Issue 18, Volume 1, November 2025

Teach2Learn: Developing and Piloting an Educational Platform for Learning-by-Teaching in CS1 Courses

Aizen Baidya

ACKNOWLEDGEMENTS

This was written in Engineering 195: Undergraduate Research with Ayush Pandey.

Teach2Learn: Developing and Piloting an Educational Platform for Learning-by-Teaching in CS1 Courses

Aizen Baidya and Ayush Pandey

University of California, Merced

ENGR 195: Upper Division Undergraduate Research

October 24, 2025

Abstract

The surge of generative AI in education calls for new learning approaches resistant to cognitive offloading. Our work explores a solution inspired by the Latin proverb, *docendo discimus*: "by teaching, we learn." We developed and piloted Teach2Learn, a web-based educational platform where CS1 students demonstrate their knowledge by teaching an LLM-simulated learner. This paper presents the pedagogical framework, design, and findings of two pilot tests (N = 87). We conducted pilots with two distinct student cohorts: an interdisciplinary summer cohort of non-engineering majors (N = 32) with limited prior programming experience and a more homogeneous fall cohort of engineering majors (N = 55). Our preliminary results indicate a positive impact on student self-efficacy. In both pilots, the majority of the students reported an improved understanding of concepts after the activity and felt that the simulated student challenged their thinking. However, the impact on conceptual understanding, measured by exam-style questions, was mixed. The non-engineering cohort showed improved scores, while the engineering cohort's performance decreased on several questions. The mixed results underscore the need for further iteration.

Keywords: Learning-by-Teaching, Large Language Models, AI for Education, Computer Science Education, Students as Learners and Teachers, Flipped Learning, Human-Centered AI

1 Introduction

The landscape of computer science (CS) education is changing due to the emergence of large language model (LLM)-based generative AI tools. Although these tools are promising, they present an educational challenge: a reduced need for "productive struggle." Now that CS students have access to their own skilled AI pair programmers, answers are often only a query away. This new dynamic contests the very ways by which we learn. As such, educators must devise ways to teach and learn in harmony with AI.

Although LLMs are often used as personal tutors, we present an approach that follows a different paradigm: learning-by-teaching (LbT). As illustrated in Figure 1, the traditional model of learning often places students as passive containers of knowledge. This is a direct instance of the "banking model" of education, as critiqued by the educator and philosopher Paulo Freire. Freire argued that this process, where teachers act as depositors of information, is non-participatory and disempowering. The LbT model, in contrast, aligns with Freire's call for a more dialogical pedagogy by turning learners into teachers.

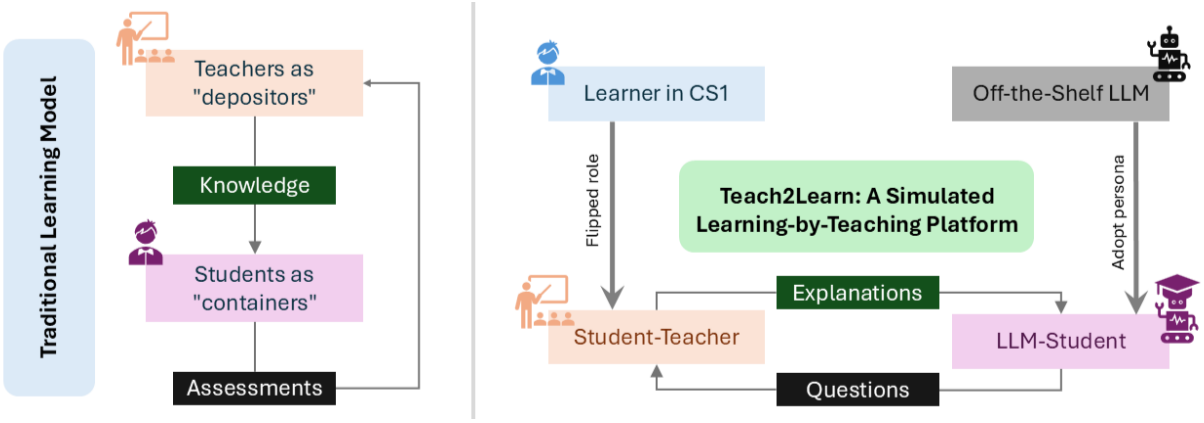


Figure 1: Traditional Learning Model vs. Learning-by-Teaching with Teach2Learn

To explore this, we developed Teach2Learn, operationalizing LbT by asking CS1 students to teach an LLM-simulated peer. The principle is straightforward: explaining a concept to others requires a depth of understanding that cannot easily be faked or outsourced. This approach aims to reintroduce productive struggle in a way that is resistant to cognitive offloading.

2 Related Work

Our work is situated within the growing field of using LLMs to create simulated personas for educational teaching and learning environments. These tools often follow the pedagogical principle of LbT, where the act of explaining a concept solidifies one's own understanding.

One line of research focuses on teacher training. An example is GPTeach, an interactive teaching preparation tool for novice teaching assistants that enables them to practice with artificially generated student personas (Markel et al. 2023). The researchers found that this low-pressure environment encouraged the TAs to refine their responses and allowed them to tailor their teaching to various simulated student needs. Although GPTeach demonstrates the usefulness of simulated students for TA training, our work adapts this model to CS1 student learning and assessment.

Other platforms also focus more directly on LbT for students. TeachYou (Jin et al. 2024) and MatlabTutee (Rogers et al. 2025) leverage LLM-based teachable agents that "play dumb" to help students identify knowledge gaps. Both TeachYou and MatlabTutee advocate the notion that these agents can positively assist in the learning process.

Although these platforms provide a strong foundation, a gap remains in their implementation context. Much of the existing research evaluated these LbT tools with volunteers or recruited participants outside of a formal class structure. Teach2Learn builds on this work by specifically designing, integrating, and evaluating an LbT platform as an in-class activity.

3 The Teach2Learn Platform

The design of the Teach2Learn platform is based on four key principles. Our primary goal was to (1) facilitate knowledge articulation, creating an environment in which students must actively explain their understanding, rather than passively consume information. To achieve this, we prioritized providing (2) a safe and judgment-free space where students can explore concepts and learn from mistakes without the social pressure of tutoring a human peer; a state of psychological safety crucial to encouraging metacognitive reflection (Lateef 2020).

Furthermore, the platform was designed to (3) scaffold the discussion with a simulated peer. The AI agent is not merely a passive recipient of knowledge, but an active conversational partner seeking clarity: ensuring that interactions are productive and focused on course learning objectives. Finally, these goals are supported by the foundation of (4) ease of use of the interface, allowing students to focus on the teaching task.

Figure 2: Main Teaching Interface of Teach2Learn

3.1 User Interaction Flow

The student's journey through Teach2Learn is intentionally structured to guide them from passive knowledge review to active teaching. After logging in, the student is first directed to a "Review" page that has key concepts and code examples. This initial step serves to prime prior knowledge and, by providing a reference, lowers the intimidation barrier for the main teaching activity.

When ready, the student navigates to the core teaching interface: a chat-based GUI where

they are paired with the LLM-simulated learner. This interface, shown in Figure 2, includes an interesting, non-traditional constraint: students can only see the most recent message and cannot scroll back through the conversation history. This intentional design choice was made to increase the commitment to the interaction, simulating a real-life dialog where one must actively listen and ask for clarification, and not be able to reread a transcript. This constraint forces the student-teacher to keep the conversation history in their working memory, compelling them to pay more attention to the dialogue. The interaction is framed by the simulated student, issuing a clear starting message asking for help.

Crucially, the session is self-directed and open-ended. The platform does not enforce a formal completion state. Instead, the teaching session ends when the student-teacher perceives that they have successfully taught the concept, guided the simulated student through its misconceptions, and answered all its questions. This workflow gives students full control over the pace of their lesson. Teach2Learn transforms a simple conversation into a structured exercise in metacognition and knowledge articulation.

3.2 System Architecture

Teach2Learn is built on a modular web architecture that separates the frontend interface from the backend logic. The client-side interface was developed using the React library. React was chosen for its ability to build interactive and stateful user experiences, which is essential for a real-time application.

The client communicates with a Python-based backend API built with the Flask microframework. We chose Flask for its lightweight nature and simplicity, which allowed rapid development to handle three primary tasks: (1) managing user authentication, (2) interfacing with the external LLM, and (3) communicating with our database.

To accelerate development and ensure robust infrastructure, we leveraged several platform-as-a-service (PaaS) and backend-as-a-service (BaaS) solutions. The entire application is deployed on Railway, a modern PaaS that handles infrastructure, allowing us to provide a stable platform

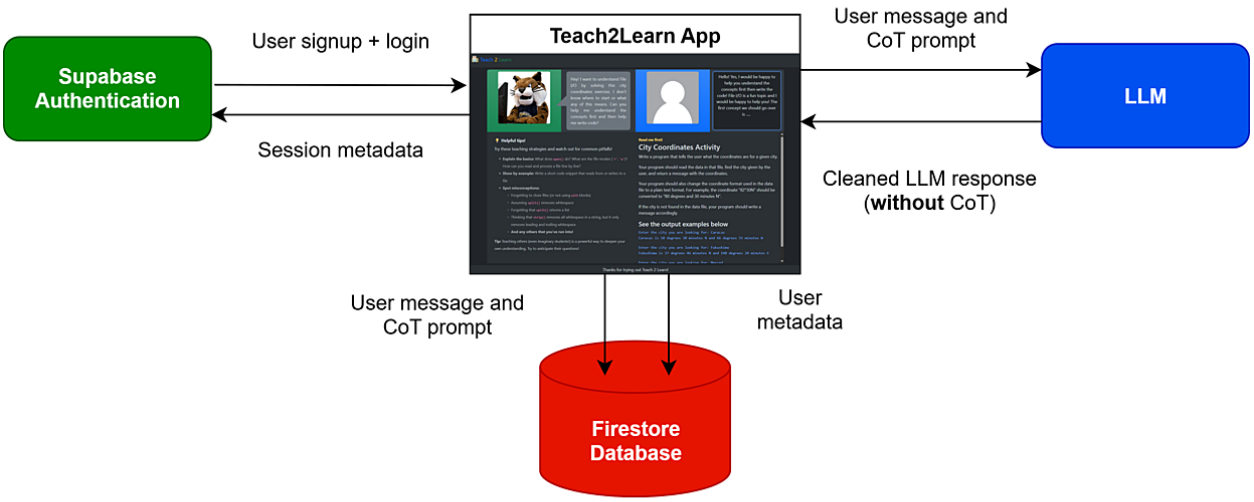


Figure 3: High-Level Diagram of Teach2Learn System Architecture

for students via a simple URL. User authentication (including signup and login) is managed securely by Supabase. Using this BaaS saved development time and provided a secure authentication system.

All chat logs are stored in Firestore, a NoSQL document-oriented database ideal for storing chat logs, as each teaching session can be saved as a single "document" containing all messages. This structure is highly scalable and simplifies the process of collecting chat data.

3.3 Teachable AI Agent Design

The central challenge in designing the AI agent was to constrain an all-knowing LLM to mimic a novice CS1 student. A simple persona prompt is insufficient, as it often results in an agent that inadvertently guides the student-teacher toward the right answer.

Even when instructed to "act as a student," an LLM's underlying keenness for helpfulness can emerge, causing it to ask leading questions. This undermines the entire LbT model, as the student-teacher is no longer truly articulating their own knowledge, and rather only following conversational cues. Our goal was to create an agent that was productively and realistically confused, thereby compelling the student-teacher to take full ownership of the teaching process.

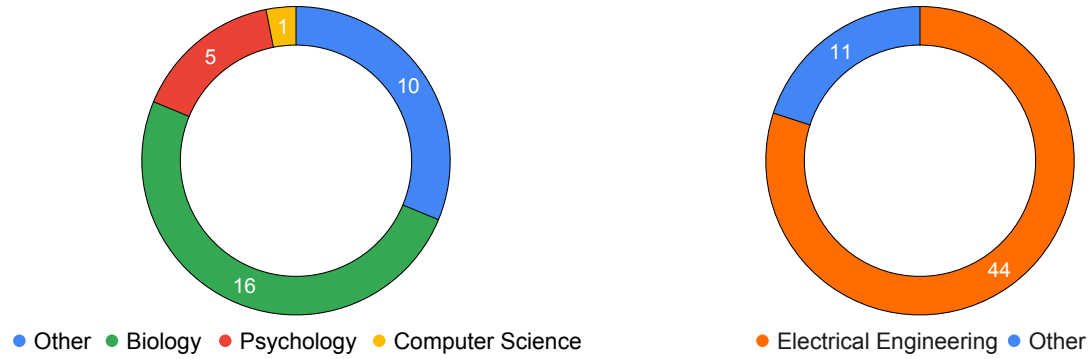
To achieve this, our technical implementation relies on a structured, multi-step prompt that

builds directly on the principles of chain-of-thought (CoT) prompting. The foundational work in this area (Wei et al. 2023) demonstrated that LLMs' reasoning abilities emerge when prompted with few-shot exemplars that include intermediate reasoning steps. Kojima et al. (2023) showed that this reasoning could also be unlocked in a zero-shot manner by simply instructing the model to "think step by step."

Our approach is a hybrid of these ideas. Rather than providing a few examples or a simple instruction, our 11-step prompt functions as an algorithm, guiding the simulated student to follow a specific CoT for every turn. This allows us to guide the LLM's mistakes, adapting the general CoT concept to create a pedagogically valuable teachable agent that prompts the student-teacher to reflect and re-articulate, rather than simply gliding through the conversation.

4 Methodology

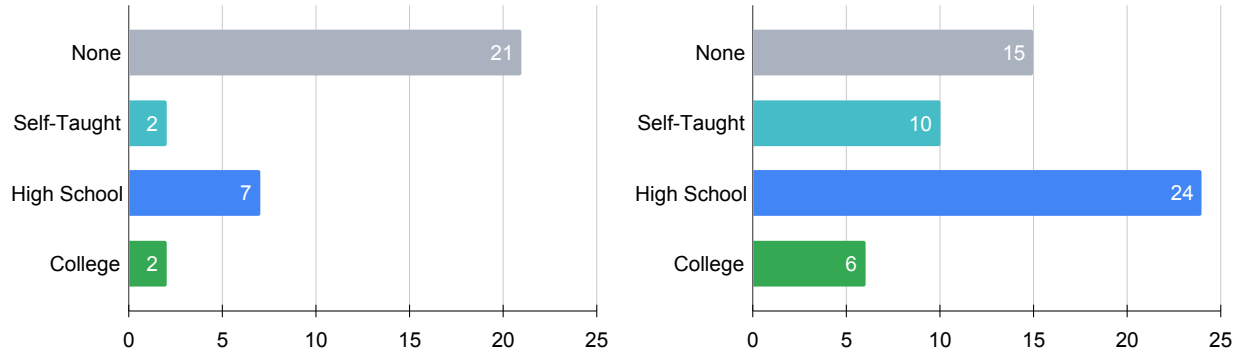
To evaluate the Teach2Learn platform, we conducted two in-class pilot tests (total N = 87) after obtaining IRB approval for this study. First, we introduced Teach2Learn over the summer in a CS1 lab setting, followed by a second, larger pilot conducted during lecture in the fall for an EE-focused CS1 class. The summer pilot presented a unique cohort for our study, as the course was almost entirely composed of non-CS majors (see Figure 4a), with nearly no previous programming experience (see Figure 5a).



(a) Summer Majors

(b) Fall Majors

Figure 4: Majors of Summer and Fall Pilot Students



(a) Summer Prior Experience

(b) Fall Prior Experience

Figure 5: Prior Programming Experience of Summer and Fall Pilot Students

4.1 Participants

Tools that are only effective for students already inclined towards CS have limited utility. Our summer pilot (N = 32), with its cohort of majors predominantly from biology and psychology, illustrates a population of true novices, many of whom are learning CS out of necessity for their fields. This composition is a strength of our first pilot, highlighting the potential generalizability of Teach2Learn beyond traditional CS audiences.

In contrast, the fall cohort (N = 55) was much more homogeneous, consisting almost entirely of students majoring in EE (see Figure 4b). Taken together, these two populations provided us with an important scope to determine the effectiveness of Teach2Learn.

To further characterize our two cohorts, we collected data on previous programming experience. The experience levels for both groups are summarized in Figure 5. As shown in Figure 5a, the summer pilot cohort was mainly composed of novices, with a high percentage of students (21 out of 32; 65.6%) with no prior programming experience. In contrast, the fall EE cohort was more experienced (see Figure 5b). Nevertheless, more than a quarter of the class (15 out of 55; 27.3%) reported having no prior programming experience, highlighting a mixed-experience cohort.

4.2 Procedure

The procedure for both pilots was identical. The specific topic for learning was file reading and processing with Python. The in-class flow for each student was as follows:

1. **Lecture:** The students first received a lecture on the topic.
2. **Pre-Survey:** Following the lecture, students were directed to complete a brief pre-survey to establish a baseline for their conceptual understanding and confidence.
3. **Teach2Learn Activity:** Students then engaged with the Teach2Learn platform as the main LbT activity.
4. **Post-Survey:** After completing the teaching session, students were asked to complete a post-activity survey.

4.3 In-Class Activity Design

From the student's perspective, the core activity began after they submitted the initial pre-survey and navigated to the Teach2Learn web platform. Here, they were presented with the main interface and a clear instructional prompt. This prompt tasked them with teaching a virtual novice student the topic covered in the preceding lecture: file reading and processing. Participation in all components (both surveys and the Teach2Learn activity) was offered for additional credit.

The primary form of scaffolding was the simulated student. As the student-teacher provided explanations, the agent would ask clarifying questions or express confusion. This design was for students to reflect on their own knowledge and practice articulating concepts more clearly. The activity was open-ended; it was "complete" not by a system-defined metric, but rather when the students felt they had successfully taught the concept. After they finished, they proceeded to the post-survey.

4.4 Data Collection

The pre- and post-activity surveys were designed to be parallel instruments to measure changes in conceptual knowledge and self-efficacy. From these sources, we measured three key areas:

- **Conceptual Understanding:** Measured via performance on exam-style questions.
- **Self-Efficacy:** Measured using 5-point Likert scale questions to assess student confidence.
- **Qualitative Feedback:** Gathered from open-ended questions in the post-survey.

5 Results

Our methodology, involving two distinct pilots, allowed us to evaluate Teach2Learn on several key dimensions: self-reported changes in understanding and confidence, objective learning gains, and qualitative user feedback. To provide context, Figure 6 illustrates a typical interaction with the activity, showing the progression from the initial question to the explanations and clarification questions.

5.1 Self-Efficacy and Perceived Understanding

We assessed students' self-reported confidence and understanding using 5-point Likert-scale questions in the post-activity survey.

Perception of Comprehension: In both pilots, most of the students reported improved understanding after the activity (Figure 7a). In the summer pilot (N = 25), 76% (19) indicated that they grasped the concepts "a bit better" or could "explain it clearly now." Similarly, in the fall pilot (N = 42), 64.3% (27) reported a positive gain in understanding.

Challenge to Thinking: We also asked if the activity challenged students' thinking as a proxy for metacognitive engagement. In the summer pilot, 88% (22 of 25) agreed that the simulated student challenged their thinking. The effect was less pronounced but still strong in the more

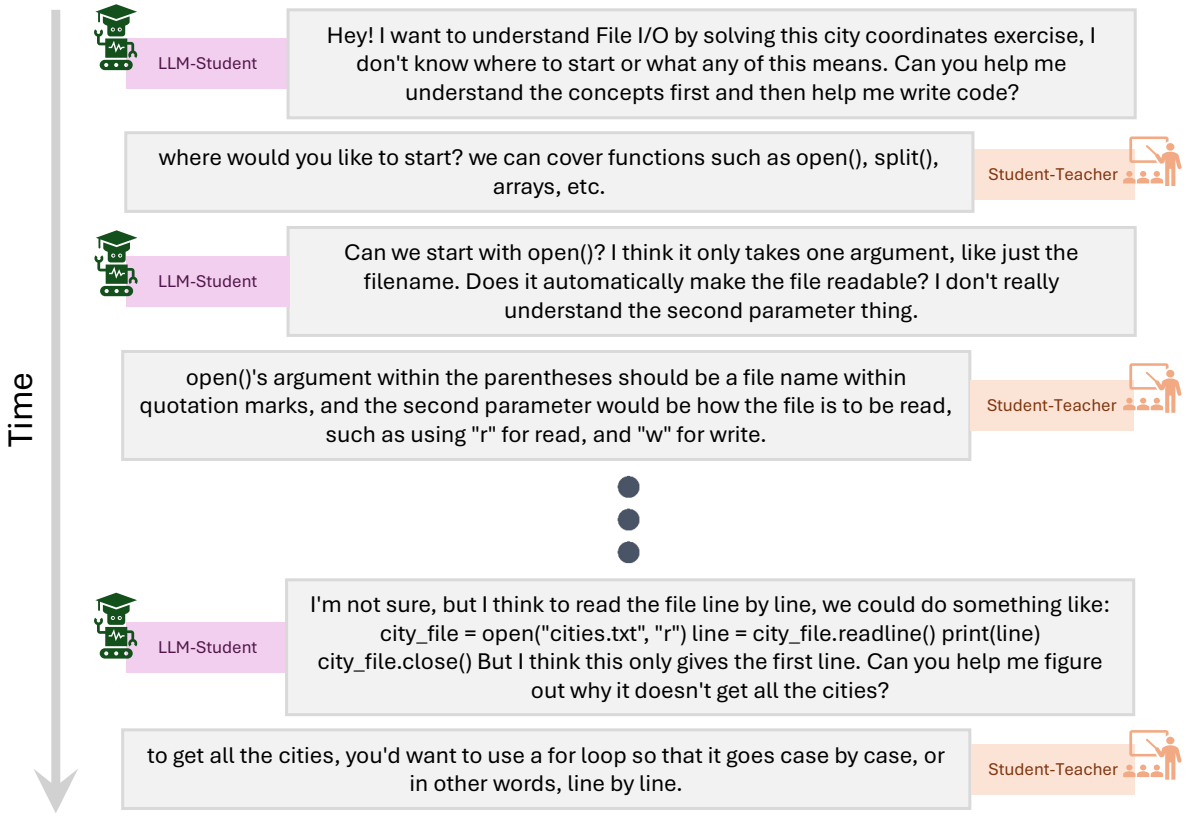


Figure 6: Dialogue of Real Interaction between a Student-Teacher and the LLM-Student

experienced fall cohort, where 66.7% (28 of 42) agreed. This suggests that the LbT activity was successful in eliciting reflection, especially for the summer cohort students.

Realism of the LLM Student: Student perception of the simulated student's realism was generally neutral (see Figure 7b). The most common response for both cohorts was "somewhat realistic." Although not perceived as highly human-like, this suggests that the agent was sufficiently believable to facilitate the teaching interaction without completely breaking immersion.

5.2 Quantifying Learning Outcomes

We measured learning outcomes by comparing performance on identical exam-style questions in the pre- and post-surveys.

Summer Pilot: The summer cohort demonstrated a strong initial understanding of the first two

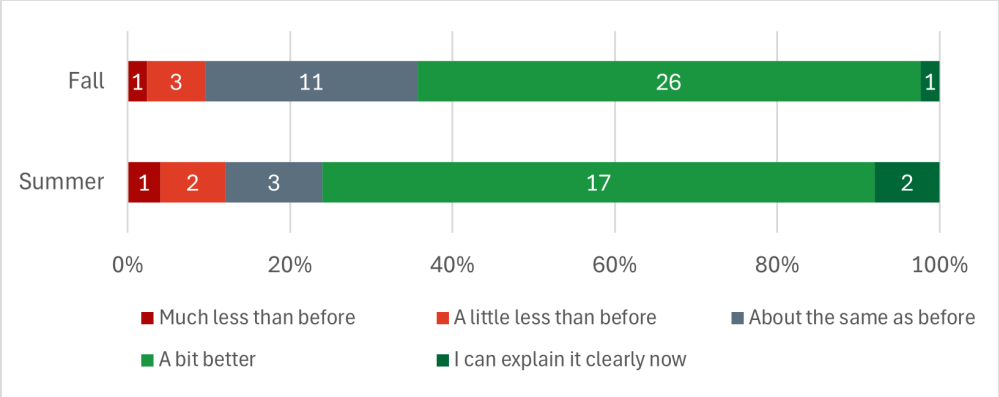
questions (87.5% and 78.1% correct) in the pre-survey (N = 32). The latter two more challenging questions showed lower baseline correctness (with 37.5% and 40.6% correct). Performance improved after the post-survey (N = 25) in all questions: correctness increased to 92% for the first two and increased to 52% for the last two questions (Figures 8a, 8b).

The mean score on the pre-activity survey was 57.03 (SD = 27.12), while the mean score on the post-activity survey increased to 68.00 (SD = 18.43). To assess whether the increase was statistically significant, we performed a t-test. The results indicated that the observed difference between the pre- and post-activity scores was not statistically significant at the conventional level $\alpha = 0.05$ (t = -1.814, p = 0.0752). Although the average score did increase numerically, this statistical test suggests that we cannot confidently conclude that the activity caused a significant change in conceptual understanding for this cohort.

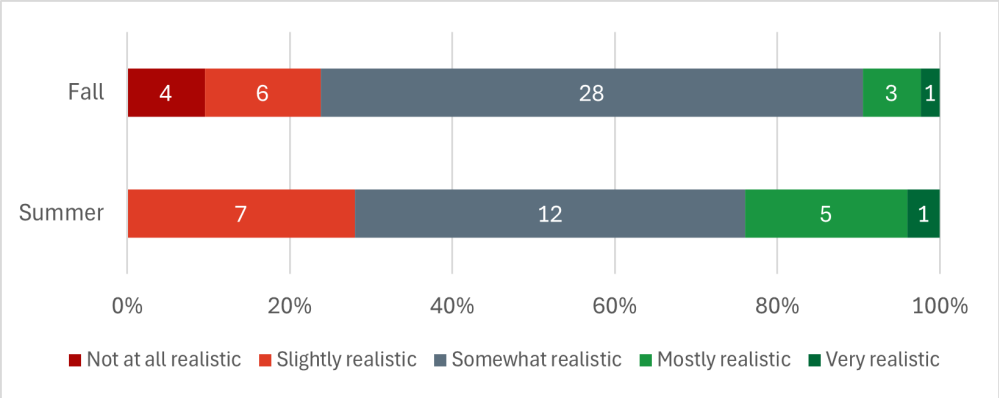
Fall Pilot: The fall cohort showed a mixed level of initial understanding (78.2%, 41.8%, 45.5%, 29.1% correct in the four questions). The post-survey results revealed a surprising trend (see Figures 8c, 8d). Although the correctness in the first question increased 7.5%, the performance decreased in the other three questions (dropping to 35.7%, 40.5%, and 26.2%).

The mean score on the pre-activity survey (N = 55) was 44.09 (SD = 29.64). In contrast to the summer results, the mean score in the post-activity survey (N = 42) decreased to 43.45 (SD = 30.27). A t-test confirmed that this small difference was not statistically significant (t = 0.104, p = 0.9175). These results indicate that the activity did not produce a statistically significant change in conceptual understanding for this cohort.

The student outcomes in the post-test decreased compared to the pre-test. We hypothesize that this is due to a lack of engagement towards the end of the activity and the fact that we did not award any grades, resulting in many students choosing random options to quickly complete the activity to obtain extra credit, as completion was the only requirement.



(a) "After this activity, I understand file handling..."



(b) "How realistic did the simulated student feel?"

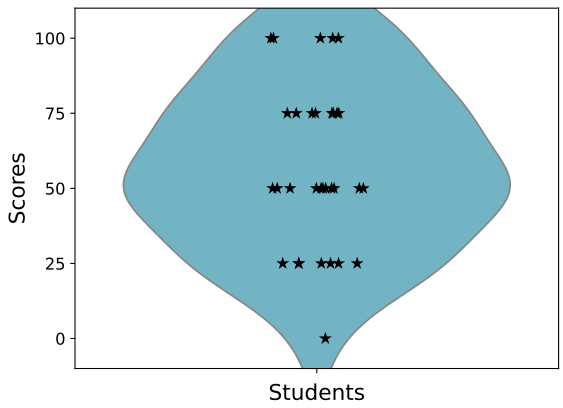
Figure 7: Student Responses to Survey Questions on Perceived Comprehension and Realism

5.3 Qualitative Student Feedback

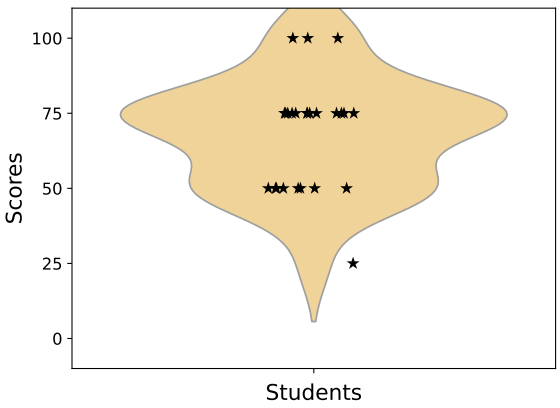
In addition to quantitative measures, we collected open-ended feedback through the post-activity survey to understand the student experience and identify areas for improvement.

Initial responses highlighted the perceived benefits of the LbT approach. Some students noted that the activity helped their learning: "[The activity] helped solidify what I know." Others recognized the value of articulating concepts: "Explaining concepts in different ways was challenging," suggesting that the activity prompted deeper reflection as intended by LbT.

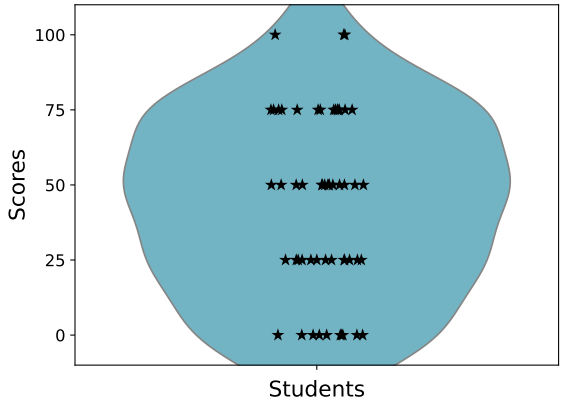
Another category of feedback was the agent's behavior: students mostly perceived the agent as "somewhat realistic," but some encountered issues. Students reported that "the AI sometimes did not understand what I was saying," its "questions felt unclear," or it got stuck in "repetitive steps



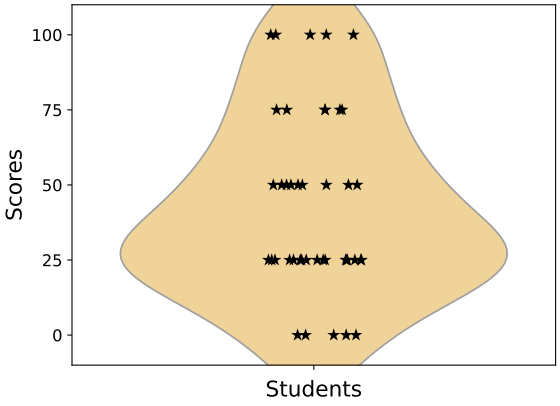
(a) Summer Students Pre-Test Scores



(b) Summer Students Post-Test Scores



(c) Fall Students Pre-Test Scores



(d) Fall Students Post-Test Scores

Figure 8: Student Scores on Pre- and Post-Tests

to which [it] kept looping back."

Finally, some students commented on the UI/UX aspects, suggesting improvements to the interface and pointing out confusion over the initial layout: "I didn't know I had to type in the box to help the student at first." The inability to write longer responses ("could not write as much as I wanted in the chat-box") and the lack of visibility for the character limit were also mentioned.

Student feedback provides crucial information on areas for improvement in clearer task definition, better AI conversational flow, and refined interface design.

6 Limitations and Future Work

Although our pilots provided valuable feedback, several limitations should be considered, pointing to important directions for future work.

6.1 Limitations

The behavior of the agent was governed solely by prompt engineering applied to a general-purpose LLM. Such models possess vast knowledge and a tendency towards helpfulness. Furthermore, the activity's design consisted of a brief intervention focused on one topic, with measurements taken immediately before and after the activity. This means that our findings reflect only short-term effects on confidence and conceptual understanding and cannot predict the impact of regular use. Student participation was also incentivized as an extra credit opportunity. This limits our findings on student engagement, as motivation may have differed if participation were mandatory.

6.2 Future Work

Many future directions result from our work. The AI model that we used (GPT 4.1) was an off-the-shelf model with CoT prompting. These models are pre-trained to play the role of a helpful assistant, which defeats our purpose of using them as a help-seeking student. Thus, fine-tuning is the immediate next step: we will annotate the data from the pilots and design a reinforcement learning from human feedback (RLHF) pipeline for our use case.

Additionally, we note that students did not engage as effectively with Teach2Learn as we would have hoped for due to two reasons: (1) the activity was not part of any regular assessments, and (2) the activity did not provide intrinsic rewards to the students. To address (1), we will integrate Teach2Learn with graded in-class activities and weekly assignments. For (2), we hypothesize that gamification of the platform may help. Gamification of learning tools is a common practice to better engage students with the material. Gamification elements, such as an engagement meter and various gamified checkboxes and badges, will likely also help students stay on track in teaching.

To integrate our platform with learning outcomes, we will introduce two rubric-based assessments: one for teaching effectiveness and one for learning outcomes. Rubrics will help us rate the tutor/tutee responses and provide a path forward to integrating this platform into regular classroom workflows. We hope that this will lead to longer-term educational gains, which were not possible to capture in our pilots.

Finally, another important next step is to incorporate other introductory Python topics into the platform. In the pilot study, we developed only a file reading and writing module. But to show that the platform can be generalized across modules, we will also develop Teach2Learn activities for other topics. LbT is not limited to only introductory Python topics either, so another interesting future direction is to build Teach2Learn activities for mathematics, science, and humanities education.

7 Conclusion

The rise of generative AI presents both opportunities and challenges for CS education, particularly about the potential for cognitive offloading and the erosion of "productive struggle." In response, we developed Teach2Learn, a web-based platform grounded in the LbT philosophy. By situating CS1 students as teachers of an LLM-simulated peer, Teach2Learn aims to foster deeper understanding and provide an assessment approach that discourages AI outsourcing.

Our findings from two in-class pilots are promising. We observed a consistent, positive impact on student self-efficacy, with a majority reporting increased confidence and the feeling that the activity challenged their thinking. However, the impact on conceptual understanding, measured by exam-style questions in the pre- and post-surveys, gave mixed results: neither cohort exhibited statistically significant gains.

These preliminary results suggest that Teach2Learn offers a potential framework to integrate LbT into CS1 courses. The positive effects on self-confidence are encouraging, but the mixed results on conceptual gains highlight that effectiveness may be nuanced and depend on factors such

as implementation details. This underscores the clear need for further improvement. Ultimately, Teach2Learn presents a step towards designing educational technologies that work in harmony with AI, leveraging its capabilities to promote genuine student learning.

Works Cited

Jin, Hyoungwook, et al. “Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education”. *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI ’24, Association for Computing Machinery, 2024, https://doi.org/10.1145/3613904.3642349.

Kojima, Takeshi, et al. Large Language Models are Zero-Shot Reasoners. 2023. *arXiv*, arxiv.org/abs/2205.11916.

Lateef, F. “Maximizing Learning and Creativity: Understanding Psychological Safety in Simulation-Based Learning”. *Journal of Emergencies, Trauma, and Shock*, vol. 13, no. 1, 2020, Epub 2020 Mar 19, pp. 5–14. https://doi.org/10.4103/JETS.JETS_96_19.

Markel, Julia M., et al. “GPTeach: Interactive TA Training with GPT-based Students”. *Proceedings of the Tenth ACM Conference on Learning @ Scale*. L@S ’23, Association for Computing Machinery, 2023, pp. 226–36, https://doi.org/10.1145/3573051.3593393.

Rogers, Kantwon, et al. “Playing Dumb to Get Smart: Creating and Evaluating an LLM-based Teachable Agent within University Computer Science Classes”. *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. CHI ’25, Association for Computing Machinery, 2025, https://doi.org/10.1145/3706598.3713644.

Wei, Jason, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2023. *arXiv*, arxiv.org/abs/2201.11903.