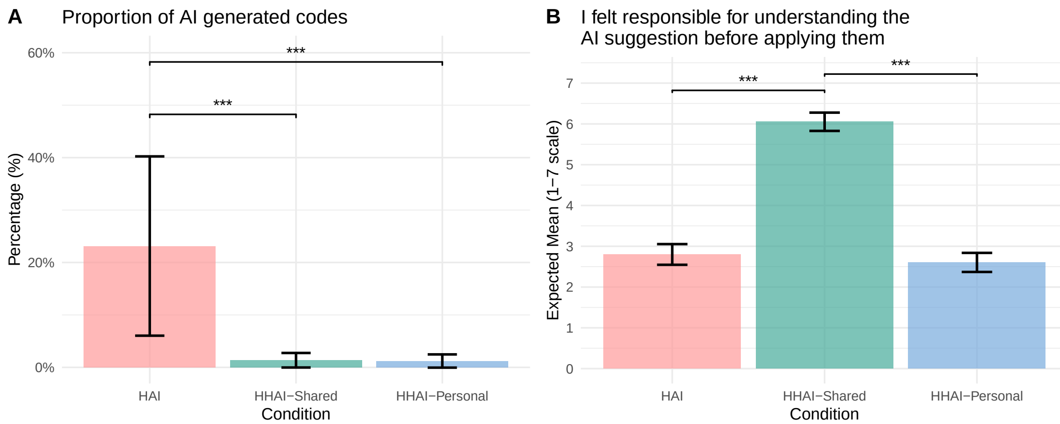

Figure 4: A) Proportion of AI-generated code; B) Perceived responsibility for understanding AI suggestions before applying

.001). As shown in Figure 4.B, HHAI–Shared produced the highest responsibility ratings (M = 6.05, 95% CI [5.61, 6.49]) compared to HAI (M = 2.80, 95% CI [2.30, 3.30]) or HHAI–Personal (M = 2.60, 95% CI [2.15, 3.06]). Post-hoc comparisons indicated that HHAI–Shared generated significantly higher responsibility ratings than both HAI ($p < .001$) and HHAI–Personal ($p < .001$). However, HAI and HHAI–Personal did not differ significantly ($p = .819$). The analysis also revealed a significant negative order effect, with participants reporting a decreasing sense of responsibility in later tasks ($\beta = -0.95$, $z = -2.88$, $p = .004$).

5.2.2 Decreased Reliance on AI-Generated Code. The same sense of accountability was evident in how often participants prompted the AI to generate code and how much of that output they incorporated into their final code submission. In HAI, a larger share of the submitted code came from the AI. Our analysis using a generalized linear mixed model with binomial distribution showed a strong effect of condition (Wald $\chi^2 = 48.4$, $p < .001$). As shown in Figure 4.A, the HAI condition showed the highest proportion of AI-generated code (23.1%, 95% CI [4.4%, 66.5%]), while both HHAI conditions demonstrated substantially lower reliance on AI-generated code: HHAI–Shared (1.4%, 95% CI [0.2%, 9.5%]) and HHAI–Personal (1.2%, 95% CI [0.2%, 8.9%]). Experimental order showed a significant effect ($\beta = 0.63$, $z = 2.44$, $p = .015$), indicating increased AI code usage in later sessions.

Many participants connected this reduction in AI-generated code to the social dynamics of working with a human peer. However, this reduction was not universal. In one pair (P15 + P16), 56% of their final code still came from the AI, despite being in the HHAI condition. P15 reflected on this reliance:

> "Because we have been so habituated to use AI rather than talking to a human, it just came out of habit." – P15

Their transcript also showed limited dialogue between the two partners, suggesting that weak human-to-human communication in collaborative programming undermined the accountability effect.

Overall, participants described feeling more responsible for understanding AI suggestions when collaborating with a partner, and this was reflected in both higher responsibility ratings and lower proportions of AI-generated code. Shared AI in particular heightened accountability by making prompts and responses visible to both collaborators.

5.3 Shared AI Supports Flow, While Personal AI Disrupts It

Shared AI interventions were experienced as the least disruptive and best aligned with group flow, while Personal AI fragmented collaboration and proactive suggestions were often ignored. As shown in Figure 5 B, Shared AI received significantly lower disruptiveness ratings, and participants were far more likely to adopt directly requested suggestions than proactive ones.

5.3.1 Sense of AI Disruptiveness. A cumulative link mixed model showed that participant ratings of AI disruptiveness differed substantially across conditions (Wald $\chi^2 = 23.5$, $p < .001$). HHAI–Shared received the lowest disruptiveness ratings (M = 2.71, 95% CI [2.05, 3.36]), while both HAI (M = 6.01, 95% CI [5.59, 6.42]) and HHAI–Personal (M = 6.18, 95% CI [5.76, 6.59]) were perceived as significantly more disruptive. As illustrated in Figure 5.B, post-hoc comparisons revealed that HHAI–Shared was rated as significantly less disruptive than both HAI ($p < .001$) and HHAI–Personal conditions ($p < .001$). However, HAI and HHAI–Personal did not differ significantly in disruptiveness ratings ($p = .820$). We also observed a significant order effect, where participants tended to rate the AI as more disruptive in later trials ($\beta = 0.77$, $z = 2.44$, $p = .015$).

Participants explained their differing perspectives on AI disruptiveness in Shared and Personal conditions by contrasting how Shared AI aligned with the group versus how Personal AI split attention between partners. P1 described Shared AI as integrated, noting that it *"actively listen[ed] and [gave] suggestions based on our thought processes."* (P1). Similarly, P3 observed that Personal AI encouraged side conversations, allowing her to check ideas privately before raising them with her partner. While useful individually,

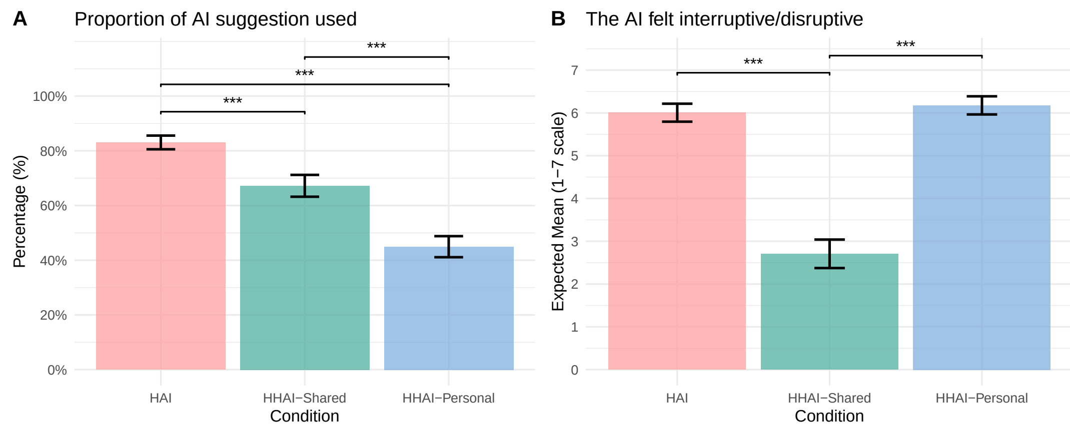

Figure 5: A) Proportion of AI suggestions being used (i.e., used in code, discussed, or followed up) based on annotated events; B) Questionnaire results on participants' experiences about AI proactivity

this parallel interaction disrupted the synchrony of the pair's dialogue. Participants also muted the AI at times when its personal interventions interrupted ongoing talk, as one explained:

> "When neither of us had access to the same information [in Personal AI]… it's like both of the users were using AI differently, so that's not like working towards the shared goal." – P6

5.3.2 Selective Uptake of AI Suggestions. Participants did not accept AI contributions at the same rate across conditions. A generalized linear mixed-effects model with a binomial distribution indicated significant differences across conditions (Wald $\chi^2 = 107.2$, $p < .001$). The HAI condition showed the highest uptake rate (83.1%, 95% CI [77.6, 87.4]), followed by HHAI–Shared (67.2%, 95% CI [59.0, 74.5]) and HHAI–Personal (45.0%, 95% CI [37.6, 52.6]). As shown in Figure 5.A, all pairwise comparisons were statistically significant ($p < .05$). Experimental order did not significantly influence suggestion usage ($\beta = -0.15$, $z = -1.62$, $p = .104$).

We observed that this difference was most pronounced for proactive interventions. In HAI, participants were more likely to use proactive input, but in HHAI pairs, many of those same interventions were filtered out unless they directly supported the ongoing conversation. When proactive input fit the moment, it was valued for introducing new perspectives:

> "We would think of one way… and then AI would jump in… and we'd be like, oh, we hadn't thought of this way before." – P2

Other participants emphasized the convenience or human-like quality of well-timed suggestions, as P4 described liking that he could *"just glance and keep going"*, and P13 noted that *"having something automatically listening felt very human-like."* Yet participants also ignored many proactive suggestions in HHAI when they were irrelevant, treating them as noise rather than support.

These uptake patterns add context to the disruptiveness ratings. In HAI, where participants relied heavily on the AI, both direct and proactive contributions were more readily incorporated ($OR = 2.39, p < .001$ vs. HHAI-Shared; $OR = 6.01, p < .001$ vs. HHAI-Personal; Figure 5 A). In contrast, in HHAI, participants filtered suggestions more carefully, often discarding proactive input that did not align with their joint reasoning between human partners. Suggestions that fit the flow of the peer dialogue were experienced as smoother and more likely to be adopted, while irrelevant or fragmented interventions felt disruptive and were often ignored.

5.4 Programming Performance Is Consistent Across Conditions

To examine whether task completion varied across the three conditions, we fitted a Poisson Generalized Linear Mixed Model (GLMM). The analysis showed no significant differences in the number of completed subtasks across the conditions (Fig. 6). Relative to the HAI baseline, neither the HHAI-Shared ($\beta = -0.11, SE = 0.35, p = 0.757$) nor the HHAI-Personal condition ($\beta = -0.21, SE = 0.36, p = 0.557$) differed significantly. Condition order also had no effect ($\beta = 0.17, SE = 0.18, p = 0.355$). Because session duration was held constant across conditions, task completion time was not included as an outcome measure.

Across all conditions, participants showed a similar pattern of progressing through the problems by completing one to two subtasks per condition (HAI: $Mean = 1.7$, $Md = 2$; HHAI-Shared: $Mean = 1.5$, $Md = 1.5$; HHAI-Personal: $Mean = 1.4$, $Md = 1$) (Fig. 6). Since some subtasks allowed both a brute-force and an optimized solution, we also examined the total amount of work performed, counting both completed subtasks and any optimized improvements. This combined measure showed a similarly consistent pattern across conditions (HAI: $Mean = 1.9$, $Md = 2$; HHAI-Shared: $Mean = 1.8$, $Md = 1.5$; HHAI-Personal: $Mean = 1.6$, $Md = 1$). In terms of optimizing their solutions, participants found the proactive AI suggestions were helpful, as P2 noted: *"the AI would jump in and provide more efficient ways of doing the problem."* However, when the AI suggested algorithms that participants had no prior familiarity with, some chose to disregard the suggestions, as P7 explained: *"sometimes the feedback was a little too advanced."* P8 similarly commented that *"the AI should know you better first…*

Category	**Description**	**Examples**	**HHAI-S (%)**	**HHAI-P (%)**	**HAI (%)**
Question (Seek Info)	Asking for unknown information, opinions, or help.	"What would the setup for a sliding window look like?" "What would I do after that?"	**12.06**	**14.22**	**35.10**
Question (Seek Confirm)	Seeking verification or agreement.	"That's what we want, right?" "Is that good?"	5.54	5.69	5.57
Answer / Reply	Direct, informative response.	"It takes the highest value." "We are returning the pair."	1.87	2.32	0.28
Proposing Strategy	Suggesting a high-level approach.	"You wanna try using a hash set?" "We could sort it."	4.16	5.31	3.90
Proposing Implementation	Suggesting specific code-level actions.	"To get the price, we can call the previous function." "You need to check for a chosen index as well."	17.53	15.04	17.55
Justifying Proposal	Explaining reasoning behind a proposal.	"If best streak is bigger than the sum, it doesn't matter." "That avoids recomputing values, so it's more efficient."	**5.34**	**2.62**	**0.00**
Think-Aloud	Verbalizing one's internal reasoning process.	"I think I need to do something with sum too." "Add step J to sum, then subtract start."	19.89	23.28	18.11
Read-Aloud	Reading text or code verbatim.	"The problem requires analyzing daily step counts." "Find the highest average step count."	1.52	0.90	1.11
Acknowledgment / Acceptance	Showing understanding or agreement.	"Yeah, that works, I guess." "Okay, code looks good."	**16.42**	**14.45**	**6.69**
Disagreement / Rejection	Rejecting a statement or proposal.	"But then that'd be O(n^2)." "I think our approach is just… not great."	2.84	2.77	1.11
Coordination / Turn-Taking	Managing roles or sequencing actions.	"Do you want to code that?" "I'm down for whatever."	5.27	4.42	3.06
Affective Expression	Expressing emotion or social signals.	"Oh, you're kidding." "Oh, shoot."	7.55	8.98	7.52

Table 2: Annotation categories, descriptions, examples, and the distribution of participant utterances across HHAI-Shared, HHAI-Personal, HAI conditions.

what are your current skills before giving suggestions," highlighting the need for AI suggestions calibrated to the user's level of coding knowledge when introducing advanced concepts.

5.5 Conversational Behaviors Differ Across Conditions

While programming performance remained stable across conditions, participants' conversational behaviors differed markedly depending on whether they were working alone with AI or collaborating with a partner in the HHAI conditions. Overall, participants in the HHAI conditions produced substantially more conversation than those in the HAI condition, as having a peer naturally created more opportunities for verbal exchange. The average number of participant utterances was higher in both HHAI settings (HHAI-Shared: *Mean* = 144.3, *Md* = 151; HHAI-Personal: *Mean* = 133.6, *Md* = 149) compared to the individual HAI condition (*Mean* = 35.9, *Md* = 35).

To examine how these conversational interactions differed qualitatively, we analyzed participants' utterances and computed the distribution of utterance types across the three conditions (Table 2). As shown in the table, the HAI condition was dominated by *Seek-Info Questions* (35.10%), many of which were short, procedural prompts such as "What do I do next?" or "What would the setup look like?" These questions were typically solution-seeking rather than reflective or explanatory, which suggests a form of problem-solving dependency on the AI. In contrast, both HHAI conditions showed higher proportions of conversation behaviors that involved

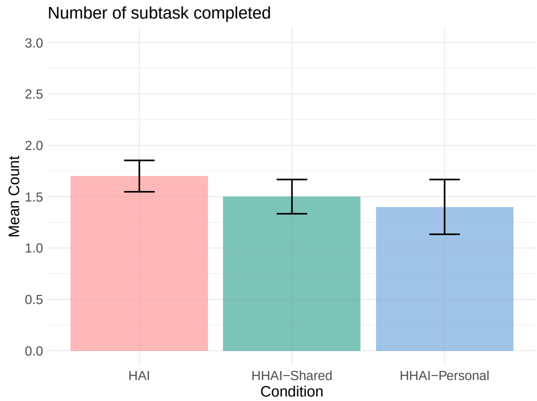

Figure 6: Mean number of subtasks completed by condition type.

confirming understanding, explaining reasoning, or coordinating next steps. For example, *Acknowledgment* was considerably higher in HHAI (14.44% to 16.42%) compared to HAI (6.69%), and *Justifying Proposal* appeared in HHAI (2.62% to 5.34%) but was absent in HAI (0%). Methodological details on our utterance analysis are provided in Appendix D.

To assess whether the pattern of utterances in Table 2 differed significantly across conditions, we conducted a Pearson Chi-square test with a Monte Carlo simulated p-value (20,000 permutations). The test was significant ($X^2 = 176.45$, $p < .001$), indicating that the overall pattern of conversational behaviors varied across the HAI, HHAI-Shared, and HHAI-Personal conditions. We then carried out post hoc analyses of category-level proportions (Table 3, Appendix D) to identify which utterance types contributed most to this effect. These analyses support the descriptive pattern in Table 2: *Seek-Info Questions* are more prevalent in HAI than in HHAI, whereas conversational utterances such as *Acknowledgment* and *Justifying Proposal* occur significantly more often in the two HHAI conditions than in HAI.

6 Discussion

6.1 Rethinking collaborative programming with AI as augmentation of human-human collaboration

Prior research on AI-assisted programming has predominantly focused on human–AI (HAI) dyads where an AI agent such as GitHub Copilot substitutes a human peer [7, 36, 63, 71, 74]. These studies consistently report faster task completion and higher code correctness, and highlight on-demand debugging and syntax assistance as concrete benefits of HAI [36]. However, participants from these studies also report a diminished sense of social presence compared with human–human settings [36]. The substitution of a human partner reflects a broader trend in AI-supported learning where systems are primarily valued for task efficiency [89, 110]. Our study challenges this substitution framing by showing that in programming practice, AI support is most valuable when it augments rather than replaces the collaborative benefits of a human partner. In both Shared and Personal HHAI conditions the presence of a human partner increased participants' sense of collaboration and social presence compared to the HAI setting (§5.1). Participants during HHAI settings described how it felt more natural to articulate their thinking and exchange ideas with a peer, which aligns with established accounts of the pedagogical and motivational benefits of collaborative programming [51, 54, 65, 99, 103].

These contrasting experiences between HAI and HHAI conditions situate our findings within the long-standing HCI discussion on augmentation versus automation [33, 68]. Automation, or replacement, frames technology as a substitute for human roles [5, 85], which is reflected in the HAI condition. For instance, in the HAI condition, participants often prompted the AI with direct solution-seeking questions, effectively shifting the sense-making steps of the problem-solving process to the AI. Augmentation, by contrast, frames technology as amplifying human agency [33, 43, 68],which was reflected in the HHAI conditions where participants engaged in substantially more conversational activity and naturally articulated their reasoning to a partner as they worked through the problem together. Our findings show that in collaborative programming, effective augmentation means designing AI to reinforce the collaborative mechanisms that human partners already provide. Studies in learning sciences show that a key reason another human matters is that human peers create interactional conditions [58, 103, 112, 120] that AI does not. With a peer present, there is social accountability [58] and an expectation to articulate thinking so that the partner can follow along [8].

6.2 Influence of Human Presence on Responsible Use of AI for Learning

Recent work in education shows that when learners outsource problem-solving to AI, it often reduces the reflective processes that underlie critical thinking [61, 64]. Recent studies in HCI similarly raise concerns about how overreliance on AI can lead to reduced learning engagement [3, 37, 40, 87]. For example, a study by Fan et al. [37] found that students using ChatGPT exhibited a greater tendency toward "metacognitive laziness," which reduced engagement in learning and increased dependence on technology. Similarly, Pu et al. [87] showed that AI programming support can cause overreliance and a lack of code understanding. Our study addresses these concerns by examining how the presence of another human partner can moderate responsible engagement with AI in collaborative programming contexts.

Our findings show that human–human–AI triads reduced overreliance, particularly in the use of AI-generated code (§5.2). This suggests that the human partners did not blindly outsource the task to the AI. Instead, the presence of a human partner encouraged them to discuss solutions and use AI suggestions more responsibly. This shift was also reflected in participants' conversational behavior. In the HHAI conditions, participants produced more *Acknowledgment* and *Justifying Proposal* utterances, which reflected coordinated checks and rationalization for next-steps. This effect became even more pronounced when shifting from personal AI to shared AI, which produced a notable increase in how participants felt accountable for understanding AI suggestions before applying them. As participants explained, simply knowing that a peer could see their interaction with the AI made them more intentional about treating AI as a learning resource rather than copy-pasting code.

This dynamic can be understood through the lens of Socially Shared Regulation of Learning (SSRL) [48, 58]. SSRL emphasizes how learners regulate goals and strategies collectively through mutual monitoring and accountability [48, 58]. In collaborative settings, regulation does not rest solely on individual cognition but develops through group interactions, where learners actively shape each other's approaches and decisions [48]. In our findings, accountability was not enforced externally but was socially constructed through peer presence (§5.2). Knowing that a partner was observing their interaction with the AI made participants use AI more responsibly. This finding also connects to recent HCI work that explores when and whether AI outputs should remain private aids for individuals and when they should be shared as resources that peers can jointly monitor [38, 88]. Our results contribute to this discussion by showing that shared visibility of AI suggestions can embed accountability into the interaction, making responsible use of AI in collaborative programming learning contexts more likely.

However, adding a human partner did not always automatically encourage responsible use of AI (§5.2). In some cases, when human-to-human collaboration was not effective (e.g., when both users communicated very little) and both participants were already used to relying heavily on AI, the presence of a human peer did little to reduce overreliance on AI. This suggests that the benefits of triadic collaboration are not automatic but depend on the quality of interaction that participants bring into the collaboration. When mutual accountability is weak, peers may reinforce each other's reliance on AI rather than moderating it [96]. Future work in collaborative learning systems in HHAI contexts should consider scaffolding productive human-to-human peer interaction so that the presence of a human partner can reliably foster more responsible engagement with AI.

6.3 Design Implications for Human-Human-AI Triadic Programming

6.3.1 Implementing Shared AI to Support Collective Learning. Our study shows the benefits of having an AI agent shared by participants, which supports collective learning and reduces overreliance on AI (§5.1, §5.2). Having shared AI in HHAI triads also enabled participants to engage in various ways of learning. For example, some participants reported that they learned by actively explaining their thinking approach to their partners while thinking aloud (P18), with the AI providing proactive suggestions. Others noted that they learned by observing their partner's actions (P3), particularly when the AI offered suggestions that one partner then applied in the code. Similarly, some participants asked their partners to help them better understand the AI's suggestions (P5, P6). As such, having a shared AI was important because *"see[ing] the same output of the AI is helpful, [otherwise, my partner] could be referencing something that I don't see," (P9)"*

6.3.2 Controlled Personal and Shared Discussion. In addition to having a shared AI, sometimes a personal AI can be useful when participants want to explore ideas that may not align with the current conversation. As P3 mentioned: *"if I'm thinking something that may or may not be relevant, I can just use a personal AI to get a clear picture of it, and then communicate with the partner about"*. However, when participants used a personal AI, there were moments when they wanted to share what they had discussed with the AI but then needed to *"describe [the personal AI suggestion] ... [and mentioning that] it could be really good if you have an option to just share [that]" (P10).* Hence, future work can implement a feature where users can simply click to share parts of the conversation with the AI, aligning with prior work on sharing AI suggestions [38].

6.3.3 Proactive AI Intervention to Support Collaboration. In line with prior work that highlights the benefits of proactive AI [18, 55, 87], our participants also acknowledged the value of proactive AI assistance in supporting the learning process (P1, P2, P13). Compared to prior work on proactive programming assistants [18, 87], we focused on AI proactivity in conversational interactions, since collaborative programming involves the exchange of ideas between humans [62]. Participants mentioned that having a proactive AI felt like *"it was listening to us, it would jump in and provide more, like, efficient ways of doing the problem" (P2).* As such, this proactive AI behavior can provide *"unexpected [learning] moments" (P2)*, where participants learned something new without having to manually ask the AI. Building upon the potential of proactive AI, future work can explore more agentic AI [92] that can autonomously decide when to intervene, determine what type of support to provide, and adapt its behavior dynamically to the ongoing conversation or shared goals.

6.3.4 Pedagogically Driven AI to Facilitate Learning. In general, participants appreciated when the AI provided just enough support for the learning process without revealing the solution too early (P11, P12, P13, P14, P18). As P12 mentioned: *"It allows you to learn regardless of how you [ask] it. It would just give you hints, or the conceptual background on how you would use it ... which I thought was really good."* One participant even suggested reducing the AI's expertise. As P9 explained: *"[The AI] was really trying to give me the answer (hints), [but] I wanted to work through it myself and give it a try."* He further described his preference of adding an option to have a *"partner-based personality [AI]"* where the AI would be at the same level as the human and primarily act as a discussion partner rather than a teacher. As such, this aligns with prior work highlighting the benefits of offering multiple options to customize AI personas [47].

6.3.5 Multimodal and Integrated Interface for Collaborative Programming with AI. Participants generally appreciated the integrated interface that combined AI access with the code editor, which made it easier to access AI support. The multimodal interaction, such as simply clicking to trigger code analysis, allowed participants to get help quickly without having to manually prompt the AI. Automated voice transcription also enabled the AI to continuously listen to the conversation and proactively provide contextually relevant suggestions (P1, P2, P13). However, participants generally disliked the AI voice output and often turned it off early in the session. As P6 mentioned: *"I think one problem with the AI [voice] is that if it starts to speak, it will keep on speaking ... maybe you [should be able to] barge in, and then, like, I can start speaking again."* Even though the

AI began speaking while the user was idle, once it started it would continue until finishing its sentences, which sometimes prevented participants from smoothly resuming their conversation with their partner. Hence, there is a need for a better turn-taking mechanism that adapts to the flow of conversation [98].

7 Limitations

We first acknowledge the limitation of not having a human–human collaborative programming baseline (no-AI). Although our findings suggest that the AI supported collaboration in meaningful ways, the absence of a human–human baseline limits our ability to determine the extent of that contribution. Without a no-AI comparison, we cannot fully assess how much the AI improved collaboration beyond what two humans might achieve on their own, or whether it may have disrupted the collaboration instead.

Secondly, our study involved college students from a single institution working on LeetCode-style problems. While this allows us to explore initial findings in a controlled setting, it limits how far the results can generalize to other programming contexts or learner populations. The single-session format also limits us from understanding longer-term development or how triadic collaboration might evolve over time. Future work could conduct longitudinal studies and explore a broader range of programming tasks to better capture how triadic collaboration unfolds across different contexts and over extended periods. It could also examine different populations, such as professional software engineers, to understand how these dynamics operate in more varied and professional settings.

Thirdly, all participant pairs consisted of students who already knew their partner and registered together. While we did not measure the closeness of these relationships, participants were asked to recruit someone they knew, so pairs were not strangers and were not randomly assigned. This familiarity may have influenced how they collaborated, as prior work shows that social relationships can shape collaborative programming dynamics [104, 118]. As a result, our findings primarily reflect triadic collaboration among partners with some prior familiarity. Future work can examine how these dynamics play out among pairs who have little or no prior relationship.

Lastly, although the experimental design used full counterbalancing, three consistent order effects emerged across conditions. As sessions progressed, participants increasingly relied on AI-generated code, reported less responsibility for understanding AI suggestions, and perceived the AI as more disruptive. Because counterbalancing ensured that order was orthogonal to experimental condition, these trends do not confound the primary comparisons across conditions. These effects can be interpreted as general patterns of how user behavior and perception evolve during a prolonged, 90-minute programming session with an AI assistant. Such patterns may reflect a combination of adaptation and fatigue effects. Early in the session, participants may have been more cautious, carefully checking AI output and assuming greater responsibility for integrating suggestions. Over time, however, familiarity with the tool and cognitive fatigue may have fostered greater reliance on AI code, accompanied by reduced vigilance and heightened perceptions of the AI as interruptive. These dynamics highlight that the temporal trajectory of interaction can shape both behavioral reliance on AI and subjective evaluations of its role, independent of the specific interaction scenario.

8 Conclusion

Our study explored human-human-AI triadic programming, where an AI agent serves as an additional collaborator rather than a substitute for a human partner. Through a study with 20 participants, we show that this triadic collaboration can enhance collaborative learning and social presence compared to a human–AI baseline, while maintaining similar programming performance. Our findings also highlight how human partners encourage more accountable AI use, primarily reducing reliance on AI-generated code. This effect becomes amplified when shifting from personal AI to shared AI support, which further increases participants' sense of responsibility to understand AI suggestions before applying them. Taken together, our findings suggest that the value of AI in educational settings extends beyond efficiency gains. By embedding AI within peer collaboration, systems can strengthen the social and pedagogical dynamics that underpin effective learning. Building on these insights, we propose design implications for future systems: make AI outputs visible to peers to scaffold accountability, align proactive interventions with conversational flow, and reinforce the social and pedagogical benefits of working with others.

References

[1] Adeola Adeliyi, Michel Wermelinger, Karen Kear, and Jon Rosewell. 2021. Investigating remote pair programming in part-time distance education. In *Proceedings of the 2021 Conference on United Kingdom & Ireland Computing Education Research*. 1–7.

[2] Oyebola Olusola Ayeni, Nancy Mohd Al Hamad, Onyebuchi Nneamaka Chisom, Blessing Osawaru, and Ololade Elizabeth Adewusi. 2024. AI in education: A review of personalized learning and educational technology. *GSC Advanced Research and Reviews* 18, 2 (2024), 261–271.

[3] Long Bai, Xiangfei Liu, and Jiacan Su. 2023. ChatGPT: The cognitive effects on learning and memory. *Brain-X* 1, 3 (2023), e30.

[4] David Baidoo-Anu and Leticia Owusu Ansah. 2023. Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. *Journal of AI* 7, 1 (2023), 52–62.

[5] Lisanne Bainbridge. 1983. Ironies of automation. In *Analysis, design and evaluation of man–machine systems*. Elsevier, 775–779.

[6] Leland Beck and Alexander Chizhik. 2013. Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. *ACM Transactions on Computing Education (TOCE)* 13, 3 (2013), 1–21.

[7] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 500–506.

[8] Andrew Begel and Nachiappan Nagappan. 2008. Pair programming: what's in it for me?. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 120–128.

[9] Sophie Berretta, Alina Tausch, Greta Ontrup, Björn Gilles, Corinna Peifer, and Annette Kluge. 2023. Defining human-AI teaming the human-centered way: a scoping review and network analysis. *Frontiers in Artificial Intelligence* 6 (2023), 1250725.

[10] Maxwell Bigman, Ethan Roy, Jorge Garcia, Miroslav Suzara, Kaili Wang, and Chris Piech. 2021. PearProgram: A more fruitful approach to pair programming. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 900–906.

[11] Adrian Bolboacă. 2021. *Practical Remote Pair Programming: Best practices, tips, and techniques for collaborating productively with distributed development teams*. Packt Publishing Ltd.

[12] Virginia Braun and Victoria Clarke. 2021. Thematic analysis: A practical guide. (2021).

[13] Virginia Braun and Victoria Clarke. 2023. Toward good practice in thematic analysis: Avoiding common problems and be (com) ing a knowing researcher. *International journal of transgender health* 24, 1 (2023), 1–6.

[14] Sallyann Bryant, Pablo Romero, and Benedict du Boulay. 2008. Pair programming and the mysterious role of the navigator. *International Journal of Human-Computer Studies* 66, 7 (2008), 519–529.

[15] Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. The importance of producing shared code through pair programming. In *Proceedings of the 49th ACM technical symposium on computer science education*. 765–770.

[16] Feng Chen. 2022. Human-AI cooperation in education: human in loop and teaching as leadership. *Journal of Educational Technology and Innovation* 2, 1 (2022).

[17] Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024. Is a large language model a good annotator for event extraction?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 17772–17780.

[18] Valerie Chen, Alan Zhu, Sebastian Zhao, Hussein Mozannar, David Sontag, and Ameet Talwalkar. 2025. Need help? designing proactive ai assistants for programming. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–18.

[19] Chun-Wei Chiang, Zhuoran Lu, Zhuoyan Li, and Ming Yin. 2024. Enhancing ai-assisted group decision making through llm-powered devil's advocate. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 103–119.

[20] Rune Haubo B Christensen. 2018. Cumulative link models for ordinal regression with the R package ordinal. *Submitted in J. Stat. Software* 35 (2018), 1–46.

[21] Karen Sunday Cockrell, Julie A Hughes Caplow, and Joe F Donaldson. 2000. A context for learning: Collaborative groups in the problem-based learning environment. *The Review of Higher Education* 23, 3 (2000), 347–363.

[22] Katherine M Collins, Ilia Sucholutsky, Umang Bhatt, Kartik Chandra, Lionel Wong, Mina Lee, Cedegao E Zhang, Tan Zhi-Xuan, Mark Ho, Vikash Mansinghka, et al. 2024. Building machines that learn and think with people. *Nature human behaviour* 8, 10 (2024), 1851–1863.

[23] Taufiq Daryanto, Xiaohan Ding, Lance T Wilhelm, Sophia Stil, Kirk McInnis Knutsen, and Eugenia H Rho. 2025. Conversate: Supporting reflective learning in interview practice through interactive simulation and dialogic feedback. *Proceedings of the ACM on Human-Computer Interaction* 9, 1 (2025), 1–32.

[24] Taufiq Daryanto, Sophia Stil, Xiaohan Ding, Daniel Manesh, Sang Won Lee, Tim Lee, Stephanie Lunn, Sarah Rodriguez, Chris Brown, and Eugenia Rho. 2025. Designing Conversational AI to Support Think-Aloud Practice in Technical Interview Preparation for CS Students. *arXiv preprint arXiv:2507.14418* (2025).

[25] Adelson de Araujo, Pantelis M Papadopoulos, Susan McKenney, and Ton de Jong. 2024. A learning analytics-based collaborative conversational agent to foster productive dialogue in inquiry learning. *Journal of computer assisted learning* 40, 6 (2024), 2700–2714.

[26] José M De la Torre-Ruiz, Vera Ferrón-Vílchez, and Natalia Ortiz-de Mandojana. 2014. Team decision making and individual satisfaction with the team. *Small group research* 45, 2 (2014), 198–216.

[27] Giuseppe Desolda, Andrea Esposito, Francesco Greco, Cesare Tucci, Paolo Buono, and Antonio Piccinno. 2025. Understanding User Mental Models in AI-Driven Code Completion Tools: Insights from an Elicitation Study. *arXiv preprint arXiv:2502.02194* (2025).

[28] Xiaohan Ding, Kaike Ping, Buse Çarık, and Eugenia Rho. 2025. A Multi-Level Benchmark for Causal Language Understanding in Social Media Discourse. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 28764–28778.

[29] Mary Dixon-Woods, Shona Agarwal, David Jones, Bridget Young, and Alex Sutton. 2005. Synthesising qualitative and quantitative evidence: a review of possible methods. *Journal of health services research & policy* 10, 1 (2005), 45–53.

[30] James Dominic, Brock Tubre, Charles Ritter, Jada Houser, Colton Smith, and Paige Rodeghero. 2020. Remote pair programming in virtual reality. In *2020 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 406–417.

[31] Amy C Edmondson and Tijs Besieux. 2021. Reflections: voice and silence in workplace conversations. *Journal of Change Management* 21, 3 (2021), 269–286.

[32] Satu Elo and Helvi Kyngäs. 2008. The qualitative content analysis process. *Journal of advanced nursing* 62, 1 (2008), 107–115.

[33] Douglas C Engelbart. 2023. Augmenting human intellect: A conceptual framework. In *Augmented education in the global age*. Routledge, 13–29.

[34] Sheikh Abdulaziz Fahad, Said A Salloum, and Khaled Shaalan. 2024. The role of ChatGpt in knowledge sharing and collaboration within digital workplaces: a systematic review. *Artificial intelligence in education: The power and dangers of ChatGPT in the classroom* (2024), 259–282.

[35] Hannah Fahnenstich, Tobias Rieger, and Eileen Roesler. 2024. Trusting under risk–comparing human to AI decision support agents. *Computers in Human Behavior* 153 (2024), 108107.

[36] Guangrui Fan, Dandan Liu, Rui Zhang, and Lihu Pan. 2025. The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches. *International Journal of STEM Education* 12, 1 (2025), 16.

[37] Yizhou Fan, Luzhen Tang, Huixiao Le, Kejie Shen, Shufang Tan, Yueying Zhao, Yuan Shen, Xinyu Li, and Dragan Gašević. 2025. Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance. *British Journal of Educational Technology* 56, 2 (2025), 489–530.

[38] Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. 2024. Coprompt: Supporting prompt sharing and referring in collaborative natural language programming. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–21.

[39] Aashish Ghimire and John Edwards. 2024. Coding with ai: How are tools like chatgpt being used by students in foundational programming courses. In *International Conference on Artificial Intelligence in Education*. Springer, 259–267.

[40] Michail Giannakos, Roger Azevedo, Peter Brusilovsky, Mutlu Cukurova, Yannis Dimitriadis, Davinia Hernandez-Leo, Sanna Järvelä, Manolis Mavrikis, and Bart Rienties. 2025. The promise and challenges of generative AI in education. *Behaviour & Information Technology* 44, 11 (2025), 2518–2544.

[41] GitHub. 2025. GitHub Copilot. https://github.com/features/copilot Accessed: 2025-09-10.

[42] Gabriel Enrique Gonzalez, Dario Andres Silva Moran, Stephanie Houde, Jessica He, Steven I Ross, Michael J Muller, Siya Kunde, and Justin D Weisz. 2024. Collaborative Canvas: A Tool for Exploring LLM Use in Group Ideation Tasks.. In *IUI Workshops*.

[43] Jonathan Grudin. 2002. Computer-supported cooperative work: History and focus. *Computer* 27, 5 (2002), 19–26.

[44] X Gu and H Cai. 2021. Predicting the future of artificial intelligence and its educational impact: A thought experiment based on social science fiction. *Educational research* 42, 05 (2021), 137–147.

[45] Philip J Guo. 2023. Six Opportunities for scientists and engineers to learn programming using AI Tools such as ChatGPT. *Computing in Science & Engineering* 25, 3 (2023), 73–78.

[46] Rajan Das Gupta, Md Tanzib Hosain, Muhammad Firoz Mridha, and Salah Uddin Ahmed. 2025. Multimodal Programming in Computer Science with Interactive Assistance Powered by Large Language Model. In *International Conference on Human-Computer Interaction*. Springer, 59–69.

[47] Juhye Ha, Hyeon Jeon, Daeun Han, Jinwook Seo, and Changhoon Oh. 2024. CloChat: Understanding how people customize, interact, and experience personas in large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–24.

[48] Allyson Hadwin, Sanna Järvelä, and Mariel Miller. 2017. Self-regulation, co-regulation, and shared regulation in collaborative learning environments. In *Handbook of self-regulation of learning and performance*. Routledge, 83–106.

[49] Vera Hagemann, Michèle Rieth, Amrita Suresh, and Frank Kirchner. 2023. Human-AI teams—Challenges for a team-centered AI at work. *Frontiers in Artificial Intelligence* 6 (2023), 1252897.

[50] Philipp Haindl and Gerald Weinberger. 2024. Students' experiences of using ChatGPT in an undergraduate programming course. *IEEE Access* 12 (2024), 43519–43529.

[51] Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. 2011. Pair programming in education: A literature review. *Computer Science Education* 21, 2 (2011), 135–173.

[52] Marijn Haverbeke. 2007–2017. *CodeMirror*. https://codemirror.net/

[53] Jessica He, Stephanie Houde, Gabriel E Gonzalez, Darío Andrés Silva Moran, Steven I Ross, Michael Muller, and Justin D Weisz. 2024. AI and the Future of Collaborative Work: Group Ideation with an LLM in a Virtual Canvas. In *Proceedings of the 3rd Annual Meeting of the Symposium on Human-Computer Interaction for Work*. 1–14.

[54] Carol Hostetter. 2013. Community matters: Social presence and learning outcomes. *Journal of the Scholarship of Teaching and Learning* 13, 1 (2013), 77–86.

[55] Stephanie Houde, Kristina Brimijoin, Michael Muller, Steven I Ross, Dario Andres Silva Moran, Gabriel Enrique Gonzalez, Siya Kunde, Morgan A Foreman, and Justin D Weisz. 2025. Controlling AI Agent Participation in Group Conversations: A Human-Centered Approach. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*. 390–408.

[56] Janet Hughes, Ann Walshe, Bobby Law, and Brendan Murphy. 2020. Remote pair programming. In *12th International Conference on Computer Supported Education*. SciTePress, 476–483.

[57] Hanna Hulkko and Pekka Abrahamsson. 2005. A multiple case study on the impact of pair programming on product quality. In *Proceedings of the 27th international conference on Software engineering*. 495–504.

[58] Sanna Järvelä, Andy Nguyen, and Allyson Hadwin. 2023. Human and artificial intelligence collaboration for socially shared regulation in learning. *British Journal of Educational Technology* 54, 5 (2023), 1057–1076.

[59] Umer Javed, Atiya Rohilla, Ghazal Adnan, and Naveen Taj. 2025. Exploring how AI can be used to Promote Collaboration in group Project reduce Conflict in Team Dynamics and Enhance Cooperative Learning Experiences. *Review of Applied Management and Social Sciences* 8, 1 (2025), 129–141.

[60] Hyoungwook Jin, Seonghee Lee, Hyungyu Shin, and Juho Kim. 2024. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–28.

[61] Binny Jose, Jaya Cherian, Alie Molly Verghis, Sony Mary Varghise, Mumthas S, and Sibichan Joseph. 2025. The cognitive paradox of AI in education: between enhancement and erosion. *Frontiers in Psychology* 16 (2025), 1550621.

[62] RK Kavitha and MS Irfan Ahmed. 2015. Knowledge sharing through pair programming in learning environments: An empirical study. *Education and Information Technologies* 20, 2 (2015), 319–333.

[63] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI conference on human factors in computing systems*. 1–23.

[64] Nataliya Kosmyna, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. 2025. Your brain on chatgpt: Accumulation of cognitive debt when using an ai assistant for essay writing task. *arXiv preprint arXiv:2506.08872* 4 (2025).

[65] Karel Kreijns, Paul A Kirschner, Wim Jochems, and Hans Van Buuren. 2011. Measuring perceived social presence in distributed learning groups. *Education and Information Technologies* 16, 4 (2011), 365–381.

[66] Aayush Kumar, Yasharth Bajpai, Sumit Gulwani, Gustavo Soares, and Emerson Murphy-Hill. 2025. Sharp Tools: How Developers Wield Agentic AI in Real Software Engineering Tasks. *arXiv e-prints* (2025), arXiv–2506.

[67] LeetCode. 2025. LeetCode. http://leetcode.com/. Accessed: 2025-08-19.

[68] Joseph CR Licklider. 2008. Man-computer symbiosis. *IRE transactions on human factors in electronics* 1 (2008), 4–11.

[69] Shu Lin, Na Meng, Dennis Kafura, and Wenxin Li. 2021. PDL: scaffolding problem solving in programming courses. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 185–191.

[70] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J Malan. 2024. 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. 1*. 750–756.

[71] Wenhan Lyu, Yimeng Wang, Yifan Sun, and Yixuan Zhang. 2025. Will Your Next Pair Programming Partner Be Human? An Empirical Evaluation of Generative AI as a Collaborative Teammate in a Semester-Long Classroom Setting. In *Proceedings of the Twelfth ACM Conference on Learning@ Scale*. 83–94.

[72] Qianou Ma, Weirui Peng, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. 2024. What you say= what you want? Teaching humans to articulate requirements for LLMs. *arXiv preprint arXiv:2409.08775* (2024).

[73] Qianou Ma, Hua Shen, Kenneth Koedinger, and Sherry Tongshuang Wu. 2024. How to teach programming in the ai era? using llms as a teachable agent for debugging. In *International Conference on Artificial Intelligence in Education*. Springer, 265–279.

[74] Qianou Ma, Tongshuang Wu, and Kenneth Koedinger. 2023. Is ai the better programming partner? human-human pair programming vs. human-ai pair programming. *arXiv preprint arXiv:2306.05153* (2023).

[75] MDN Web Docs. 2024. WebSockets API. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. Accessed: 2025-08-19.

[76] Microsoft. 2025. Visual Studio Live Share. https://visualstudio.microsoft.com/services/live-share/. Accessed: 2025-08-19.

[77] Nailia Mirzakhmedova, Marcel Gohsen, Chia Hao Chang, and Benno Stein. 2024. Are Large Language Models Reliable Argument Quality Annotators?. In *Conference on Advances in Robust Argumentation Machines*. Springer, 129–146.

[78] Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P Bigham. 2025. CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–15.

[79] Michael Muller, Stephanie Houde, Gabriel Gonzalez, Kristina Brimijoin, Steven I Ross, Dario Andres Silva Moran, and Justin D Weisz. 2024. Group brainstorming with an ai agent: Creating and selecting ideas. In *International conference on computational creativity*. 10.

[80] Verena Nitsch, Vera Rick, Annette Kluge, and Uta Wilkens. 2024. Human-centered approaches to AI-assisted work: the future of work? *Zeitschrift für Arbeitswissenschaft* 78, 3 (2024), 261–267.

[81] Timothy J Nokes-Malach, J Elizabeth Richey, and Soniya Gadgil. 2015. When is it better to learn together? Insights from research on collaborative learning. *Educational Psychology Review* 27, 4 (2015), 645–656.

[82] John T Nosek. 1998. The case for collaborative programming. *Commun. ACM* 41, 3 (1998), 105–108.

[83] Ann L Oberg and Douglas W Mahoney. 2007. Linear mixed effects models. *Topics in biostatistics* (2007), 213–234.

[84] OpenAI. 2025. ChatGPT (GPT-5). https://chat.openai.com/. ChatGPT.

[85] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans* 30, 3 (2000), 286–297.

[86] Tung Phung, Victor-Alexandru Pădurean, José Cambronero, Sumit Gulwani, Tobias Kohn, Rupak Majumdar, Adish Singla, and Gustavo Soares. 2023. Generative AI for programming education: benchmarking ChatGPT, GPT-4, and human tutors. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2*. 41–42.

[87] Kevin Pu, Daniel Lazaro, Ian Arawjo, Haijun Xia, Ziang Xiao, Tovi Grossman, and Yan Chen. 2025. Assistance or disruption? exploring and evaluating the design and trade-offs of proactive ai programming support. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–21.

[88] Jeba Rezwana and Mary Lou Maher. 2022. Understanding user perceptions, collaborative experience and user engagement in different human-AI interaction designs for co-creative systems. In *Proceedings of the 14th Conference on Creativity and Cognition*. 38–48.

[89] Lianne Roest, Hieke Keuning, and Johan Jeuring. 2024. Next-step hint generation for introductory programming using large language models. In *Proceedings of the 26th Australasian Computing Education Conference*. 144–153.

[90] Norsaremah Salleh, Emilia Mendes, and John Grundy. 2010. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering* 37, 4 (2010), 509–525.

[91] Borhan Samei, Haiying Li, Fazel Keshtkar, Vasile Rus, and Arthur C Graesser. 2014. Context-based speech act classification in intelligent tutoring systems. In *International conference on intelligent tutoring systems*. Springer, 236–241.

[92] Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468* (2025).

[93] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 27–43.

[94] Julia Schenk, Lutz Prechelt, and Stephan Salinger. 2014. Distributed-pair programming can work well and is not just distributed pair-programming. In *Companion Proceedings of the 36th International Conference on Software Engineering*. 74–83.

[95] Shayle R Searle, F Michael Speed, and George A Milliken. 1980. Population marginal means in the linear model: an alternative to least squares means. *The American Statistician* 34, 4 (1980), 216–221.

[96] Anne C Sinclair, Samantha A Gesel, and Christopher J Lemons. 2019. The effects of peer-assisted learning on disruptive behavior and academic engagement. *Journal of Positive Behavior Interventions* 21, 4 (2019), 238–248.

[97] Nikhil Singh, Lucy Lu Wang, and Jonathan Bragg. 2024. Figura11y: Ai assistance for writing scientific alt text. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 886–906.

[98] Gabriel Skantze and Bahar Irfan. 2025. Applying general turn-taking models to conversational human-robot interaction. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 859–868.

[99] Hyo-Jeong So and Thomas A Brush. 2008. Student perceptions of collaborative learning, social presence and satisfaction in a blended learning environment: Relationships and critical factors. *Computers & education* 51, 1 (2008), 318–336.

[100] Zhensu Sun, Xiaoning Du, Fu Song, Shangwen Wang, Mingze Ni, Li Li, and David Lo. 2025. Don't complete it! Preventing unhelpful code completion for productive and sustainable neural code completion systems. *ACM Transactions on Software Engineering and Methodology* 34, 1 (2025), 1–22.

[101] Xiaohang Tang, Sam Wong, Kevin Pu, Xi Chen, Yalong Yang, and Yan Chen. 2024. VizGroup: An AI-Assisted Event-Driven System for Real-Time Collaborative Programming Learning Analytics. *arXiv preprint arXiv:2404.08743* (2024).

[102] Jessica To and David Carless. 2016. Making productive use of exemplars: Peer discussion and teacher guidance for positive transfer of strategies. *Journal of Further and Higher Education* 40, 6 (2016), 746–764.

[103] Eva Vass and Karen Littleton. 2010. Peer collaboration and learning in the classroom. *International handbook of psychology in education* (2010), 105–135.

[104] Maureen Villamor and MM Rodrigo. 2018. Do friends collaborate and perform better?: A pair program tracing and debugging eye-tracking experiment. In *18th Philippine Computing Science Congress*. 9–16.

[105] Dakuo Wang, Elizabeth Churchill, Pattie Maes, Xiangmin Fan, Ben Shneiderman, Yuanchun Shi, and Qianying Wang. 2020. From human-human collaboration to Human-AI collaboration: Designing AI systems that can work together with people. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*. 1–6.

[106] Deliang Wang, Dapeng Shan, Ran Ju, Ben Kao, Chenwei Zhang, and Gaowei Chen. 2024. Investigating dialogic interaction in K12 online one-on-one mathematics tutoring using AI and sequence mining techniques. *Education and Information Technologies* (2024), 1–26.

[107] Haoming Wang, Chengliang Wang, Zhan Chen, Fa Liu, Chunjia Bao, and Xianlong Xu. 2025. Impact of AI-agent-supported collaborative learning on the learning outcomes of University programming courses. *Education and Information Technologies* (2025), 1–33.

[108] Tianjia Wang, Daniel Vargas Díaz, Chris Brown, and Yan Chen. 2023. Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives. In *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 92–102.
[109] Justin D Weisz, Shraddha Vijay Kumar, Michael Muller, Karen-Ellen Browne, Arielle Goldberg, Katrin Ellice Heintze, and Shagun Bajpai. 2025. Examining the use and impact of an ai code assistant on developer productivity and experience in the enterprise. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–13.
[110] Lance T Wilhelm, Xiaohan Ding, Kirk McInnis Knutsen, Buse Carik, and Eugenia H Rho. 2025. How Managers Perceive AI-Assisted Conversational Training for Workplace Communication. *arXiv preprint arXiv:2505.14452* (2025).
[111] Laurie Williams and Richard L Upchurch. 2001. In support of student pair-programming. *ACM Sigcse Bulletin* 33, 1 (2001), 327–331.
[112] Xiaoying Wu, Richard C Anderson, Kim Nguyen-Jahiel, and Brian Miller. 2013. Enhancing motivation and engagement through collaborative discussion. *Journal of Educational Psychology* 105, 3 (2013), 622.
[113] Zarifa Zakaria, Jessica Vandenberg, Jennifer Tsan, Danielle Cadieux Boulden, Collin F Lynch, Kristy Elizabeth Boyer, and Eric N Wiebe. 2022. Two-computer pair programming: Exploring a feedback intervention to improve collaborative talk in elementary students. *Computer Science Education* 32, 1 (2022), 3–29.
[114] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. Visar: A human-ai argumentative writing assistant with visual programming and rapid draft prototyping. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*. 1–30.
[115] Zheng Zhang, Weirui Peng, Xinyue Chen, Luke Cao, and Toby Jia-Jun Li. 2025. LADICA: a large shared display interface for generative AI cognitive assistance in co-located team collaboration. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–22.
[116] Gang Zhao, Lijun Yang, Biling Hu, and Jing Wang. 2025. A Generative Artificial Intelligence (AI)-Based Human-Computer Collaborative Programming Learning Method to Improve Computational Thinking, Learning Attitudes, and Learning Achievement. *Journal of Educational Computing Research* (2025), 07356331251336154.
[117] Sebastian Zhao, Alan Zhu, Hussein Mozannar, David Sontag, Ameet Talwalkar, and Valerie Chen. 2025. Codinggenie: A proactive llm-powered programming assistant. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 1168–1172.
[118] Baichang Zhong, Qiyun Wang, and Jie Chen. 2016. The impact of social factors on pair programming in a primary school. *Computers in Human Behavior* 64 (2016), 423–431.
[119] Sijin Zhu, Zheng Wang, Yuan Zhuang, Yuyang Jiang, Mengyao Guo, Xiaolin Zhang, and Ze Gao. 2024. Exploring the impact of ChatGPT on art creation and collaboration: Benefits, challenges and ethical implications. *Telematics and Informatics Reports* 14 (2024), 100138.
[120] Barry J Zimmerman and Dale H Schunk. 2011. Self-regulated learning and performance: An introduction and an overview. *Handbook of self-regulation of learning and performance* (2011), 15–26.

A Appendix A: Large Language Model Prompt

Prompt for Proactive Intervention and Direct Request

You are Bob, an AI pair programming assistant focused on LEARNING.

Problem: {problem_title or 'General coding'} - {problem_description}

Language: {programming_language}

Current code: {code_context or "No code visible yet"}

{ai_history_context}

INTERVENTION APPROACH:

- Help users when they need it, but avoid unnecessary responses when they're satisfied
- When users say 'I'm not sure', 'I need help', or ask questions, provide helpful guidance
- When they say 'okay', 'thanks', 'got it', return "NO RESPONSE"
- CRITICAL: Look at your recent messages above - you CANNOT repeat the same type of response
- Each response must be MORE CONCRETE than your previous ones if they still need help
- NEVER end responses with questions like "Need help with...?" or "Want me to...?"
- Focus on brief conceptual hints rather than code snippets

Return EXACTLY "NO RESPONSE" (if no response needed) OR provide a helpful response (10-30 words).

{current_conversation}

Prompt for Task Scaffolding

You are a coding tutor that creates minimal scaffolding to help students learn by doing.

A user wrote this comment in a {language} file:

"{comment_line.strip()}"

Full code context:

{full_code}

CRITICAL RULES:

(1) Only provide scaffolding if the comment indicates the user wants to implement something
(2) Generate MINIMAL scaffolding - just structure, NO solutions
(3) Use descriptive TODO comments instead of __ placeholders
(4) Keep it SHORT (max 5-8 lines)
(5) Students must fill in ALL the actual implementation

SCAFFOLDING REQUIREMENTS:

- TODO comments
- Clear, descriptive TODO guidance
- NO actual implementation or solutions

OUTPUT FORMAT:

- If scaffolding needed: Return ONLY the minimal scaffolding code (NO markdown formatting, NO code blocks, just raw code)
- If no scaffolding needed: Return exactly "NO SCAFFOLDING"
- Do NOT use "'language"' formatting in your response
- Return plain text code only

GOOD scaffolding examples:

Comment: "# Create a function to calculate average"

Output:

```
# TODO: calculate sum of all numbers
# TODO: divide sum by count of numbers
# TODO: return the average
```

Comment: "// Implement bubble sort"

Output:

```
# TODO: loop through array multiple times
# TODO: compare adjacent elements
# TODO: swap if in wrong order
# TODO: return sorted array
```

BAD examples (too much solution):

- Any actual calculations or logic
- Complete implementations
- Specific values or algorithms

Prompt for Code Run Feedback

Code execution analysis:

Code: {code}

Problem: {problem_context or 'General coding'}

Success: {result.get('success', True)}

Output: {output if output else 'None'}
Error: {error if error else 'None'}
CRITICAL RULES:

- Single loops through function results = EFFICIENT O(n)
- ONLY suggest hashmap for actual nested for loops (for i, for j pattern)
- Max-finding with single loop = CORRECT and EFFICIENT
- ONLY analyze subtasks that have actual code implementation
- Skip subtasks that are just comments, TODOs, or placeholders without code
- Ignore commented-out subtask implementations
- Ignore subtasks that only contain comments followed by "return None" or similar placeholders

Response (max 150 chars):

- Errors: "Fix: [issue]"
- Wrong output: "Output: [issue]"
- Actual inefficiency: "Optimize: [suggestion]"
- Working efficiently: "correct"

Examples: "Fix: Missing)", "correct",
"Subtask 1: correct, subtask 2: replace nested loops with hashmap" (only mention subtasks with actual code)

Prompt for Code Block Analysis

Analyze this {language} code:
{code}
Problem: {problem_context.get('title', 'Unknown') if problem_context else 'General coding'}
CRITICAL RULES:

- Max-finding algorithms using single loops are EFFICIENT and CORRECT
- ONLY suggest hashmap for actual nested loops (for i in range, for j in range pattern)
- Single loop iterating through function results = O(n) = EFFICIENT
- Don't flag: style, comments, missing subtasks, working algorithms

Only flag: undefined variables, syntax errors, actual logic bugs
JSON: {"issue": {"title": "...", "description": "...", "hint": "..."}}
Good code: {"issue": {"title": "Code looks good!", "description": "Correct and efficient.", "hint": "Well done!"}}

B Appendix B: Programming Tasks during User Study

B.1 Task 1: Gift Card Purchase Assistant

Scenario: A user has a gift card with a fixed value (e.g., $100) and wants to buy two items from their shopping cart whose prices add up exactly to the gift card value.

Subtasks:

(1) Given a chosen item index, find another item whose price equals the remaining gift card balance.
(2) Return all pairs of item indices whose prices add up exactly to the gift card value *(bonus: optimize to $O(n)$)*.
(3) Find the pair that includes the highest possible individual item price among all valid pairs.

B.2 Task 2: Step Tracker Insight

Scenario: A fitness tracking app that tracks user daily step counts in an array.

Subtasks:

(1) Daily Average: Calculate the overall average steps per day across all recorded days.
(2) Best k-Day Streak Average: Find the highest average step count within any k consecutive days *(bonus: optimize to $O(n)$)*.
(3) Shortest Target Subarray: Find the shortest contiguous subarray where the total steps is at least a given target.

B.3 Task 3: Meeting Room Scheduler

Scenario: A meeting room booking system that manages time slots for conference rooms. Each booking is represented as an interval [start_time, end_time].

Subtasks:

(1) Longest Meeting: Find the meeting with the longest duration.
(2) Detect Conflicts: Identify all pairs of meetings that have overlapping time intervals *(bonus: optimize to $O(n \log n)$)*.
(3) Merge Overlapping: Merge all overlapping or adjacent meeting intervals into a single interval.

C Appendix C: Questionnaire

C.1 Collaborative Learning Scale

Note: Group here can either be other human + AI or AI partner.

(1) I felt part of a learning community in my group.
(2) I actively exchanged my ideas with group members.
(3) I was able to develop new skills and knowledge from other members in my group.
(4) I was able to develop problem-solving skills through peer collaboration.
(5) Collaborative learning in my group was effective.
(6) Collaborative learning in my group was time-consuming.
(7) Overall, I am satisfied with my learning experience.

C.2 Social Presence Questionnaire

(1) When I had real-time interaction during this programming task, I had my collaborator(s) in my mind's eye (able to imagine how they were engaging in the task).
(2) During real-time interaction in this programming task, I felt that I was working with real partners, not with abstract or anonymous entities.
(3) Real-time interactions in this programming task could hardly be distinguished from face-to-face collaboration.

C.3 Additional Questionnaire

(1) I felt responsible for understanding the AI's suggestions before applying them.
(2) The AI felt too interruptive/disruptive.

Table 3: Frequency Distribution of Utterance Types Across Conditions and Post-Hoc Chi-Square Test Results

Utterance Type	Condition (n / %)			Statistical Test		
	HHAI-S	HHAI-P	HAI	$\chi^2(df = 2)$	V	p_{adj}
Question (Seek Info)	**174 (12.06)**	**190 (14.22)**	**126 (35.10)**	**119.24**	**.19**	$< .001^{***}$
Question (Seek Confirm)	80 (5.54)	76 (5.69)	20 (5.57)	0.03	.00	1.000
Answer / Reply	27 (1.87)	31 (2.32)	1 (0.28)	6.40	.05	.205
Strategy Proposal	60 (4.16)	71 (5.31)	14 (3.90)	2.58	.03	1.000
Implementation Proposal	253 (17.53)	201 (15.04)	63 (17.55)	3.46	.03	1.000
Elaboration / Justification	**77 (5.34)**	**35 (2.62)**	**0 (0.00)**	**29.88**	**.10**	$< .001^{***}$
Think-Aloud (Cognitive)	287 (19.89)	311 (23.28)	65 (18.11)	7.00	.05	.251
Read-Aloud / Verbalization	22 (1.53)	12 (0.90)	4 (1.11)	2.31	.03	1.000
Acknowledgment/Acceptance	**237 (16.42)**	**193 (14.45)**	**24 (6.69)**	**22.03**	**.08**	$< .001^{***}$
Disagreement / Rejection	41 (2.84)	37 (2.77)	4 (1.11)	3.59	.03	1.000
Coordination / Turn-Taking	76 (5.27)	59 (4.42)	11 (3.06)	3.44	.03	1.000
Affective Expression	109 (7.55)	120 (8.98)	27 (7.52)	2.11	.03	1.000

Note. Values in parentheses indicate column percentages. HHAI-P = HHAI-Personal; HHAI-S = HHAI-Shared. V represents Cramer's V effect size. p_{adj} represents p-values adjusted for multiple comparisons using the Holm-Bonferroni method. Bold rows indicate statistical significance ($^{***}p < .001$).

Table 4: Accuracy of GPT-4o-mini annotations across conditions.

Condition	User Studies	User Utterances	Accuracy (%)
HHAI-Shared	3	515	88.54
HHAI-Personal	3	461	90.89
HAI	3	151	82.78

D Appendix D: Supplementary Analysis on User Utterances

To understand how participants engaged throughout the session and to further analyze their interactions, we annotated a randomized subset of user utterances (3,138 utterances across all studies). Our annotation process is as follows:

(1) To construct the annotation scheme, one author conducted initial open coding on $\sim 10\%$ of the utterances to identify recurring patterns [32]. The initial codes were then reviewed and discussed with two other authors, and through this process, we finalized the annotation scheme presented in the table 2.
(2) We then annotated three sampled user studies (3 studies across 3 conditions = 1127 user utterances = $\sim 36\%$ of the data). Two annotators independently coded this subset (Cohen's $\kappa = 0.853$) and resolved all disagreements to produce final labels.
(3) To annotate the remaining portion of the dataset ($\sim 64\%$), we used the gpt-4o-mini model to classify each utterance using our annotation scheme, building on prior work in AI-assisted annotation [17, 28, 77]. To provide additional context [91], the model was given the three preceding and three following utterances as the context window for each target instance. We evaluated the model by comparing its predictions on our human-annotated subset and reported its resulting accuracy (table. 4).
(4) Finally, we calculated and reported the proportion of each utterance type across all conditions, as presented in the table 2.

E Enlarged Views of Figure 2 Components

The enlarged views of components from figure 2 is provided in the next page as follows.

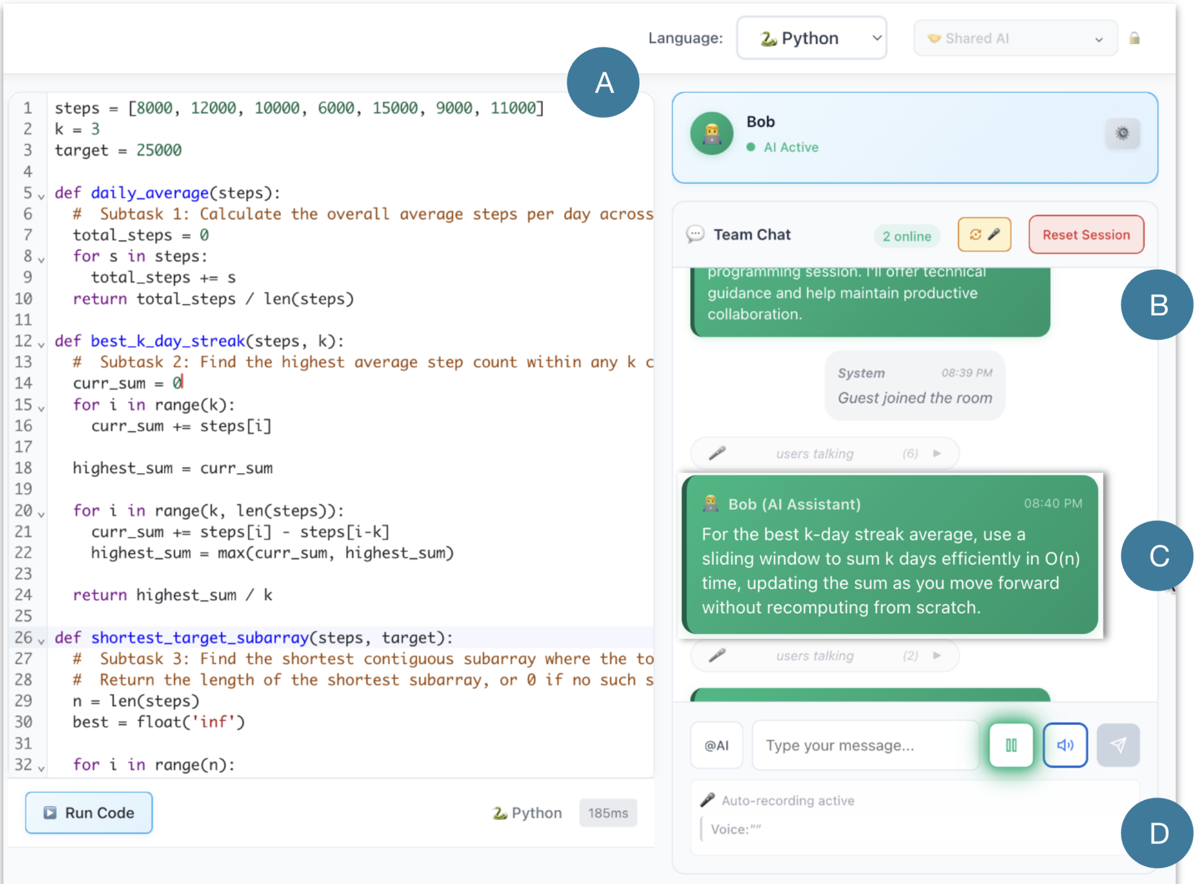

Figure 7: Enlarged Views of System Components. A) Collaborative Code Editor. B) Conversational Interface. C) Proactive Intervention. D) Live Transcription.

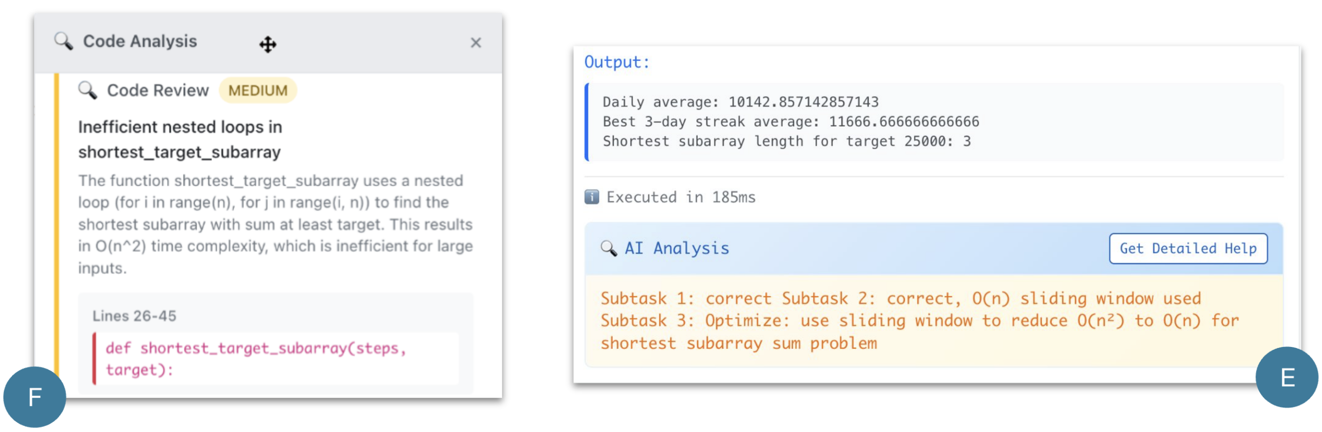

Figure 8: Enlarged Views of System Components. E) Code Run Feedback. F) Code Block Analysis.