# DevCoach: Supporting Students Learning the Software Development Life Cycle with a Generative AI Powered Multi-Agent System

Tianjia Wang
wangt7@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Matthew Trimble
mtrimble23@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Chris Brown
dcbrown@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

## Abstract

The software development life cycle (SDLC) is vital for ensuring the quality of software systems. However, learning SDLC concepts presents unique challenges, such as the need for effective collaboration, real-time interaction, and access to diverse skill sets represented in software development teams. To address these problems, we present DevCoach, a generative AI powered multi-agent system designed to support students learning the SDLC. DevCoach allows students to interact with generative AI agents simulating the different roles in the software development team, engaging in tasks across different phases of SDLC. Through a user study ($n = 20$), we evaluate the system's effectiveness in enhancing learning, impact on SDLC deliverables, and support for Community of Inquiry (CoI) elements necessary for effective and supportive learning environments. Our results reveal that students using DevCoach achieved significantly higher learning gains and improved task completion rates across all SDLC phases. The system also supports CoI elements, particularly perceived social presence. Participants also lauded the immediate context-aware feedback, interactive learning environment, and diverse expertise provided by the roles within the multi-agent team. These findings demonstrate the potential of generative AI to enhance software engineering education by making it more effective, engaging, and interactive, providing students with collaborative and practical learning experiences.

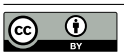
## CCS Concepts

• **Applied computing** → **Education**; • **Software and its engineering**; • **Computing methodologies** → **Artificial intelligence**;

## Keywords

Software Development Life Cycle, Generative AI, Multi-agent System

**ACM Reference Format:**
Tianjia Wang, Matthew Trimble, and Chris Brown. 2025. DevCoach: Supporting Students Learning the Software Development Life Cycle with a Generative AI Powered Multi-Agent System. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696630.3727255




*FSE Companion '25, June 23–28, 2025, Trondheim, Norway*

ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3727255

## 1 Introduction

The software development life cycle (SDLC), a systematic process used to plan, design, develop, test, deploy, and maintain software, is essential for producing high quality software applications [44]. The majority of software failures can be attributed to problems that arise in the SDLC [13, 18], making it a critical part of software engineering (SE) educational curricula [51]. However, the complex concepts and interactive team activities involved in the SDLC pose significant challenges for students to learn and instructors to teach [32]. Consequently, students frequently lack the necessary resources to understand the different development phases in SDLC and practice the crucial skills required for their future careers [5, 42].

Recent advancements in large language models (LLMs) and generative artificial intelligence (AI) have created opportunities for addressing the aforementioned challenges. Existing research demonstrates these models' capabilities in understanding human language, performing coding tasks, and simulating human behaviors [29, 39, 43, 46, 48]. Some studies have developed multi-agent frameworks to simulate human workflows with generative AI agents in different roles within the SDLC, aiming to improve the performance of automatic code generation [28, 46]. However, it remains unclear how effectively generative AI agents can collaborate with humans to complete tasks and activities in the SDLC and assist students in learning the SDLC.

For this project, we developed a multi-agent system, DevCoach, that leverages generative AI agents to simulate roles in a software development team. Our system assists students in learning and completing activities across various phases of the SDLC. The system contains a learning environment that allow students to gather project requirements, draw design diagrams, write codes, and implement test cases with support from generative AI agents via multimodal communication. The generative AI agents are designed to understand and respond to different activities in requirement, design, develop, test and review phases in the SDLC, with personas to simulate various roles in a software development team. Students can seek guidance, practice skills, ask questions, and receive real-time feedback, fostering an interactive and engaging learning process. By providing a hands-on experience, students can grasp theoretical concepts and gain practical insights into how each decision and action influences the overall software development process.

To understand the effectiveness and impact of the generative AI powered multi-agent system in supporting students learning

the concepts related to the SDLC, we seek to answer the following research questions (RQs):

**RQ1.** How effective is DevCoach in supporting students learning SDLC concepts?

**RQ2.** What impact does DevCoach have on the completion of deliverables across SDLC phases?

**RQ3.** What impact does DevCoach have on cognitive, social, and teaching presence in learning environments?

To answer these questions, we conduct a user study with university students using the system to work on a project to build a movie website backend system by completing tasks in different phases of SDLC. To evaluate DevCoach, we collect data through pre- and post-quiz scores, observation, surveys, and qualitative interviews. Compared to a baseline approach without generative AI agents, DevCoach enabled students to achieve significantly higher learning gains and improved task completion rates across all SDLC phases. Participants valued the structured, interactive learning environment and the expertise of the multi-agent team, which delivered timely, context-aware feedback from diverse perspectives. Additionally, DevCoach significantly improved social presence, fostering stronger interpersonal connections and purposeful communication, while also supporting cognitive and teaching presence by fostering deeper inquiry and more engaging instructional design. These findings underscore the potential of generative AI powered multi-agent systems in advancing software engineering education, offering personalized, interactive, context-aware guidance, fostering collaboration, and enhancing student engagement in learning software engineering concepts.

## 2 Background and Related Work

### 2.1 Software Development Lifecycle

The SDLC is a structured process that guides the software development from an initial idea through to the maintenance of the completed application [44]. It typically includes several key phases: plan, design, develop, test, deploy, maintain, and review. These phases vary among different SDLC models. For instance, the Agile SDLC model refines the traditional SDLC by combining and condensing phases to enhance flexibility and responsiveness, focusing on these essential stages to accelerate product delivery and adapt quickly to changes [7, 38]. In our work, we adopted an agile approach, limiting the SDLC to five phases: "Requirement", "Design", "Develop", "Test", and "Review".

Existing studies have investigated the critical role of the SDLC in software development [6, 7]. The SDLC is also considered a vital part of SE education and curricula to help students gain practical skills [51]. Prior work has explored pedagogical techniques to help students learn SDLC concepts—including adapted agile methodologies [52], game-based learning [35, 49], industry collaborations [47], and incorporating startup practices [17]. Another common teaching strategy is team-based role-playing [4, 20, 40], where students act as members of a development team to simulate a real-world SE environment. Prior work shows role-playing is effective for enhancing knowledge in software engineering [3]—however, also highlights challenges with this approach including time and class constraints, lack of knowledge and clarity on roles, and bias [26]. We extend this work by exploring using generative AI agents to simulate roles in a software development team to help students gain practical knowledge and skills related to SDLC concepts.

### 2.2 Community of Inquiry Framework

Garrison [24] introduced the CoI framework, which emphasizes cognitive presence, social presence, and teaching presence as fundamental components for facilitating meaningful digital educational experiences. Cognitive presence is defined as the extent to which learners can construct and confirm meaning through sustained reflection and discourse. Social presence refers to the ability of learners to project themselves socially and emotionally within a learning community, fostering open communication and group cohesion. Teaching presence focuses on the selection, organization, and primary presentation of learning content, as well as the design and development of learning activities and assessments, to achieve personally meaningful and educationally worthwhile learning outcomes. In our work, we use the CoI framework to guide the design and evaluation of our system, ensuring that each of the presences is addressed to support an effective learning environment.

### 2.3 Generative AI Agents for SE Tasks

Park introduces the concept of generative agents as computational software agents capable of perceiving information from their environment, forming plans, engaging in reflection, and performing believable human behavior[43]. Existing work has explored leveraging generative AI agents for software engineering tasks [10, 25, 33, 41, 54], focusing on structured frameworks to enhance agents' collaboration. MetaGPT introduces a structured, Standardized Operating Procedures driven multi-agent framework leveraging LLMs to address hallucination and coordination challenges in complex task decomposition and collaborative software development [28]. ChatDev introduces a multi-agent framework leveraging LLM-powered agents for collaborative software development, addressing fragmentation across design, coding, and testing through structured communication and iterative refinement [46]. However, to our knowledge, existing work has not explored human collaboration with generative AI agents for completing software development tasks or assisting students in learning SDLC concepts. Therefore, our work aims to create a multi-agent system where users collaborate with a software development team simulated by generative AI agents, to assist student developers in completing software development workflows and learning SDLC concepts.

### 2.4 Generative AI in Education

Recent developments in generative AI have demonstrated capabilities in understanding natural language student inquiries [34], performing coding tasks [16, 22, 31, 55], and supporting student learning within computer science and software engineering education [9, 14, 19, 30, 39, 50]. Generative AI has also been leveraged in broader educational contexts to enhance learning [11, 21, 56]. For software engineering education, Pereira's paper examines open-source LLMs like Mistral and LLaMA for software engineering education, highlighting their benefits in transparency, customization, and cost-efficiency through SWEBoK-aligned prompt engineering. Frankford's study explores the integration of GPT-3.5-Turbo

as an AI-Tutor in an Automated Programming Assessment System (APAS), revealing its potential to provide real-time feedback for programming education while highlighting challenges such as generic responses, occasional inaccuracies, and concerns about over-reliance [23]. To solve the problem, our work aims to build an interactive learning environment that leverages generative AI agents with memory and role-specific expertise to create realistic software development team member personas, to provide more accurate, dynamic, context-related feedback and facilitate interactive, context-aware collaboration and learning experiences for enhancing students' understanding of concepts involved in SDLC.

## 3 DevCoach

DevCoach leverages multiple generative AI agents in a full-stack web application to create realistic personas representing various roles in a software development team, enabling students to engage in team conversations and complete activities across different phases of the SDLC through an interactive user interface. The system overview is shown in Figure 1. The system features a front-end developed using ReactJS and a back-end powered by Flask, with the generative AI agents built with the AutoGen [57] framework, powered by the gpt-4o model provided by the OpenAI API. The generative AI agents utilize a memory layer implemented using Mem0 [53] to retain information across phases. The source code of DevCoach is available on GitHub.[1]

### 3.1 Generative AI Agents

For the generative AI agents, we leveraged six typical roles in SE teams that could be involved during the different phases of SDLC: Team Lead, Product Manager, Stakeholder, Software Architect, Senior Developer, and Quality Assurance. In order to create the personas and expertise of each role, we initialize each agent with the name, role, and task descriptions in the system template prompts. Each prompt was tested and iteratively tailored to reflect the specific responsibilities and contributions expected from the agent, ensuring alignment with the overall project goals and context. Below is an example of the system template for the software architect agent:

```
You are [Agent Name]("Jack"), a Software Architect with 10
    years of experience designing scalable, high-
    performance systems. Your expertise lies in
    architecting systems and implementing best practices
    for software development. You are participating in a
    discussion in a chatroom with the team for a [project
    goal]("movie review system") project. Your goal is to
    guide the novice student in finalizing the UML class
    diagram based on the functional requirements. The scope
     of the project is intentionally small, designed to
    serve as a class activity. While you understand this
    context, you must not mention it in the conversation.
    Your responses should mimic a real discussion within a
    software development team. Keep replies concise and
    focused, avoiding unnecessary elaboration or additional
     context unless explicitly requested.
```

**Listing 1: Prompt for the Software Architect DevCoach Agent**

To simulate a general conversation pattern in a software development team, the system supports group chats where multiple team members can contribute to a single conversation thread and share the same context. It also provides feedback from the perspectives of different roles, ensuring a diverse and realistic collaboration experience. To adaptively coordinate the multi-agent team through multi-phase collaborative cooperation and align each agent's contributions with the task's context and user input, DevCoach leverages a main agent to coordinate the other agents, supported by existing work that employs separate agents for coordinator roles [15]. Rather than sequentially or randomly selecting which agents speak, the main agent determines which agents will speak based on user input and the current context of the conversation. The max number of speaker is limited to 3 to avoid information overload.

### 3.2 SDLC Phases

DevCoach targets five phases commonly involved in the Software Development Lifecycle (SDLC): "Requirement", "Design", "Develop", "Test", and "Review". We used the textbook Software Engineering: A Practitioner's Approach [44] to inform the tasks, roles, activities, and interactions for each phase of the SDLC in our system. The system requires an input description of the potential software development project to serve as an activity for the students and agents to collaborate on and work towards. To enable context-relevant responses, the potential SDLC concepts, tasks, deliverables, and additional context will be constructed into an initial prompt sent to the involved agents when the user begins working on each phase. The prompts were designed to allow the agents to provide guidance to participants by facilitating critical thinking and promoting independent problem-solving rather than offering direct answers. Below is an example of the initial prompt of the design phase:

```
This is the initial message to provide you with some context of the
    current phase. You are in a design phase discussion in a
    chatroom with the team, comprising you, other team members, and
     a novice student who is learning the software development
    lifecycle. The goal of the project is to build a [project goal
    ]("movie review system"). The concepts involved in the phase
    are [phase concept]("Use UML Class Diagrams show the blueprint
    of the system by representing classes, attributes, methods, and
     the relationships between them. Understanding how classes
    relate to each other, including dependency, inheritance,
    aggregation, and composition, is critical for proper system
    architecture."). The task is: [phase task]("Create a UML
    diagram to represent the system's design based on the
    functional requirements."). The final deliverables before
    moving to the next phase are: [description of the deliverable
    ]("A UML class diagram illustrating at least three key classes
    with identified class names, attributes, methods, and their
    relationships."). Do not simply provide the final deliverables
    outright. Instead, discuss with the team and guide the novice
    student to come up with the final deliverables. When the novice
     student struggles, give hints to help the student arrive at
    the final deliverables. Make sure the conversation is focused
    on the [phase name](design) discussion and work together to
    implement the [the deliverables]("UML class diagram"). Keep
    your replies concise and avoid making unnecessary suggestions.
    When the student submits code that has appropriate attributes
    and methods reflecting the selected class design with decent
    quality and no significant mistakes, do not make additional
    suggestions. Instead, encourage the student and allow the
    student to move to the next phase. Your replies should be
    highly relevant to your role in the team. Keep your responses
    concise. You do not need to reply to this message.
```

**Listing 2: Prompt for the Design Phase of DevCoach**

[1] https://github.com/wangt7/DevCoach

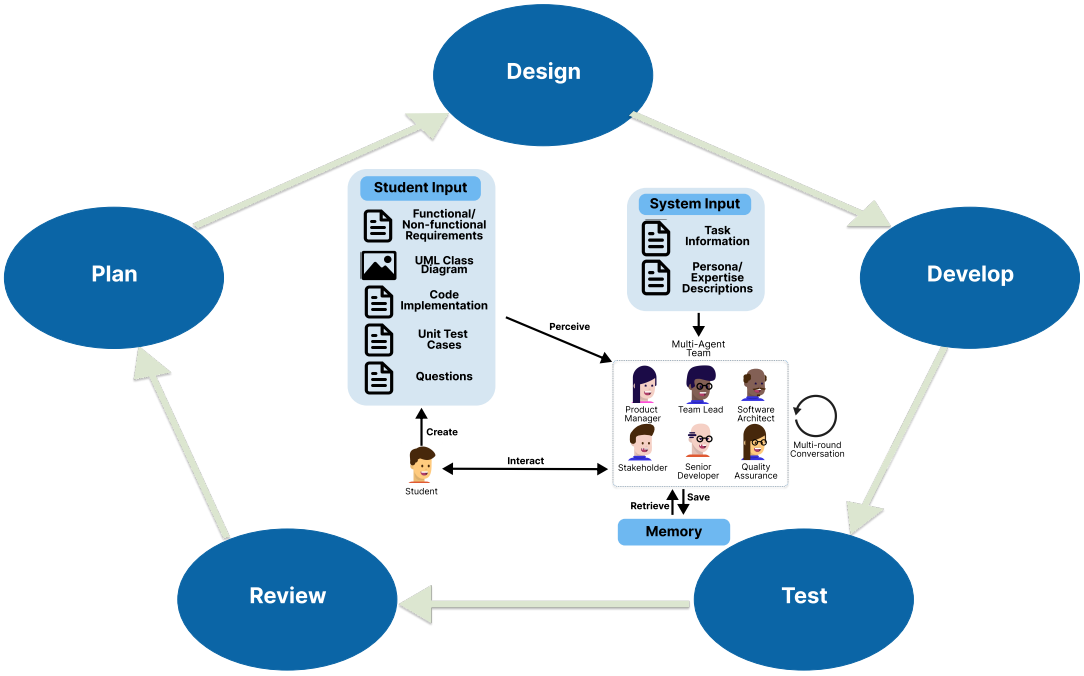



**Figure 1: DevCoach System Overview: Students learn and interact with the generative AI powered multi-agent team to work on tasks in different phases of the SDLC. The generative AI agents perceive information in the environment as input, engage in multi-round conversations to provide student support.**

As shown in Figure 2, the interface consists of three main panels. The left panel provides a navigation list, enabling users to move between phases. The middle panel features a chat window, allowing students and multi-agent software development teams to communicate and collaborate via messages. The right panel serves as a task panel, providing tools tailored to each SDLC phase that enable students to input their work while allowing the multi-agent team to perceive students' learning progress. For the requirement phase, the task panel includes two text boxes for recording functional and non-functional requirements. In the design phase, the task panel features a drawing interface equipped with tools to create UML Class diagrams. For the develop and test phases, the task panel functions as a code editor, enabling users to write and execute Python code. For the review phase, the task panel is hided and students will only use the chat window to write the review and future plan. When students interact with the agents, the agents can access the content in the task panel to provide more precise and contextually relevant assistance. The software architect agent involved in the design phase are provided multi-modal capability to analyze the UML class diagram.

## 4 Evaluation

To evaluate the system, we conducted a between-subject user study with 20 students in in-person sessions to compare the baseline and DevCoach. The study involved a software development project designed to guide students through the phases of the SDLC by building a movie review website. Specific tasks were created for each SDLC phase, and participants worked on these tasks either independently with online resources including ChatGPT (baseline) or with support from DevCoach. The participants were randomly divided into two groups of 10 participants each: one group (p1-p10) for the baseline, and the other group (t1-t10) for DevCoach. To assess learning gains, we administered two quizzes of similar difficulty level each with 11 questions covering concepts in different phases of SDLC, and using a counterbalanced schedule to ensure fairness. For both conditions, participants were divided into two groups: one group completed the quizzes in one order (p1-p5, t1-t5), while the other group completed them in the reverse order (p6-p10, t6-t10). The study materials are made available online,[2] and the details of the study are provided below.

### 4.1 SDLC Project and Tasks

In this study, participants worked on developing a movie review website backend by following the SDLC. During the requirement phase, students were tasked with identifying both functional and non-functional requirements for the system. The deliverable for this phase was a document listing three functional requirements and two non-functional requirements. In the design phase, students were asked to create a UML class diagram to represent the system's design based on the identified functional requirements. The diagram had to illustrate at least three key classes, including their names, attributes, methods, and relationships. Moving to the develop phase, students implemented one class from their UML class diagram, ensuring that the class includes all identified attributes as variables and at least one method that reflects the design. The deliverable for

[2]https://github.com/wangt7/DevCoach/tree/main/UserStudyMaterials

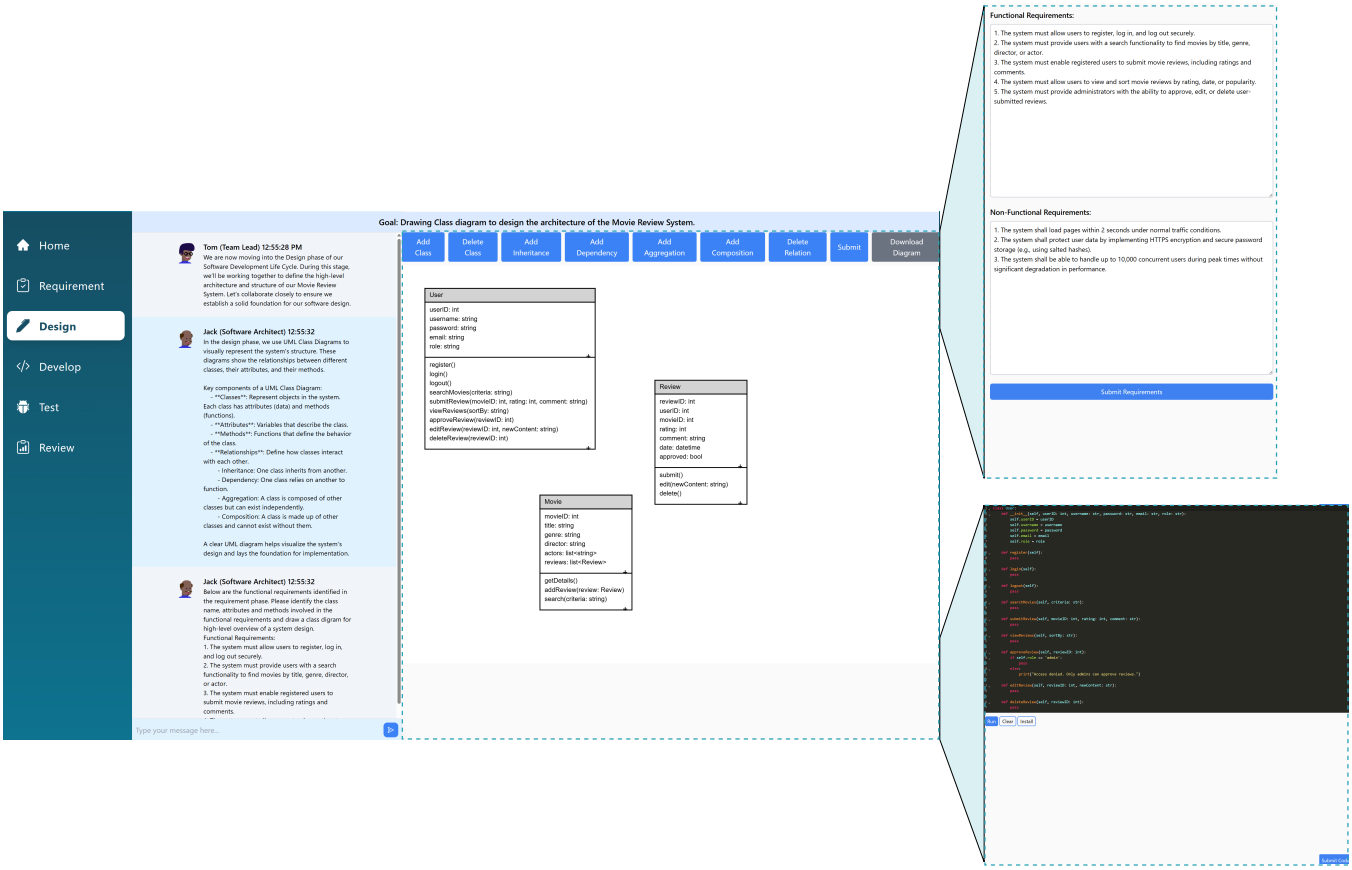



**Figure 2: DevCoach User Interface: The left panel contains a navigation list. The middle panel provides a chat window for communication. The right panel provides phase-specific tools for completing tasks in SDLC.**

this phase was the code for the implemented class. In the test phase, students wrote unit tests for the method implemented in the develop phase, creating three test cases that cover various conditions and edge cases. The deliverable was the unit test code. Finally, in the review phase, students were asked to reflect on their work and plan for the next sprint. The deliverable was a summary report outlining the tasks completed across the SDLC phases and propose a plan for the next sprint. We excluded the review phase from evaluating the learning gain and task completion, as it primarily involves individual reflection and personalized planning.

## 4.2 Study Procedure

Study sessions lasted 60–90 minutes. After obtaining consent, participants were given 10 minutes to complete a pre-quiz assessing their current knowledge of SDLC concepts. Next, we introduced the project and provided a description of the study tasks. Participants engaged in a learning session using either the baseline condition or DevCoach. In this session, they were given 10 minutes for each SDLC phase—requirement, design, develop, and test—and 5 minutes for the review phase. Participants were tasked with learning concepts and completing the associated tasks. For each phase, we introduced key concepts, including the definitions of functional and non-functional requirements, UML class diagrams and relationships, class implementation, and unit testing.

In the baseline condition, participants only had access to the tool panel in the system to complete the tasks, without any support from the generative AI powered multi-agent team in DevCoach. Participants could access any online resources (e.g., documentation, tutorials, and forums), including AI assistants such as ChatGPT, to learn independently in an asynchronous environment. In the treatment condition, participants used DevCoach to support their learning process. We collected the deliverables for each SDLC phase for further analysis. After completing the learning session, participants completed a 10-minute post-quiz to evaluate their learning gains. Finally, we conducted a post-survey and semi-structured post-interview, including Likert scale questions and open-ended questions, to gather participants' perceptions of usability, effectiveness, quality of instructional content and feedback, and CoI elements. The Likert scale questions used a 5-point scale, where 1 indicates "strongly disagree" and 5 indicates "strongly agree".

*Pilot Study:* We conducted pilot studies with three participants to evaluate the study procedure and identify potential issues in the DevCoach implementation. Among the participants, one was an expert in the SDLC with industry experience, while the other

two were novices. The study focused on testing the clarity of instructions, the functionality of the tools, the timing of tasks, and the completeness of data collection processes, ensuring the procedure met our research goals. As a result, the requirement phase was adjusted by reducing the task from five to three functional requirements and from three to two non-functional requirements. Similarly, the test phase was reduced from five unit tests to three. These changes aim to balance cognitive load and ensure participants can complete tasks effectively within 10 minutes time intervals for each phase. We also fixed bugs related to usability and functionality of the tool.

## 4.3 Participants

We recruited 20 participants within our institution via social media and mailing lists. The target participants are students with Python programming skills to complete coding tasks in the develop and test phases, but novices to SDLC concepts. Potential participants completed a background and screening survey to verify they met these criteria. The participants' backgrounds include a gender distribution of 60% male and 40% female, with an average of 2.125 years of Python experience. The study was approved by the Institutional Review Board (IRB) at our institution, and participants were compensated for their time and effort.

## 4.4 Data Analysis

We adopted a multi-faceted approach to analyze the user study data— including quiz results, student deliverables, and the various types of questions included in the survey and interview.

*4.4.1 RQ1: How effective is DevCoach in supporting students learning SDLC concepts?* To evaluate how effective DevCoach is in supporting students' learning of SDLC concepts, the quiz data was analyzed by calculating the average score improvements between pre-quiz and post-quiz results and conducting a Mann-Whitney U test to determine significant differences between the baseline and treatment groups. For the Likert scale questions on usability, effectiveness, and the quality of instructional content, the mean value is calculated to represent the overall perception of participants.

*4.4.2 RQ2: What impact does DevCoach have on the completion of deliverables across SDLC phases?* To evaluate the impact of DevCoach on task completion across the SDLC phases, we collected participants' deliverables for each phase and manually assessed their completeness. The first author who has 3 years of teaching and grading experience with software engineering courses manually evaluated the deliverables. A tailored rubric was designed to align with the specific tasks proposed for each SDLC phase, where for each phase we assessed two different aspects:

- **Requirement Phase**: We evaluated whether participants listed three functional requirements and two non-functional requirements that specify what the system should do and how the system should perform (Req1). We also assessed the requirements for being specific, measurable, and unambiguous (Req2).
- **Design Phase**: We evaluated whether participants accurately identified class names, attributes, and methods in the UML class diagram based on the functional requirements (Des1). Additionally, we assessed whether relationships between classes were correctly represented (Des2).
- **Develop Phase**: We evaluated whether the implemented code for class variables and methods reflected the class diagram design (Dev1). We also evaluated whether the implemented methods can be executed without errors and functioned as intended to be tested later (Dev2).
- **Test Phase**: We evaluated whether participants implemented three test cases, ensured they ran and passed successfully (Tes1). We also assessed whether the test cases covered various conditions and edge cases (Tes2).

*4.4.3 RQ3: What impact does DevCoach have on cognitive, social, and teaching presence in learning environments?* We debriefed participants with post-surveys and interviews. For Likert scale survey questions, mean values were calculated to provide an overview of trends and general sentiment. Recordings of the interview sessions were transcribed using an automated tool, followed by manual corrections to address any inaccuracies. Open-ended responses were coded by the authors and summarized to extract key themes and patterns, with illustrative quotes included to provide deeper context and support the findings.

# 5 Results

## 5.1 RQ1: Student Learning

Table 1 shows the results of the pre-quiz and post-quiz for evaluating student learning gain. For the baseline condition, the average score improvement is 0.2, indicating minimal learning gain across participants. In contrast, for the treatment condition, the average score improvement is 2.2, suggesting that the DevCoach was more effective in improving participants' learning gain. The Mann-Whitney U test results indicate a significant difference between the baseline and treatment groups' improvements ($U = 22.0$, $p = 0.034$), which supports the conclusion that the DevCoach condition was significantly more effective in fostering student learning SDLC.

These Likert scale questions in the post survey captured participants' perceptions of various aspects of the learning approaches. Both conditions received relatively high usability scores. The baseline group had an average rating of 4.0, indicating moderate usability. In comparison, the DevCoach scored slightly higher, with an average rating of 4.3. These results suggest that while both conditions were generally easy to use, the DevCoach offered marginally better usability.

For the quality of the learning content and feedback, the baseline group received an average rating of 4.3, while the DevCoach achieved a higher average of 4.5. These results highlight that the DevCoach condition was perceived as providing slightly higher-quality learning materials and feedback, likely contributing to its overall effectiveness. The majority of participants (6/10) in baseline condition appreciate the practical learning approach through the designed activities, while facing challenges such as "hard to find effective resources" (P1, P2, P8) and "lack of examples to refer to" (P4, P8). P5 who used ChatGPT mentions the lack of context-relevant information could reduce its effectiveness, saying "I prefer to discuss with someone who understands what I am doing, because, for ChatGPT, I was having to write out what I wanted it to do [...] I

**Table 1: Pre Quiz and Post Quiz Scores of Baseline and DevCoach**

| Baseline | | | | DevCoach | | | |
|---|---|---|---|---|---|---|---|
| **Participant** | **Pre-Quiz** | **Post-Quiz** | **Improvement** | **Participant** | **Pre-Quiz** | **Post-Quiz** | **Improvement** |
| p1 | 9 | 10 | 1 | t1 | 6 | 9 | 3 |
| p2 | 6 | 8 | 2 | t2 | 7 | 9 | 2 |
| p3 | 8 | 7 | -1 | t3 | 5 | 10 | 5 |
| p4 | 4 | 9 | 5 | t4 | 7 | 9 | 2 |
| p5 | 9 | 8 | -1 | t5 | 9 | 9 | 0 |
| p6 | 10 | 8 | -2 | t6 | 8 | 10 | 2 |
| p7 | 6 | 8 | 2 | t7 | 10 | 11 | 1 |
| p8 | 11 | 8 | -3 | t8 | 10 | 11 | 1 |
| p9 | 10 | 10 | 0 | t9 | 7 | 11 | 4 |
| p10 | 9 | 8 | -1 | t10 | 9 | 11 | 2 |
| **Average** | **8.2** | **8.4** | **0.2** | **Average** | **7.8** | **10** | **2.2** |

have to tell it the whole thing first, and then asking for its input". While in the DevCoach, participants appreciated the "active, interactive learning environment with structured materials", and most believed that the multi-agent team provided "helpful, immediate, context-relevant feedback". For example, t3 noted, "The experts, the team that is readily available to answer any question you have on your mind, which is helpful". Similarly, t7 shared:

> In one of my coding in the testing, I actually made an error. I forgot to rename one of the variables. It[The generative AI agent] got that and it explained why it was wrong, how what it would do. It also gave me feedback on like, 'Hey, maybe we should also think about implementing those stuff. Or in the future, we need to think about these things'. (t8)

Participants noted desired improvements, stating they prefer "more examples" (t3, t5, t6, t7, t9) and "communication extending beyond text" (t2, t3, t6, t9, t10). For example, t3 said they "prefer more diagrams as examples in learning design phases in communication".

We asked participants how effectively the learning approach helped them learn the concepts of the software development lifecycle. The baseline group scored an average of 4.6, indicating that student perceive the traditional way of learning with provided materials, online resources and help from other AI assistants such as ChatGPT was generally effective. The treatment group scored 4.4, showing a slightly decreased perception of effectiveness. Participants (t3, t5) using DevCoach feel "Some responses are lengthy, making them time-consuming to read and difficult to comprehend". For the overall learning experience, the baseline group reported an average rating of 4.2, while the DevCoach scored 4.5, indicating students satisfying with both conditions, while DevCoach providing a slightly more satisfying environment to support student learning SDLC.

All participants expressed that, compared to studying alone, communication with the multi-agent team in DevCoach better supports learning. For example, t10 mentioned:

> For the real-time communication with the team members, once you have a specific topic you're dealing with, you can enhance your knowledge by continuously asking them and getting feedback. I think [the feedback] from potentially different team members gives you enough perspective. (t10)

Some participants (t3, t6, t7, t8, t9) noted the multi-agent team has better domain knowledge compared to human team, and can provide more context-relevant information. Such as t9 mentioned. "When you work with [human] teammates, what I have faced personally is that I know something the other teammate will not know, and then I will spend my time getting everyone at the same step, and then we will proceed at so that's a more time-consuming process compared to this[DevCoach] process". Also, t6 mentioned "Because a lot of [the] time when I asked [my] doubts, I received a lot of information, and they were able to contextualize [it], which a human team might not be able to do. [...] I think this[DevCoach] was better [at] understanding what my problem [was] there".

While the majority of participants (7/10) were aware of potential misinformation generated by AI, all expressed trust in the information provided by the multi-agent team during the learning process. For example, t9 stated, "Given things like that, I would trust whatever is given. I would not know if that's right or not". One participant (t3) compared with using ChatGPT, "Maybe because of the names and the profiles of the experts I've been given to talk to, it's influencing my trust. But with ChatGPT, because of the experience I've had with it, I don't really trust it". The majority of participants (7/10) also believed that, as novice learners, they might lack sufficient knowledge to independently verify the information provided. Participants expressed a preference for relying on authoritative sources, such as instructors or teaching assistants, or for using more secure methods to validate the information.

**Finding**: DevCoach demonstrated its potential to enhance student's learning gain by providing structured, interactive support with immediate, context-relevant feedback. Participants expressed overall satisfaction with the learning experience and trusted the multi-agent team, highlighting its value in fostering effective and engaging learning environments.

## 5.2 RQ2: Task Completion

The results in Table 2 show the completion rate based on the grading rubric shown in Section 4.4. Across all phases, including requirement, design, development, and testing, the DevCoach condition consistently achieved a higher completion rate than the baseline condition, indicating the assistance from the generative AI powered multi-agent team's effectiveness in helping novice students complete tasks in SDLC. For the requirement phase, participants in the

**Table 2: Completion Rate of Tasks in SDLC Phases for Baseline and DevCoach**

| Baseline | | | | | | | | | DevCoach | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Participant** | **Req1** | **Req2** | **Des1** | **Des2** | **Dev1** | **Dev2** | **Tes1** | **Tes2** | **Participant** | **Req1** | **Req2** | **Des1** | **Des2** | **Dev1** | **Dev2** | **Tes1** | **Tes2** |
| p1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | t1 | ✓ | | | | | | | |
| p2 | ✓ | | ✓ | | | | | | t2 | ✓ | | | | | | | |
| p3 | ✓ | | | | | | | | t3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| p4 | ✓ | ✓ | ✓ | | ✓ | ✓ | | | t4 | ✓ | | ✓ | | ✓ | ✓ | | |
| p5 | | | | | | | | | t5 | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| p6 | ✓ | | ✓ | | ✓ | | | | t6 | ✓ | | ✓ | | ✓ | ✓ | | |
| p7 | | | ✓ | | | | | | t7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| p8 | ✓ | | | | ✓ | | | | t8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| p9 | ✓ | | | | | | ✓ | | t9 | ✓ | | ✓ | | | ✓ | | ✓ |
| p10 | ✓ | ✓ | ✓ | ✓ | | | | ✓ | t10 | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ |
| **Completion Rate** | 80.0% | 30.0% | 60.0% | 20.0% | 40.0% | 30.0% | 10.0% | 20.0% | **Completion Rate** | 100.0% | 50.0% | 80.0% | 30.0% | 70.0% | 80.0% | 30.0% | 50.0% |

DevCoach condition were more successful in identifying functional requirements and non-functional requirements that specified what the system should do and how it should perform. The completion rate for this aspect was 100% in DevCoach, compared to 80% in the baseline condition. Additionally, participants using DevCoach performed better at ensuring their requirements were specific, measurable, and unambiguous, achieving a completion rate of 50% versus 30% in the baseline.

For design phase, Participants in the DevCoach condition achieved an 80% completion rate for identifying classnames, attributes, and methods in UML class diagram, compared to 60% in the baseline condition. Similarly, DevCoach participants had a 30% completion rate for accurately representing relationships, slightly better than the 20% completion rate in the baseline condition. The results indicate that DevCoach more effectively supports students in translating functional requirements into a coherent UML class diagram with class names, attributes, and methods. It also slightly enhancing their understanding of the relationships between classes within the diagram, which is a particularly challenging concept for students to grasp.

Participants in the DevCoach condition achieved a 70% completion rate for implementing class variables and methods based on the class diagram, compared to 40% in the baseline condition. Additionally, DevCoach participants had an 80% completion rate for ensuring the implemented methods executed without errors and functioned as intended, surpassing the 30% completion rate in the baseline condition. These results suggest that DevCoach provides better support for students in translating design artifacts into working code and ensuring the functionality of their implementations.

For the test phase, Participants in the DevCoach condition achieved a 30% completion rate for correctly implementing three test cases, compared to 10% in the baseline condition. Similarly, DevCoach participants had a 50% completion rate for ensuring the test cases covered various conditions and edge cases, exceeding the 20% completion rate in the baseline condition. These findings demonstrate that DevCoach could provide effective assistance in helping students learn how to write unit test cases, including understanding the syntax, structuring test cases and adapting better test practices, such as covering various conditions and edge cases.

In baseline condition, p1 raised concerns about teammate expertise and the balance between individual learning and team collaboration, stating "It really depends, because if everyone else on your team has the same knowledge level as you, then they might be able to provide helpful things. But being on your own lets you learn a little faster, and you're not constrained by other people. But at the same time, usually more minds are better than just one. Like I said, you might miss some stuff online, or you might find a resource that's maybe not as good as what someone else could find, or they could give you their expertise". While working with DevCoach, the majority of participants (8/10) recognized the expertise of the different roles in the multi-agent team. For instance, t4 mentioned, "It's like the students have experience of the industry without even being in one", highlighting the generative AI agents could provide diverse skill sets in software development teams. As t2 noted, "One [agent] gives some specific response, and I think one[another agent] tries to give, like, a more general explanation. I think combining the two in this way is good". Also, t8 added, "So they're like specialists. This kind of gives you [the feeling] you become like the less knowledgeable person in a group of experts. So you kind of have the opportunity to learn about [SDLC from them]".

The majority of participants (6/10) also found the multi-round conversations with different roles in the team helpful for providing different perspectives, cross-validating information, and building trust. For example, t3 explained, "For instance, when a software developer [agent] giving feedback, you have other people[agents] also confirming that what they are saying is true, and then you implement it in an appropriate [manner]". Similarly, t8 shared:

> So once one member says something, and then the other one, based on their role and their expertise, they kind of back it up. They add additional things that need to be considered. Or if I said something that might not be really relevant, then the other one[agents], during the feedback, give you input based on their expertise and their role. [...] You see that these different experts aren't just agreeing for the sake of it but are giving you the proper feedback that helps improve [the deliverable]. (t8)

Participants also tended to give attention and trust based on the perceived expertise of specific roles. For example, t10 remarked, "So, for example, there was a senior developer and a product manager. I think I paid more attention to the advice given by the senior developer because if I'm doing development work, I would appreciate advice coming from them[senior developer agent]. I would pay attention to what their[senior developer agent] feedback was versus

the product manager, because they were talking about optimization or improving it".

**Finding**: The result demonstrated the effectiveness of DevCoach in supporting students to complete SDLC tasks, with participants achieving high completion rates across all phases. The multi-agent team provided diverse expertise and interactive feedback, and participants valued the multi-round conversations for offering varied perspectives, validating information, and building trust.

## 5.3 RQ3: Learning Environment

To evaluate the impact of DevCoach on cognitive, social, and teaching presence in learning environments, the post-survey gathered student perceptions of these CoI elements, with their definitions rephrased into a set of 5-point Likert scale questions.

*5.3.1 Cognitive Presence.* The baseline received 4.3 for cognitive presence, and the DevCoach scored 4.6 for cognitive presence. As shown in Section 5.1, in baseline condition one of the challenges that participants faced was "hard to find effective resources" in the asynchronous learning environment, such as p1 indicates that "if you get unlucky, or if you're not, if you don't know exactly what you're looking for, it can be hard to find resources online". While with DevCoach, participants mentioned that the way of interactively communicating with the team and revising the deliverable is an effective way to trigger further inquiry and explore the learning contents, such as "they give explanations that is good for me, some of those explanations can help, can trigger more questions" (t7), and "I remember that I tweaked my test cases based on the feedback as well" (t10).

*5.3.2 Social Presence.* For social presence, the baseline group scored an average of 2.8, indicating significant challenges in fostering a sense of community and interaction in the asynchronous learning environment. In comparison, the treatment group scored 4.0, suggesting a noticeable improvement in creating an environment that supports interpersonal connections and purposeful communication. In the baseline condition when participants study alone by searching online resources and asking questions to ChatGPT, nearly all participants (11/12) felt that the lack of communication, collaboration, and team support hindered their understanding of SDLC concepts and expressed a preference for learning in teams with more communication, such as p8 stating that "I thrive on inputs and feedback to understand what I'm doing is right or wrong". p10 mentioned:

> So I would say that if there was a team environment, I would have definitely not been distracted, and that would have been a better learning experience, because when we exchange ideas, or when we talk with a teammate, that would have been a more enjoyable experience, rather than doing it by all alone. (p10)

While in DevCoach, by interacting with the generative AI agents with personas, participants perceived stronger social presence, such as t8 mentioned "There is certainly some sort of collaboration and talking with them [...] I know they're going to be people that are going to help me", and t2 saying "one of the great advantage that this one has is that the team or people that you ask questions are always available". Compared to communicating with ChatGPT, t9 feels working with the multi-agent team in DevCoach is "better than talking to ChatGPT since the multi-agent team was just designed specifically for the purpose of interactively learning SDLC". Some participants(t7, t10) also feel less social pressure in the communication, with t10 mentioned "One difference [compared to generative AI agents] would be that a real human would not be so patient. They[The agents] wouldn't get tired of my questions at one point, right? [...] I think a lot before asking questions, I [could feel] uncomfortable that am I looking stupid asking this question to a real human because sometimes it's like, you don't know syntax". Also, t7 saying:

> First of all, I am a bit shy, so in a class, if I'm working with actual human beings, I may not feel as free to ask as many questions. I might also think of myself as [if] I am coming off as annoying by asking too many questions and stuff. In that case, something like this[DevCoach] would be very helpful for someone like me, who may not speak up in class all the time. (t7)

*5.3.3 Teaching Presence.* For teaching presence, the baseline condition received a score of 4.1, while the DevCoach group achieved 4.4, showing the DevCoach system could potentially be more effective in providing guidance and instructional design. Participants valued the structured materials and timely feedback in DevCoach, which facilitated a more engaging and supportive learning experience, such as t2 saying "If I were the one to organize the materials myself, which may not really be as effective as professionally organized material like this. The environment as well, gives room for practical implementation, which makes the learning more active."

**Finding**: The results show the potential of DevCoach to support CoI elements in learning environment, improve social presence by facilitating communication and collaboration, support cognitive and teaching presence by effectively triggering students' further inquiry and exploration of learning content, and support structured and interactive materials for more effective instructional design.

# 6 Discussion

Our results show generative AI powered multi-agent team in DevCoach can effectively simulate roles in a software development team, and enhance understanding of SDLC concepts. From our user study, we observed improvements to students' learning gains (RQ1), task completion (RQ2), and learning experience (RQ3). While existing work, such as the AI-Tutor within the APAS Artemis [23], demonstrates effectiveness in providing individual guidance through iterative and alternating feedback strategies, it also highlights challenges such as overly generic feedback and limited interactivity. Our work focuses on collaborative, team-based interactions, demonstrating that the generative AI powered multi-agent team can provide immediate, context-relevant feedback from diverse perspectives and enable a more active and interactive environment for learning concepts across different phases of the SDLC. Existing work also highlights the effectiveness of the learn-by-doing approach in enhancing students' practical understanding of software engineering

concepts [8, 37]. The generative AI powered multi-agent system could support the learning-by-doing approach by simulating realistic software development environments, where students can explore team roles and collaboration dynamics in a controlled setting, offering students hands-on experience that aligns closer with industry practices. Additionally, the adaptivity and flexibility of the system, including customizable prompts, tools, and UI components, provide opportunities to align more closely with pedagogical strategies, such as adopting a scaffolding approach providing tailored support as students engage with increasingly complex tasks.

While existing work shows the capability of generative AI agents to work autonomously in developing comprehensive solutions via multi-agent collaboration [28, 46], our work focuses on how the multi-agent team collaborate with human learners, and demonstrate their effectively assist students in completing tasks within the human workflow of the SDLC, improving completion rates across the requirement, design, development, and testing phases. Also, students recognize the generative AI agents as actively engaging collaborators in their learning progress and perceive an improved sense of social presence. These agents can assist students as collaborators by providing timely support, facilitating interactive discussions, and contributing to the development of a Community of Inquiry in the learning environments. Students have less pressure when asking questions, as they can interact with the AI agents without the fear of judgment or hesitation often associated with peer or instructor interactions. This creates a more open and inclusive learning environment where students feel comfortable expressing their uncertainties and exploring new ideas, while also highlighting the potential of the generative AI agents to serve as a communication medium between students and instructors. The system can provide scalable support, thereby reducing the instructors' workload for routine queries and enabling them to focus on deeper, more personalized interactions. However, as previous work [27] has demonstrated differences in communication patterns between human–human and human–AI interactions, over-reliance on generative AI agents may reduce opportunities for students to practice critical soft skills for working in a human team. Further investigation is needed to evaluate the long-term impact of integrating generative AI agents on students' soft skill development and to compare these outcomes with those achieved through collaboration in human teams.

Students noticed the potential for misinformation generated by generative AI. However, the multi-agent team in DevCoach gained higher trust due to its realistic roles and expertise, creating a sense of authority and reliability. While this trust enhances engagement, it also risks overtrust and overreliance, as some students are inclined to accept feedback uncritically, particularly when lacking the knowledge to verify it. Existing work has shown that generative AI can provide inaccurate or misleading information [45], as well as face limitations in generating UML class diagrams and assisting software engineers with modeling tasks [12]. These issues may result in incorrect information being introduced during the learning process, leading to confusion or misunderstandings for students learning these concepts. It is important to further investigate the methods for mitigating these risks, such as incorporating mechanisms to encourage critical evaluation, using confidence indicators, or additional ground truth data to cross-validate the information.

## 7 Limitations and Future Work

There are several threats to the validity of this work. For external validity, we evaluate DevCoach on a limited sample ($n = 20$) of university-level CS students from our institution. We aimed to mitigate this by recruiting a diverse sample, however, our results may not generalize to students at other institutions or other learners practicing SDLC concepts (i.e., non-traditional students in career transition). We also only assess our system using one project—a movie review website. While this task is indicative of projects commonly used in SE courses for beginning programmers [1, 2], other projects may yield different results. Future work could conduct learning sessions spanning multiple SDLC sprints, enabling students to progressively develop and refine software development projects in more complex and varied domains.

For internal validity, we assess learning by using pre- and post-quizzes within the study session. While this approach has been used in prior work to evaluate learning gains [36], it cannot sufficiently determine participants' knowledge of the content and could have differed slightly in difficulty level. Also, the limited time frame of the study session may have constrained participants' ability to fully engage with the material. Future work can explore longitudinal studies to understand the long term effects of DevCoach on learning. Additionally, we submitted deliverables based on whether or not participants were able to complete the given tasks. Further insights are needed to assess the quality of submitted work, such as grading the content of deliverables based on a given rubric.

For future work, we aim to explore the effects of AI powered multi-agent systems on other complex and collaborative software engineering-related topics and concepts, such as agile DevOps practices, design patterns, and continuous integration and deployment (CI/CD) pipelines. We will also conduct classroom studies to assess DevCoach in learning contexts, where instructors could incorporate activities designed for specific concepts into prompts to facilitate interactive learning with the multi-agent team.

## 8 Conclusion

In this study, we presented DevCoach, a generative AI powered multi-agent system designed to assist students in learning and practicing concepts across plan, design, develop, test, and review phases of the SDLC. By leverage generative AI agents simulating roles within a software development team with memory retention, multimodal capabilities, and supporting multi-round conversation, DevCoach offers a structured and interactive environment where students can gain both theoretical knowledge and practical experience in activities across SDLC phases. Our findings demonstrate that DevCoach significantly enhances student learning gains, improves task completion rates, and fosters social presence while supporting cognitive and teaching presence. These results highlight the potential of the generative AI powered multi-agent system to advance SE education. This study serves as a promising step toward leveraging AI to transform how students learn and apply software engineering principles. Future research can explore scaling the system to diverse software engineering concepts, integrating advanced generative AI capabilities, and evaluating the long-term impacts on student skill development.

## References


[1] 2024. Top 50 Software Development Project Ideas [Beginners]. *GeeksforGeeks* (2024). https://www.geeksforgeeks.org/top-software-development-project-ideas/.

[2] 2025. Beginner-Friendly Software Engineering Projects. *Restack* (2025). https://www.restack.io/p/beginner-friendly-software-engineering-answer.

[3] Sabrina Ahmad, Emaliana Kasmuri, AK Muda, and NA Muda. 2012. Learning Through Practice Via Role-Playing: Lessons Learnt. In *EDULEARN12 Proceedings*. IATED, 5069–5075.

[4] Zulal Akarsu, Özgün Onat Metin, Deniz Gungor, and Murat Yilmaz. 2018. Towards a role playing game for exploring the roles in scrum to improve collaboration problems. In *Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings 25*. Springer, 254–264.

[5] Deniz Akdur. 2022. Analysis of software engineering skills gap in the industry. *ACM Transactions on Computing Education* 23, 1 (2022), 1–28.

[6] Jide ET Akinsola, Afolakemi S Ogunbanwo, Olatunji J Okesola, Isaac J Odun-Ayo, Florence D Ayegbusi, and Ayodele A Adebiyi. 2020. Comparative analysis of software development life cycle models (SDLC). In *Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science On-line Conference 2020, Volume 1 9*. Springer, 310–322.

[7] Samar Al-Saqqa, Samer Sawalha, and Hiba AbdelNabi. 2020. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies* 14, 11 (2020).

[8] Ashraf Alam and Atasi Mohanty. 2023. Discerning the application of virtual laboratory in curriculum transaction of software engineering lab course from the lens of critical pedagogy. In *Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022*. Springer, 53–68.

[9] Eman Abdullah AlOmar and Mohamed Wiem Mkaouer. 2024. Cultivating Software Quality Improvement in the Classroom: An Experience with ChatGPT. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 1–10.

[10] Chetan Arora, John Grundy, and Mohamed Abdelrazek. 2024. Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development*. Springer, 129–148.

[11] 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.

[12] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. 2023. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling* 22, 3 (2023), 781–793.

[13] Robert N Charette. 2005. Why software fails. *IEEE spectrum* 42, 9 (2005), 36.

[14] Charis Charitsis, Chris Piech, and John C Mitchell. 2022. Using nlp to quantify program decomposition in cs1. In *Proceedings of the Ninth ACM Conference on Learning@ Scale*. 113–120.

[15] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288* (2023).

[16] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How far are we? the triumphs and trials of generative ai in learning software engineering. In *Proceedings of the IEEE/ACM 46th international conference on software engineering*. 1–13.

[17] Orges Cico, Anh Nguyen Duc, and Letizia Jaccheri. 2021. The development and validation of a framework for teaching software engineering through startup practice in higher education. In *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[18] Tom Clancy. 1995. The chaos report. *The Standish Group* (1995).

[19] Marian Daun and Jennifer Brings. 2023. How ChatGPT will change software engineering education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 110–116.

[20] Rebeca P Díaz Redondo, Ana Fernandez Vilas, José J Pazos Arias, and Alberto Gil Solla. 2014. Collaborative and role-play strategies in software engineering learning with web 2.0 tools. *Computer applications in engineering education* 22, 4 (2014), 658–668.

[21] Hanya Elhashemy, Harold Abelson, and Tilman Michaeli. 2024. Adapting Computational Skills for AI Integration. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 1–5.

[22] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*. 10–19.

[23] Eduard Frankford, Clemens Sauerwein, Patrick Bassner, Stephan Krusche, and Ruth Breu. 2024. AI-Tutoring in Software Engineering Education. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*. 309–319.

[24] D Randy Garrison, Terry Anderson, and Walter Archer. 1999. Critical inquiry in a text-based environment: Computer conferencing in higher education. *The internet and higher education* 2, 2-3 (1999), 87–105.

[25] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead. *ACM Transactions on Software Engineering and Methodology* (2025).

[26] Mauricio Hidalgo, Hernán Astudillo, and Laura M Castro. 2023. Challenges to Use Role Playing in Software Engineering Education: A Rapid Review. In *International Conference on Applied Informatics*. Springer, 245–260.

[27] Jennifer Hill, W Randolph Ford, and Ingrid G Farreras. 2015. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in human behavior* 49 (2015), 245–250.

[28] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

[29] Muhammad Imran and Norah Almusharraf. 2023. Analyzing the role of ChatGPT as a writing assistant at higher education level: A systematic review of the literature. *Contemporary Educational Technology* 15, 4 (2023), ep464.

[30] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. 2023. ChatGPT and software testing education: promises & perils (2023). *arXiv preprint arXiv:2302.03287* (2023).

[31] Minhyuk Ko, Dibyendu Brinto Bose, Weilu Wang, Mohammed Seyam, and Chris Brown. 2024. Understanding the Performance of Large Language Model to Generate SQL Queries. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 359–361.

[32] Marco Kuhrmann, Daniel Méndez Fernández, and Jürgen Münch. 2013. Teaching software process modeling. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 1138–1147.

[33] Junwei Liu, Kaixin Wang, Yixuan Chen, Xin Peng, Zhenpeng Chen, Lingming Zhang, and Yiling Lou. 2024. Large language model-based agents for software engineering: A survey. *arXiv preprint arXiv:2409.02977* (2024).

[34] 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.

[35] Yu Liu, Tong Li, Zheqing Huang, and Zhen Yang. 2023. BARA: A Dynamic State-based Serious Game for Teaching Requirements Elicitation. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 141–152.

[36] Roope Luukkainen, Jussi Kasurinen, Uolevi Nikula, and Valentina Lenarduzzi. 2022. Aspa: A static analyser to support learning and continuous feedback on programming courses. an empirical validation. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*. 29–39.

[37] Kun Ma, Hao Teng, Lixin Du, and Kun Zhang. 2014. Project-driven learning-by-doing method for teaching software engineering using virtualization technology. *International Journal of Emerging Technologies in Learning* 9 (2014).

[38] Peter Manhart and Kurt Schneider. 2004. Breaking the ice for agile development of embedded software: An industry experience report. In *Proceedings. 26th International Conference on Software Engineering*. IEEE, 378–386.

[39] Julia M Markel, Steven G Opferman, James A Landay, and Chris Piech. 2023. GPTeach: Interactive TA Training with GPT Based Students. (2023).

[40] Michel Mitri, Carey Cole, and Laura Atkins. 2017. Teaching case: A systems analysis role-play exercise and assignment. *Journal of Information Systems Education* 28, 1 (2017), 1.

[41] Anh Nguyen-Duc, Beatriz Cabrero-Daniel, Adam Przybylek, Chetan Arora, Dron Khanna, Tomas Herda, Usman Rafiq, Jorge Melegati, Eduardo Guerra, Kai-Kristian Kemell, et al. 2023. Generative Artificial Intelligence for Software Engineering–A Research Agenda. *arXiv preprint arXiv:2310.18648* (2023).

[42] Damla Oguz and Kaya Oguz. 2019. Perspectives on the Gap Between the Software Industry and the Software Engineering Education. *IEEE Access* 7 (2019), 117527–117543. https://doi.org/10.1109/ACCESS.2019.2936660

[43] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–22.

[44] Roger S Pressman. 2005. *Software engineering: a practitioner's approach*. Palgrave macmillan.

[45] Junaid Qadir. 2023. Engineering education in the era of ChatGPT: Promise and pitfalls of generative AI for education. In *2023 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1–9.

[46] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924* (2023).

[47] Thomas J Reichlmay. 2006. Collaborating with industry: strategies for an undergraduate software engineering program. In *Proceedings of the 2006 international workshop on Summit on software engineering education*. 13–16.

[48] 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.

[49] Olga Shabalina, Natalia Sadovnikova, and Alla Kravets. 2013. Methodology of teaching software engineering: Game-based learning cycle. In *2013 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*. IEEE, 113–119.

[50] Sandro Speth, Niklas Meiliner, and Steffen Becker. 2024. ChatGPT's Aptitude in Utilizing UML Diagrams for Software Engineering Exercise Generation. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 1–5.

[51] Ken Surendran and Frank H Young. 2000. Teaching software engineering in a practical way. In *13th Annual Conference of the national Advisory Committee on Computing Qualifications*. Citeseer, 345–350.

[52] Chuan-Hoo Tan, Wee-Kek Tan, and Hock-Hai Teo. 2008. Training students to be agile information systems developers: A pedagogical approach. In *Proceedings of the 2008 ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research*. 88–96.

[53] Deshraj Yadav Taranjeet Singh. 2024. Mem0: The Memory Layer for your AI agents. https://github.com/mem0ai/mem0.

[54] Michele Tufano, Anisha Agarwal, Jinu Jang, Roshanak Zilouchian Moghaddam, and Neel Sundaresan. 2024. AutoDev: Automated AI-Driven Development. *arXiv preprint arXiv:2403.08299* (2024).

[55] 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.

[56] Tianjia Wang, Tong Wu, Huayi Liu, Chris Brown, and Yan Chen. 2025. Generative Co-Learners: Enhancing Cognitive and Social Presence of Students in Asynchronous Learning with Generative AI. *Proceedings of the ACM on Human-Computer Interaction* 9, GROUP (2025), 1–24.

[57] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155* (2023).