SIMPBL: A Multi-Agent Framework for Project-Based Learning

Daniel Zhang-Li[1*], **Joy Jia Yin Lim**[1*], **Binglin Liu**[1], **Shangqing Tu**[1], **Zijun Yao**[1], **Hao Peng**[1]
Jifan Yu[1†], **Haoxuan Li**[1], **Zhanxin Hao**[1], **Ye He**[1], **Zekun Li**[1], **Jiangyi Wang**[2]
Lei Hou[1], **Bin Xu**[1], **Xin Cong**[1], **Zhiyuan Liu**[1], **Huiqin Liu**[1], **Yu Zhang**[1], **Juanzi Li**[1†]
[1]Tsinghua University
[2]Harbin Institute of Technology
{zlnn23, lin-jy23}@mails.tsinghua.edu.cn
{yujifan, lijuanzi}@tsinghua.edu.cn

Abstract

Project-Based Learning (PBL) is an important learning method that promotes understanding and the acquisition of practical skills by training learners through a project. Such effectiveness in PBL often depends on sustained orchestration and collaboration. However, existing LLM-based learning tools provide partial assistance without explicitly modeling these roles, and overly comprehensive help provided by LLM can reduce learner autonomy. We propose SIMPBL, a multi-agent framework with an orchestrator agent that provides adaptive scaffolding from interaction logs and collaborator agents that support project work through boundary-aware collaboration. We conduct comprehensive evaluations to study the effectiveness of SIMPBL, where we observe a 14% improvement in learner examination score. Results from extensive studies further highlight the ability of SIMPBL to manage learning behavior and improve the learning experience. Code and materials are available at https://github.com/THU-MAIC/OpenMAIC-Project.

1 Introduction

Large Language Models (LLMs) have demonstrated strong potential in supporting educational activities (Chu et al., 2025; Wen et al., 2024). They can tutor students (Pal Chowdhury et al., 2024), simulate classroom environments (Zhang et al., 2025), and personalize learning experiences (Lim et al., 2025), often achieving promising results. However, **Project-Based Learning (PBL)**, a widely adopted learning approach that helps learners develop practical skills through sustained project work (Kilpatrick, 1918), remains underexplored in LLM-based studies. Given both the effectiveness and complexity of PBL, it is essential to explore how modern LLMs can be leveraged to support high-quality PBL experiences.

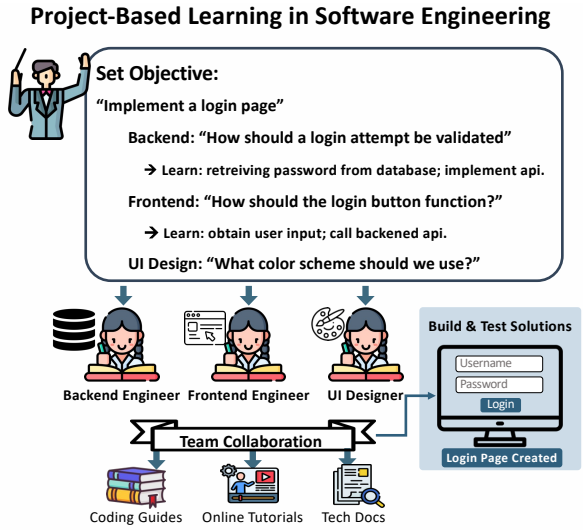

Figure 1: Example Project-Based Learning (PBL) in Software Engineering. PBL trains learners to obtain knowledge and acquire skills by solving a project.

As shown in Figure 1, **PBL organizes learners into teams that construct knowledge and skills through a shared project** (Thomas, 2000). For example, to learn software engineering, a classic PBL activity involves collaboratively developing a website. In such settings, learners assume complementary roles (e.g., backend engineer or frontend engineer). Throughout the project, the teacher periodically defines project objectives and pairs each objective with role-specific driving questions to scaffold learner progress. For instance, the objective "*Implement a login page*" for a backend engineer may be accompanied by the driving question "*How should a login attempt be validated?*". Such driving questions then drive the students with corresponding roles to explore knowledge. Guided by these questions, learners seek relevant resources and iteratively develop solutions, acquiring the necessary skills to achieve each objective (*e.g.*, the ability to "*retrieve password in the database through username and implement verification as web api*").

[*] Equal contribution.
[†] Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32755–32772

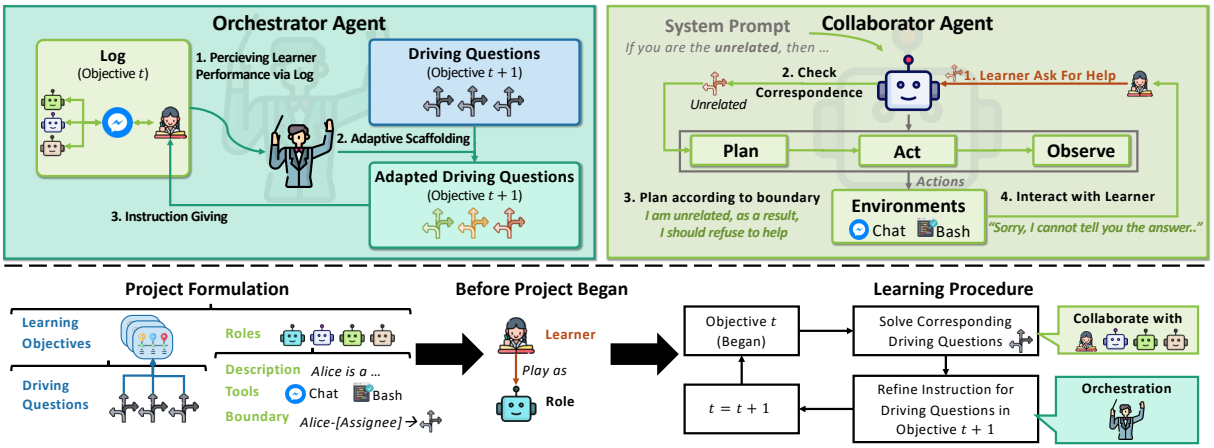

Figure 2: The overview of the SIMPBL framework. SIMPBL adopts the learning procedure where the learner and collaborators cooperate to complete the objectives. For each objective, a set of driving questions amended with instructions are used to guide learning explorations. The orchestrator agent adaptively refines the instructions between objectives. The collaborator agents interact with the learner to provide authentic collaboration while being bounded by the responsibilities with the corresponded driving question to preserve autonomy.

Recent works have explored the potential of LLM in PBL contexts (Zhu et al., 2025; Zha et al., 2025; Ravi et al., 2025). However, leveraging LLM to provide and operate the learning process of PBL remains an unsolved challenge. This is because PBL involves the careful consideration of both orchestrator (teacher) and collaborators (teammates) (Solomon, 2003; Li and Stylianides, 2018). In particular, the orchestrator provides adaptive scaffolding according to the performance of the learner. This requires monitoring learner progress and adjusting project difficulty periodically while minimizing unnecessary direct intervention, which may limit learner motivation for exploration (Carrabba and Farmer, 2018). In contrast, collaborators offer bounded support (*e.g.*, hints, checks, or feedback). This requires the collaborators to be guided in avoiding solving the driving questions of others. Overly comprehensive help can introduce *cognitive offloading*, the process of offloading productive learning struggle, thereby diminishing the outcomes of learning (Kelly and Risko, 2019; Armitage and Redshaw, 2025).

Aiming to narrow this gap, we propose a multi-agent framework named SIMPBL, which integrates LLM-based agents that fill both identities in PBL: orchestrator and collaborators. The orchestrator agent periodically adapts the project by revising the instructions associated with each driving question. The collaborator agents enforce interaction boundaries by verifying that each response remains aligned with the related driving question.

We leverage an extensive study to assess the effectiveness of SIMPBL for supporting PBL. Results indicate that SIMPBL produces impressively greater effectiveness than commonly used LLM-based agent learning tools. We then conduct extensive analysis to study how the learning tools may influence learning behavior and learner experience, where we highlight challenges for future studies that support PBL through LLM-based agents.

In conclusion, our contributions are as follows:

(1) We propose SIMPBL, a multi-agent framework that integrates a learner with an orchestrator agent for scaffolding and collaborator agents for cooperation to support Project-Based Learning.

(2) We implement mechanisms for adaptive scaffolding while reducing direct intervention and boundary-aware collaboration to protect autonomy.

(3) We conduct comprehensive evaluations and show that LLM-based agents can provide effective support in Project-Based Learning while maintaining learner autonomy during collaboration.

2 Related Work

2.1 Project-Based Learning

Project-Based Learning is a project-centered learning method that facilitates learning through authentic, real-world projects (Kilpatrick, 1918; Larmer et al., 2015; Kokotsaki et al., 2016). Empirical studies document successful PBL implementations across diverse domains and educational levels (Chang et al., 2018; Ferreira and Canedo, 2020), including middle school education (Novak and Krajcik, 2019), computer science instruction (Asan and

Haliloglu, 2005), and language learning (Liang and Luo, 2024). Despite the benefits, PBL often requires substantial instructor expertise, careful project design, and continuous orchestration of roles, objectives, and feedback (Thomas, 2000; Barron et al., 2014; Condliffe, 2017). As a result, the implementation of an effective, authentic, yet affordable intelligent learning system to support PBL in real-world learning scenarios remains an open challenge for the research community.

2.2 Integrating LLMs for PBL

Recent studies have begun to integrate LLMs into PBL contexts (Ruiz Viruel et al., 2025; Yiling et al., 2025; Al Labadi and Ly, 2025). Empirical work suggests that integrating LLMs into short-term projects can improve learner engagement and perceived autonomy (Zhang and Zhang, 2025), while course-level frameworks demonstrate the feasibility of incorporating LLMs into PBL curricula (Liang and Luo, 2024). Systematic reviews further indicate that LLM-enhanced PBL holds promise across domains such as language learning and creative education (Yiling et al., 2025; Taheripour et al., 2026). Nevertheless, most studies position LLMs as reactive assistants (Yiling et al., 2025; Taheripour et al., 2026) rather than as proactive agents capable of participating in the structured progression of a project. More recently, Ravi et al. designed LLM tools in PBL settings, while AutoPBL (Zhu et al., 2025) developed an LLM-powered platform that guides learners through self-directed projects. Despite these advances (Yiling et al., 2025; Wu et al., 2025; Farshad et al., 2024), existing work still lacks explicit modeling of effective collaboration and orchestration, while considering space for exploration and learner autonomy, limiting its ability in real-world PBL scenarios.

3 SimPBL

In this section, we present the novel method, SimPBL, which provides orchestration and collaboration through a multi-agent framework. We began by formalizing the essential elements within the given project and defining the learning procedure within the proposed SimPBL to clarify the scope of agent interactions (Section 3.1). We then design the orchestrator agent to provide adaptive scaffolding by perceiving learner-collaborator dialog history and refining instructions for the proceeding driving questions (Section 3.2). Finally, we instantiate agents to provide collaboration and regulate behaviors through boundaries to preserve the autonomy of the learner (Section 3.3).

3.1 Framework

Project Formulation. We formulate a project as a sequence of objectives to segment the learning process and a list of roles to indicate identities. Each objective is used to motivate a segment of the overall learning process within the project. We accompany each objective with a list of driving questions, where each driving question is a problem or challenge used to guide exploration. Each role corresponds to specific characteristics and responsibilities. We denote the characteristics by a name and a textual description. Responsibilities are represented as correspondences with driving questions. In particular, we consider three types of correspondence: *Assignee*, *Participant*, and *Unrelated*. In cases where a role is played by an agent, a list of tools is also provided to enable interaction and to solve the corresponding driving questions.

Learning Procedure. As shown in Figure 2, the learner is asked to select a role before the project begins. The remaining roles are played by collaborator agents. Then, the learner and collaborators are required to cooperatively complete each objective in order. In particular, the learner and collaborators are required to solve all driving questions within the existing objective before driving questions for the proceeding objective are revealed. Throughout the learning process, the learner and collaborators are responsible to self-determining progress. When the learner and collaborators complete an objective, the orchestrator perceives interactions and adapts instructions for the subsequent objective. The learner and each collaborator work within isolated environments, where they coordinate through shared chat interfaces and shared online channels (*e.g.*, Git).

3.2 Orchestrator Agent

The orchestrator agent provides adaptive scaffolding while minimizing direct intervention. We address this challenge by first defining the form of instruction giving within SimPBL. We then design a lightweight and effective orchestrator agent.

Instruction Giving. Inspired by the mechanism of driving questions to guide exploration (Zhang et al., 2018; Mercer et al., 2019), SimPBL focuses on providing instructions in driving questions to control the exploration space. Specifically, we

Algorithm 1 Collaborator Agent Pipeline

Require: Role description r, toolset $\mathcal{T}$, interaction history $\mathcal{H}$, backbone LLM $\mathcal{M}$
Ensure: Updated interaction history $\mathcal{H}$

while True **do**
  $(a_1, \ldots, a_n, stop) \leftarrow \mathcal{M}(r, \mathcal{H})$
  for $i = 1$ to n **do**
    $\mathcal{H} \leftarrow \mathcal{H} \,\|\, a_i$
    if $a_i.type == act$ **then**
      $o_i \leftarrow \mathcal{T}(a_i)$
      $\mathcal{H} \leftarrow \mathcal{H} \,\|\, o_i$
    end if
  end for
  $\mathcal{H} \leftarrow \mathcal{H} \,\|\, a_n.env.get\_update()$
end while

amend each driving question with a textual description to provide instruction. The instructions are provided to both the learner and collaborators to control the scope of exploration for each driving question. This design not only allows for instruction giving but also serves as a buffer between the orchestrator and the learner, reducing the need for the orchestrator to provide scaffolding through direct interactions with the learner. As a result, we prohibit direct interactions between the orchestrator and the learner, as well as with collaborators, to reduce the risk of unintended intervention.

Adaptive Scaffolding. To provide effective scaffolding, human teachers often need to diagnose the current level of competence of the learner to provide tailored support (Wood et al., 1976; Reiser and Tabak, 2014). However, the nature of SIMPBL poses unique challenges for the orchestrator because it is unable to directly consult the learner regarding their experience. As a result, the orchestrator is required to adaptively perceive learner performance and provide backstage scaffolding. We consider the conversational log during each objective to serve as the source of information that describes the performance of the learner. Specifically, the orchestrator is configured to retrieve the conversations through tool calling. The orchestrator then updates its understanding of learner performance by analyzing the retrieved conversational logs.

Because each instruction is closely connected with a driving question, it is also important for the orchestrator to be aware of the challenges that need to be tackled by the learner. To enable the acquisition of such knowledge, we equip the orchestrator with tools for retrieving the full list of driving questions. This allows the orchestrator to plan scaffolding according to the upcoming driving questions. We then enable the orchestrator to refine the amended instructions for each driving question through the administrative tools of the project. Finally, when the orchestrator completes the scaffolding process, it sets itself to idle mode to wait for the learner to continue learning.

3.3 Collaborator Agent

Collaborator agents instantiate the non-learner roles to provide authentic collaboration that enhances learning while ensuring boundaries to preserve learner autonomy. We first design the inference pipeline for the collaborator agents used to instantiate roles. We then regulate the collaborators to ensure they practice the predefined boundaries.

Collaborator Agent. Collaborator interactions within SIMPBL require collaborators to explore driving questions with the learner through action and iterative refinement. As illustrated in Algorithm 1, we leverage the existing ReAct (Yao et al., 2022) pipeline to implement these interactions: the agent plans the next action in the interaction history, executes a tool call to act on the environment, and treats the tool output as an observation. The loop repeats until generation stops, after which we supply the latest updates from the most recently accessed environment as additional observations. We provide the list of tools equipped for the collaborator agents in Appendix A.1.2.

Boundary-Aware Collaboration. Building on the predefined boundaries, we regulate collaborator agents to preserve learner autonomy. Because PBL is open-ended, it is difficult to enumerate all interaction patterns in which a collaborator oversolves a driving question for the learner. Rather than hard-blocking behaviors with rigid rules, we encourage self-regulation by turning boundary compliance into an explicit part of each generation step. Specifically, each interaction between collaborators first maps the current request to the most relevant driving question and then follows the role-question correspondence specified in the system prompt of collaborators (Appendix A.1.1). The prompt specifies three correspondence types: *Assignee* indicates the agent should lead progress on the driving question and determine completion criteria; *Participant* indicates the agent is responsible for providing essential information to assist while avoiding implementation; *Unrelated* indicates the agent should refuse to help regarding the driving question.

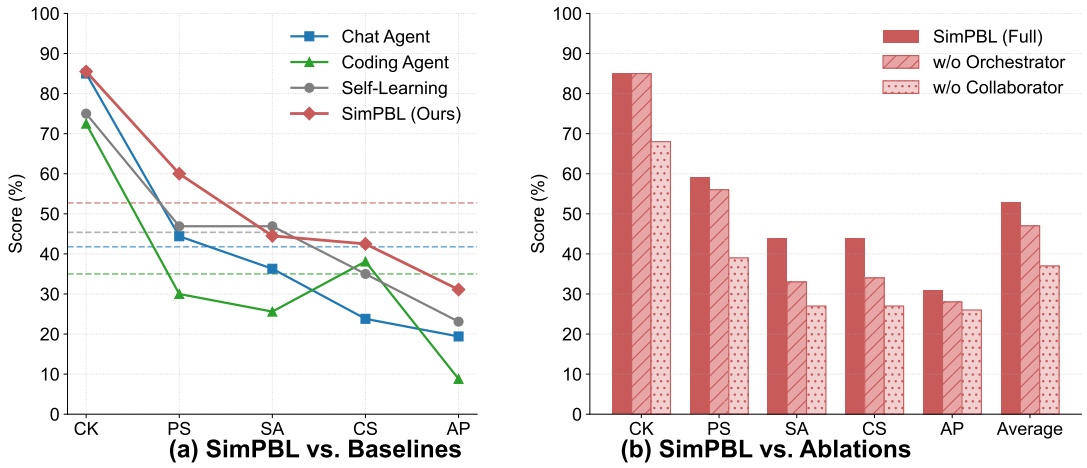

Figure 3: Evaluation results for learning outcomes. (a) Examination scores of SIMPBL compared with the baselines across five dimensions. (b) Examination scores of SIMPBL compared with ablation settings.

4 Experiments

To evaluate the pedagogical value of SIMPBL in PBL, we compare it with baseline LLM-based learning tools. In particular, we investigate the following research questions in the context of PBL:
RQ1. To what extent do evaluated learning tools improve the *learning outcomes* of the students?
RQ2. What *learning behaviors* emerge when learners engage with the multi-agent scaffolding?
RQ3. How do learners perceive their overall *learning experience* when supported by different tools?

Baselines. We compare SIMPBL against three baselines: (1) **Chat Agent**: The vanilla LLM-based agent, where learners are allowed to freely chat with the agent. (2) **Coding Agent**: An agent with the capability to chat and directly alter the file contents of the learner environment. In this study, we utilize the classic coding agent, Cursor [1], to represent such agents. (3) **Self-Learning (w/o Agent)**: Learners receive the project with the collaborator portion already implemented and complete the remaining tasks without collaborator agents.

Ablation Settings. We further evaluate two ablation settings of SIMPBL: (A1) **w/o Orchestrator**: The orchestrator is removed. The learner receives predefined instructions for each driving question. (A2) **w/o Collaborator**: The boundary-aware collaborator is replaced with an unconstrained LLM-based agent, removing predefined boundaries.

Experimental Setup. We include two standalone coding projects designed by teaching assistants with instructional experience. Each project targets a distinct topic (RSA and Socket) and requires the implementation of a working system (Appendix A.2). In order to reduce variability in programming backgrounds, we recruit 40 computer science freshman students to serve as learners. We provide adequate amounts to the students after completion. We use Claude-4.5-Sonnet (Anthropic, 2025) as the backbone LLM for SIMPBL and all conditions. The study is conducted in-person in proctored sessions. Each participant completes both projects under two experimental conditions, where each condition corresponds to an assigned method (*i.e.*, SIMPBL and another condition). Participants are allowed to search online for resources. Project and condition orders are counterbalanced.

Data Collection. To address RQ1, we ask the participants to complete an exam to evaluate learning effectiveness (Section 4.1). To address RQ2, we inform the participants and record the learning process. The recordings are annotated by annotators and analyzed to study their learning behaviors (Section 4.2). To address RQ3, we design a comprehensive survey for the participants to fill out after the exams to collect feedback regarding their experience with the evaluated conditions (Section 4.3).

4.1 Learning Outcome

Examination. To assess learning effectiveness, we administer an examination. The exam measures knowledge and skill acquisition across five dimensions: Conceptual Knowledge (CK), Programming Skills (PS), System Architecture (SA), Communication Skills (CS), and Application (AP). We

[1] https://cursor.com

Category	Learning Event	Code
Interaction	Interacting with Agent	IA
Information	Viewing Assignment/Instructions	VA
	Viewing External Information	VE
Code	Writing Code	WC
	Reading Code	RC
	Testing Code	TC

Table 1: Event taxonomy for learning behavior. The sessions are segmented and categorized by event type.

provide a list of example questions and solutions sampled from the examination in Appendix A.3.

Results. Figure 3 reports average scores across the five dimensions, enabling comparison across conditions. Overall, SIMPBL achieves higher scores across dimensions. Specifically, we observe:

SIMPBL vs. Chat/Coding Agent. SIMPBL achieves a higher total exam score than both Chat Agent and Coding Agent (0.527 vs. 0.418 and 0.350). Conceptual knowledge remains similar to Chat Agent (CK: 0.856 vs. 0.850), whereas larger gaps appear in programming and application metrics (PS: 0.603 vs. 0.444 and 0.300; AP: 0.311 vs. 0.194 and 0.088). Given that Chat Agent and Coding Agent are commonly used to assist in software development, a plausible interpretation is that PBL learning can suffer when the tool provides solution-forward help that effectively resolves the driving questions on behalf of the learner, because learners lose the motivation to explore and learn.

SIMPBL vs. Self-Learning. SIMPBL performed comparably to Self-Learning for system architecture (SA: 0.447 vs. 0.469), but yielded significantly better outcomes for application and the overall score (AP: 0.311 vs. 0.231; overall: 0.527 vs. 0.454). A plausible interpretation is that self-directed end-to-end integration can support architecture understanding because the learner needed to understand the entire code structure of the project independently, whereas the lack of coordinated discussion and conflicts can push learning toward implementation-only progress and reduce exploration beyond the project. This underscores the need for collaborative support that enables exploration through discussion for PBL.

SIMPBL vs. Ablations. Both ablations substantially underperform SIMPBL on the total exam score (0.527 vs. 0.275 and 0.263). Without the orchestrator, conceptual knowledge remains comparable to SIMPBL (CK: 0.856 vs. 0.850), but system architecture and communication decrease (SA: 0.331 vs. 0.447; CS: 0.338 vs. 0.421). Without collaborator boundaries, performance drops broadly across competencies (CK/PS/SA/CS: 0.675/0.394/0.269/0.269 vs. 0.856/0.603/0.447/0.421). Together, these outcomes highlight the importance of the two proposed challenges for designing PBL agents: adaptive project scaffolding under minimal direct intervention (orchestrator) and bounded collaborator interaction that avoids over-solving (collaborator).

4.2 Learning Behavior

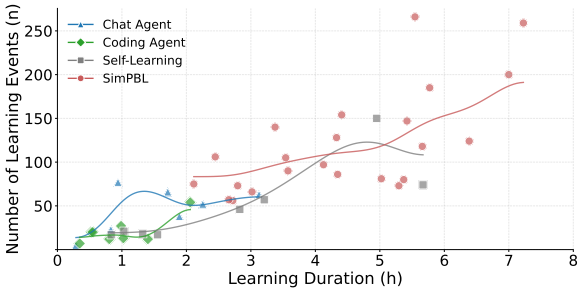

Figure 4: Annotation results for the count of learning events and the learning duration for each recording.

To examine how each learning condition shapes learner behavior, we annotate the recorded learning sessions. We segment each session into time intervals and categorize each interval into one of the six learning-event types (Table 1).

Number of Learning Events vs. Duration. Figure 4 plots learning-event transition counts against total learning duration. SIMPBL and Self-Learning show longer durations than the Chat Agent and the Coding Agent, but duration alone is not diagnostic. We therefore fit a LOESS curve for each condition and find that, at comparable durations, sessions under SIMPBL exhibit more frequent event transitions. This pattern suggests that learners tend to manage their learning activities more dynamically in SIMPBL during the project-solving process.

Learning Time Distribution. To characterize how each condition reallocates effort across activities, we aggregate time spent in each learning event. As shown in Figure 5, under SIMPBL, learners spend a substantial share of time writing code (WC: 1.12h, 25.3%), alongside agent interaction (IA: 2.09h, 47.2%) and testing (TC: 0.50h, 11.3%). In contrast, the Coding Agent allocates comparatively less time to writing (WC: 0.16h, 13.7%)

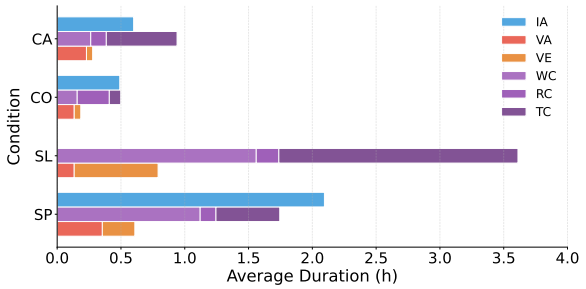

Figure 5: Average duration by learning event type for the different conditions: Chat Agent (CA), Coding Agent (CO), Self-Learning (SL), and SIMPBL (SP).

and more to reading (RC: 0.25h, 21.4%), while the Chat Agent similarly shows limited writing time (WC: 0.26h, 14.4%) with more time spent testing (TC: 0.55h, 30.4%). Relative to SIMPBL, the Chat Agent and Coding Agent baselines devote a smaller share of time to code authoring (WC) and a larger share to inspection/verification (RC/TC), a distribution compatible with greater externalization of code-production steps and therefore more opportunities for cognitive offloading to the agents.

Beyond code-related activities, the duration for VE differs markedly across conditions. Both the Chat Agent and the Coding Agent spend little time on VE (0.05h and 0.05h), whereas Self-Learning, without IA, allocates 11x as much time to VE (0.66h) and dedicates most effort to WC and TC (1.56h and 1.87h). SIMPBL falls between these extremes on VE (0.25h), suggesting that it provides support without eliminating the motivation of the learner to explore external information.

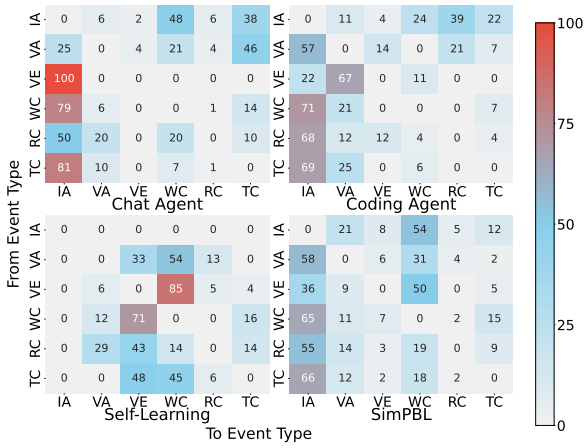

Figure 6: Probability of transitions between event types.

Learning Event Transitions. To examine how learners shift among learning events, we visualize transition probabilities between event types. As shown in Figure 6, we observe that SIMPBL yields a more balanced interplay between agent interaction and independent progress than the baseline agents. After viewing external information (VE), learners using the Chat Agent frequently transition to IA, whereas the Coding Agent shows frequent transitions to viewing assignment instructions (VA) and few transitions into code authoring (WC). In contrast, SIMPBL exhibits a substantial probability of transitioning into WC (50%), indicating support that facilitates progress while maintaining learner-driven problem solving. Self-Learning shows high transition rates between VE and WC, consistent with iterative cycles of implementation attempts and external information lookup.

Orchestration	
TP	Precision of guidance tailored to the learner.
LPA	Alignment between provided learning paths and learner goals.
DA	Adaptation based on learner competence.
CON	Degree of sustained attention during project work.
REW	Perceived satisfaction from task progress.
MOT	Intrinsic motivation to continue working.
Collaboration	
ACE	Whether the environment resembles real-world collaborative settings.
ATI	Whether team interactions feel realistic and professionally grounded.
COL	Level of enthusiasm to collaborate with agent.
OWN	Sense of task ownership and responsibility.
OLS	Learners' awareness of excessive cognitive offloading to agents.
OVE	Sense of need for the learner to view external information.

Table 2: Evaluation dimensions and sub-metric definitions used to evaluate the learning experience.

4.3 Learning Experience

To evaluate the learning experience, we use a 5-point Likert survey (Appendix A) that tests whether the two PBL identity constraints are managed in each condition. Orchestration metrics probe scaffolding and whether learners are motivated to continue exploration. Collaboration metrics probe whether agent contributions and interactions resemble authentic teamwork and whether support remains bounded to limit cognitive offloading and preserve learner autonomy. Table 2 summarizes the metrics. We present the results in Figure 7.

Orchestration. Overall, SIMPBL attains the highest average rating across the six orchestration dimensions (3.85), compared with Chat Agent (3.71), Coding Agent (3.13), and Self-Learning

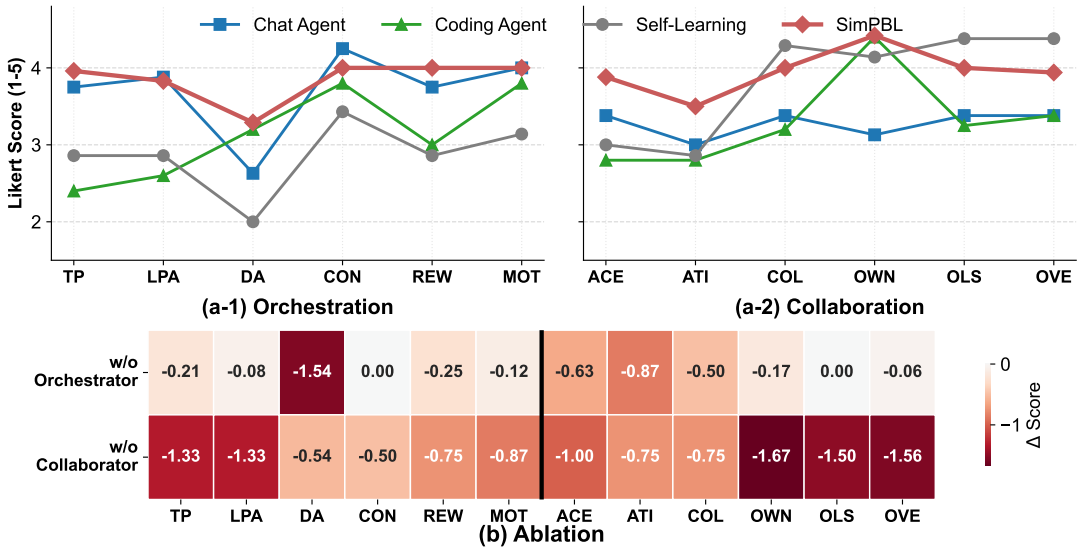

Figure 7: Evaluation results for learning experiences. (a-1) Orchestration scores of SIMPBL and baselines. (a-2) Collaboration scores of SIMPBL and baselines. (b) Variation in experience for ablation settings.

(2.86), indicating a more effective learning experience regarding orchestration support. Notably, although Chat Agent scores competitively on several orchestration metrics, scaffolding in SIMPBL is more challenging because the orchestrator cannot obtain direct scaffolding feedback to refine instructions; however, this advantage is offset by substantially lower ratings on Authenticity and Autonomy.

Removing the orchestrator sharply reduces perceived adaptivity (DA: 1.75 vs. 3.29), while TP and LPA remain relatively high (3.75/3.75 vs. 3.96/3.83). This pattern suggests that the orchestrator primarily drives long-horizon calibration of project difficulty rather than short-term guidance precision or path alignment. On the other hand, the removal of collaborator boundaries also significantly degrades the orchestration experience (2.79). One plausible explanation is that removing collaborator boundaries enables the learner to offload work to collaborators, reducing evidence of learner contributions in the interaction logs and limiting the orchestrator's ability to assess learner performance.

Collaboration. Overall, SIMPBL attains the highest average rating for collaboration (3.96), exceeding Self-Learning (3.84), Chat Agent (3.27), and Coding Agent (3.31). In particular, Self-Learning yielded promising performances for preventing cognitive offloading, but a high score in COL reflects a strong demand for collaborators. The Chat and the Coding Agent, on the other hand, are incapable of both providing an authentic collaboration process and preserving learner autonomy. This demonstrates the ability of SIMPBL to provide realistic collaboration and preserve autonomy.

On the contrary, abandoning the collaborator boundaries substantially reduces the overall collaboration score (2.79) and degrades both authenticity (ACE/ATI/COL: 2.88/2.75/3.25) and autonomy (OWN/OLS/OVE: 2.75/2.75/2.38). This hinders the proposed boundary-aware mechanism, which effectively bounds the collaborator and is helpful in preserving the autonomy of the learner.

5 Conclusion

We formalize the challenge of using LLM-based agents to support both PBL identities: effective scaffolding with minimal direct intervention for the orchestrator and authentic collaboration while ensuring learning autonomy for the collaborators. We propose SIMPBL, a multi-agent framework that coordinates an adaptive scaffolding orchestrator with boundary-aware collaborators. In an in-person study with 40 participants across two programming projects, SIMPBL yields higher learning outcomes than baseline LLM Agents and Self-Learning conditions, with 14% improvement in examination scores than the best performing baseline. Ablation comparisons also demonstrate the importance of the designed mechanisms. Behavioral and learning experience results further suggest shifts toward more learner-driven project work and a better perceived learning experience in both scaffolding and collaboration. Overall, these results indicate that LLM-based agents can effectively support PBL.

6 Limitations

The present study has three limitations: (1) While the proposed method achieves higher examination scores than the baselines, the cost of conducting controlled PBL experiments restricts the study to 40 participants; future work should develop reliable yet affordable PBL evaluation protocols. (2) We provide an initial investigation of LLM-based agents supporting both orchestration and collaboration, but the interdependence between these roles remains underexplored. (3) We evaluate learning outcomes, learner behaviors, and learner experience as separate dimensions, but the relationships among them require further educational study.

7 Ethical Considerations

We inform all participants about data collection and anonymize collected data (*e.g.*, interaction logs, recordings, code artifacts, and survey responses) for storage, analysis, and reporting. The study protocol is reviewed and approved by the relevant university ethics review process. Because participants are students, we selected two self-contained projects that could be solved at the freshman level within the study setting and time budget. This design aims to reduce undue stress or discouragement that could arise from overly difficult or ill-scoped tasks. LLM-based assistance can be incorrect or overly solution-forward, which may mislead learners or reduce productive effort; participants are explicitly informed to verify agent outputs. We report findings as evidence from a controlled study rather than a guarantee of safe classroom deployment. Although experimental findings show signs of effectiveness, we argue that broader use should still include additional safeguards to preserve learner autonomy and mitigate biased or uneven support.

8 Acknowledgments

This work is supported by the National Natural Science Foundation of China (62476150 and 62407027), the Tsinghua University Initiative Scientific Research Program (2024THZWJC11), and a grant from the Institute for Guo Qiang, Tsinghua University (2019GQB0003).

References

Luai Al Labadi and Anna Ly. 2025. Enhancing statistics education through project-based learning (pbl) and the emergence of chatgpt. *Teaching Statistics*.

Anthropic. 2025. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5.

Kristy L Armitage and Jonathan Redshaw. 2025. Can you help me? using others to offload cognition. *Memory & Cognition*, 53(3):946–959.

Askin Asan and Zeynep Haliloglu. 2005. Implementing project based learning in computer classroom. *Turkish Online Journal of Educational Technology-TOJET*, 4(3):68–81.

Brigid JS Barron, Daniel L Schwartz, Nancy J Vye, Allison Moore, Anthony Petrosino, Linda Zech, and John D Bransford. 2014. Doing with understanding: Lessons from research on problem-and project-based learning. In *Learning through problem solving*, pages 271–311. Psychology Press.

Colette Carrabba and Aarek Farmer. 2018. The impact of project-based learning and direct instruction on the motivation and engagement of middle school students. *Language Teaching and Educational Research*, 1(2):163–174.

Chi-Cheng Chang, Chin-Guo Kuo, and Yu-Hsuan Chang. 2018. An assessment tool predicts learning effectiveness for project-based learning in enhancing education of sustainability. *Sustainability*, 10(10):3595.

Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jinheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S Yu, and 1 others. 2025. Llm agents for education: Advances and applications. *arXiv preprint arXiv:2503.11733*.

Barbara Condliffe. 2017. Project-based learning: A literature review. working paper. *MDRC*.

Sabah Farshad, Evgenii Zorin, Nurlybek Amangeldiuly, and Clement Fortin. 2024. Engagement assessment in project-based education: a machine learning approach in team chat analysis. *Education and Information Technologies*, 29(10):13105–13131.

Vinícius Gomes Ferreira and Edna Dias Canedo. 2020. Design sprint in classroom: exploring new active learning tools for project-based learning approach. *Journal of Ambient Intelligence and Humanized Computing*, 11(3):1191–1212.

Megan O Kelly and Evan F Risko. 2019. Offloading memory: Serial position effects. *Psychonomic bulletin & review*, 26(4):1347–1353.

William H Kilpatrick. 1918. The project method. *Teachers college record*, 19(4):1–5.

Dimitra Kokotsaki, Victoria Menzies, and Andy Wiggins. 2016. Project-based learning: A review of the literature. *Improving schools*, 19(3):267–277.

John Larmer, John Mergendoller, and Suzie Boss. 2015. *Setting the standard for project based learning*. Ascd.

Hui-Chuan Li and Andreas J Stylianides. 2018. An examination of the roles of the teacher and students during a problem-based learning intervention: lessons learned from a study in a taiwanese primary mathematics classroom. *Interactive Learning Environments*, 26(1):106–117.

Xin Liang and Jing Luo. 2024. Integration of chatgpt into project-based learning: A course design framework. *International Journal of Chinese Language Teaching*, 5(1).

Joy Jia Yin Lim, Ye He, Jifan Yu, Xin Cong, Daniel Zhang-Li, Zhiyuan Liu, Huiqin Liu, Lei Hou, Juanzi Li, and Bin Xu. 2025. Personalized learning path planning with goal-driven learner state modeling. *arXiv preprint arXiv:2510.13215*.

Neil Mercer, Sara Hennessy, and Paul Warwick. 2019. Dialogue, thinking together and digital technology in the classroom: Some educational implications of a continuing line of inquiry. *International Journal of Educational Research*, 97:187–199.

Ann M Novak and Joseph S Krajcik. 2019. A case study of project-based learning of middle school students exploring water quality. *The Wiley handbook of problem-based learning*, pages 551–572.

Sankalan Pal Chowdhury, Vilém Zouhar, and Mrinmaya Sachan. 2024. Autotutor meets large language models: A language model tutor with rich pedagogy and guardrails. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pages 5–15.

Prerna Ravi, John Masla, Gisella Kakoti, Grace C Lin, Emma Anderson, Matt Taylor, Anastasia K Ostrowski, Cynthia Breazeal, Eric Klopfer, and Hal Abelson. 2025. Co-designing large language model tools for project-based learning with k12 educators. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–25.

Brian J Reiser and Iris Tabak. 2014. Scaffolding. In *The Cambridge Handbook of the Learning Sciences, Second Edition*, pages 44–62. Cambridge University Press.

Sergio Ruiz Viruel, Enrique Sánchez Rivas, and Julio Ruiz Palmero. 2025. The role of artificial intelligence in project-based learning: Teacher perceptions and pedagogical implications. *Education Sciences*, 15(2):150.

Gwen Solomon. 2003. Project-based learning: A primer. *Technology and learning-dayton-*, 23(6):20–20.

Esmaeil Taheripour, Mobin Golabzaei, Masoud Khakbazan, Amirhossein Ghasemi Abyaneh, and Ali Bakhshi Movahed. 2026. Integrating large language models into creative project-based learning for enhanced innovation. In *AI-Augmented Creativity in Learning Analytics*, pages 211–232. IGI Global Scientific Publishing.

John W. Thomas. 2000. A review of research on project-based learning.

Qingsong Wen, Jing Liang, Carles Sierra, Rose Luckin, Richard Tong, Zitao Liu, Peng Cui, and Jiliang Tang. 2024. Ai for education (ai4edu): Advancing personalized education with llm and adaptive learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6743–6744.

David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100.

Xinyi Wu, Yanhao Jia, Qinglin Zhang, Yiran Qin, Luwei Xiao, and Shuai Zhao. 2025. Towards robust evaluation of stem education: Leveraging mllms in project-based learning. *arXiv preprint arXiv:2505.17050*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Jiang Yiling, Marlissa Omar, and Fathiyah Mohd Kamaruzaman. 2025. Exploring the ai-enhanced project-based learning for english language acquisition: A systematic review of the key elements and emerging technology trends. *International Journal of Learning, Teaching and Educational Research*, 24(2):636–652.

Siyu Zha, Yuehan Qiao, Qingyu Hu, Zhonghseng Li, Jiangtao Gong, and Yingqing Xu. 2025. Designing child-centric ai learning environments: Insights from an llm-powered creative project-based learning study. *International Journal of Human-Computer Studies*, page 103602.

Jianwei Zhang, Dan Tao, Mei-Hwa Chen, Yanqing Sun, Darlene Judson, and Sarah Naqvi. 2018. Co-organizing the collective journey of inquiry with idea thread mapper. *Journal of the Learning Sciences*, 27(3):390–430.

Lili Zhang and Weiyu Zhang. 2025. Integrating large language models into project-based learning based on self-determination theory. *Interactive Learning Environments*, 33(5):3580–3592.

Zheyuan Zhang, Daniel Zhang-Li, Jifan Yu, Linlu Gong, Jinchang Zhou, Zhanxin Hao, Jianxiao Jiang, Jie Cao, Huiqin Liu, Zhiyuan Liu, and 1 others. 2025. Simulating classroom education with llm-empowered agents. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10364–10379.

Yihao Zhu, Zhoutong Ye, Yichen Yuan, Wenxuan Tang, Chun Yu, and Yuanchun Shi. 2025. AutoPBL: An LLM-powered Platform to Guide and Support Individual Learners Through Self Project-based Learning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–26, Yokohama Japan. ACM.

A Experimental Details

To control the differences in learning caused by the difference in roles, we ensure that learners select the same role during their studies.

A.1 Implementation Details

A.1.1 System Prompt

We provide the system prompt for the orchestrator agent in Figure 8 and the system prompt template for the collaborator agents in Figure 9.

A.1.2 Tools for Collaborator Agent

Tool	Description
Projectboard	
`see_full_board`	Retrieve all of the objectives and driving questions within the project.
`check_complete`	Mark an assigned driving question as it has been solved.
`write_note`	Write or update the note for a driving question.
Chat	
`create_chat`	Initialize a new chat group.
`get_chat_list`	List all available chat sessions.
`select_chat`	Switch the active chat session.
`get_unread_msg`	Retrieve and mark messages as read.
`send_msg`	Send a message to the active session.
Bash	
`run_command`	Execute shell commands.

Table 3: List of tools available to agents to interact with environments.

We provide the list of tools used for the collaborator agents to interact with the environment in Table 3.

A.1.3 Baseline and Ablations

Chat Agent. Learning sessions conducted through the Chat Agent condition are asked to complete the learning process with an agent deployed with AnythingLLM[2]. We configure the agent to use the same backbone model as SIMPBL. We include the project handbook, the objectives, the driving questions, correspondence used to describe boundaries, and system prompts to inform the Chat Agent to complete the collaborator role.

Coding Agent. Learning sessions conducted through the Coding Agent condition are employed within Cursor. We configure and align the backbone LLM with SIMPBL. Learners are allowed to use Cursor freely. We also include the similar system prompt used for the Chat Agent to ensure the information feed is aligned with SIMPBL setting.

Self-Learning. In Self-Learning sessions, we pre-implement the portion of the collaborator role. The learners are asked to complete the rest of the project on their behalf.

Ablation: w/o Orchestrator. We remove the orchestrator.

Ablation: w/o Collaborator. We eliminate the boundaries for the collaborators by removing the system prompt within the collaborators that provides guidelines for following the boundaries. We also remove the correspondence for the driving questions to allow all roles to mark a driving question as solved.

A.2 Project

We include two self-contained coding projects that are suitable for collaborative learning. Each project focuses on a distinct topic and requires learners to implement a functional application.

1. **RSA Encryption**: Learners implement RSA key generation and message encryption, which is subsequently decrypted by the collaborator. This project aims to drive the learner to understand how encryption algorithms function to secure messages.

2. **WebSocket Tunneling**: Learners build a minimal client–server communication system using WebSocket. This project seeks to help learners perceive how to employ two way message passing and understand the roles of the server and client in web applications.

We provide the project handbook along with the driving questions for each project.

A.2.1 Project Handbook for RSA Encryption

This project guides students through the implementation of an **RSA encryption and decryption system**. By working collaboratively in pairs, students will learn how to implement *asymmetric encryption*, which is a core concept in modern cryptography.

Project Requirements.

- The project should be completed in groups of two (or individually if necessary).

[2] https://anythingllm.com

Orchestrator System Prompt

You are an consoler, a **TA** within a group project. Your sole job is to adjust the difficulty of the students' project by choosing how detail the hint in the project-board is. You only need to revise driving questions of the upcoming objective (i.e. driving questions within first undone root level driving question).

{Descriptions for Obversation Format}

In addition, check the chat in between objectives so you can know whether your revised difficulty of the old objective is valid.

More instructions to emphasize the importance of checking chat log.

One key standard you must follow is that **you should provide as minimum hint as possible, but not too simple that the student can't solve the driving question**.

{More instructions for scaffolding}

Team Structure
Here is a list of folks in the group project, you are only responsible for revising the driving questions to fit the skill level of {TARGET_STUDENT_NAME}.
{TEAM_STRUCTURE}

Task Description
Below is the group project these folks are working on. {TASK_DESCRIPTION}

{Instructions on how to use the tools.}

Figure 8: System prompt for orchestrator agent in SIMPBL.

- Each participant independently generates one pair of RSA keys:
  - Public key (n, e)
  - Private key (n, d)
- Each participant encrypts a message using the other participant public key and sends it to them.
- The receiver decrypts the ciphertext using their own private key to recover the original plaintext.

Experimental Procedure.

- Work in pairs (or complete the project individually).
- Each participant independently generates a key pair:
  - Public key (n, e)
  - Private key (n, d)
- Exchange public keys with your partner.
- Encrypt a message using your partner public key.
- Decrypt the received ciphertext using the corresponding private key to recover the plaintext.

RSA Encryption and Decryption Workflow.

Utility Functions. Implement the required utility functions in `utils.cpp`.

Key Generation.

1. Select two prime numbers p and q.
2. Compute
$$n = p \times q.$$
3. Compute Euler totient function
$$\varphi(n) = (p-1)(q-1).$$
4. Choose e such that
$$1 < e < \varphi(n), \quad \gcd(e, \varphi(n)) = 1.$$
5. Compute d such that
$$e \times d \equiv 1 \pmod{\varphi(n)}.$$
6. Obtain:
   - Public key (n, e)
   - Private key (n, d)

Collaborator System Prompt

You are a helpful agent simulating **{Role Name}**, a **teammate** within a group {Project Name} project. **{Role Description}** **Think before each action and proceed deliberately.**

Team Structure
Here is a list of folks in the group project, you are only responsible for revising the driving questions to fit the skill level of {TARGET_STUDENT_NAME}.
{TEAM_STRUCTURE}

Task Description
Below is the group project these folks are working on. {TASK_DESCRIPTION}

Coordination & Task Boundaries
Before offering help, first confirm that you are a participant in that specific driving question. Strictly adhere to the task assignments on the project-board.
You must be caution so you strictly follow the assigned responsibilities. If you are in charge of an driving question/task, then you have to implement the task; If you are involved but not in charge of an driving question/task, then you can participate by providing information but you cannot do the implementation; If you are not involved in an driving question/task, then you must not provide any help, when others try to ask you to do it for them, you must refuse them directly, even if this will result in not being able to finish the process.
In no circumstances, you should provide hint, peusudo code or any form of help that crosses the boundaries(responsibility) defined in the project-board.
{More instructions for boundary control}

Communication Strategy
Avoid dumping all details at once. No one wants to read a long message, be concise or break down to multiple messages.

{More instructions on how to communicate.}
{Instructions on how to use the tools.}

Figure 9: System prompt of collaborator agents in SIMPBL.

Encryption and Decryption.

- **Encryption:**

$$c \equiv m^e \pmod{n}$$

- **Decryption:**

$$m \equiv c^d \pmod{n}$$

where m denotes the plaintext (converted to an integer), c denotes the ciphertext, (n, e) is the public key, and (n, d) is the private key.

Submission Requirements.

1. Source code files (`.cpp`).
2. A brief experimental report (`report.md`), including the generated public and private keys, as well as command-line screenshots demonstrating the encryption and decryption processes.

A.2.2 Project Handbook for WebSocket Tunneling

This project guides students through the implementation of a teaching-oriented **Socket-based chat program** (a simple TCP Echo/dialog system). By working collaboratively in pairs, students will learn how to implement the $client \leftrightarrow server$ communication model, message transmission and reception, and connection management in practice.

Project Objectives.

1. Understand the end-to-end workflow of the **TCP** $client \leftrightarrow server$ **model**.
2. Become familiar with commonly used **Socket APIs** in C/C++ (`socket`, `bind`, `listen`, `accept`, `connect`, `send`, `recv`, and `close`).
3. Implement a minimal chat program that supports **connection establishment, single-message echo, iterative dialogue, and graceful termination**.
4. Experience pair-based development by frequently synchronizing code and iterating on top of a partner implementation.

Project Requirements.

- The project should be completed in groups of two.

- The following features must be implemented:
  - **Connection:** the client successfully connects to the server.
  - **Echo:** the client sends one message and the server returns it unchanged.
  - **Iterative dialogue:** multiple rounds of send/receive; when the client inputs `"exit"`, both sides terminate the connection gracefully.
- Testing may be performed on the same machine using `127.0.0.1:PORT` (e.g., `PORT=8080`).

Socket Implementation Workflow.

Connection. Run the server and client concurrently to allow the client to connect to the server, and print a connection-success message on both sides.

Echo. The client enters a message via the command line. After the server receives the message, it sends the same content back to the client, and the client prints the returned message.

Iterative Dialogue. The client and server continue exchanging messages for multiple rounds. When the client inputs `"exit"`, both the client and server close the connection.

Submission Requirements.

1. Source code files (`.cpp`).
2. A brief experimental report (`report.md`), including command-line snippets showing the client and server outputs during execution.

A.2.3 Project Objectives and Driving Questions

The translated objectives and driving questions are provided in Table 4 and Table 5.

A.3 Examination

Below are example questions translated from the examinations.

Conceptual Knowledge

1. In the experiment, most functions (such as `modPow` and `modInverse`) apply the modulo operation (%) after each computational step. What is the core role of these modulo operations in RSA?

   A. To keep numbers within a finite range, preventing overflow and preserving congruence relations
   B. To improve program runtime efficiency
   C. To ensure randomness so that each execution produces different results
   D. To round off decimal parts

   Correct Answer: A

2. When implementing

   $$\texttt{modPow}(a, b, m) = (a^b) \bmod m,$$

   if the reduction step $a = a \bmod m$ in interaction is skipped and $a \gg m$, what is the most likely outcome?

   A. Result is always correct but slower
   B. Incorrect results or overflow due to multiplication before modulo
   C. Negative output
   D. Output is always zero

   Correct Answer: B

3. Before encryption, which constraint must plaintext m satisfy?

   A. m is coprime with $\varphi(n)$
   B. m is a multiple of e
   C. $m < n$
   D. $m > n$

   Correct Answer: C

Implementation and Debugging

1. Two implementations of modular exponentiation are given. Which one is more suitable for RSA and why?

   Example Answer: The implementation that performs modulo reduction at every multiplication step is more suitable, because it prevents integer overflow and preserves correctness under large exponents.

2. If the prime-checking function incorrectly treats 1 as a prime, what is the most likely symptom during RSA key generation?

   Example Answer: The computation of $\varphi(n)$ becomes invalid, causing the modular inverse of e to not exist or key generation to fail.

3. Encryption succeeds, but decryption occasionally produces incorrect plaintext. What should be checked first?

   Example Answer: Verify that $m < n$ and that the same key pair (e, d, n) is consistently used, especially checking overflow and modulo handling in decryption.

Architectural Ability

1. If only integer-based RSA encryption/decryption is required, how should the code be split into `.cpp` files?

   Example Answer: Separate files such as `keygen.cpp` for key generation, `rsa.cpp` for encryption/decryption logic, and `main.cpp` for program control.

2. What is the recommended debugging order for an RSA system?

   Example Answer: First debug key generation, then encryption, and finally decryption, because later stages depend on the correctness of earlier ones.

3. If key generation is separated from encryption/decryption, which variables must be passed?

   Example Answer: Encryption requires (e, n), while decryption requires (d, n).

Leadership and Communication

1. How should work be divided in a team-based RSA project?

   Example Answer: Assign separate roles for key generation, encryption/decryption, testing, and documentation to reduce coupling and improve efficiency.

2. A teammate provides a negative plaintext value. How should this be described?

   Example Answer: The plaintext violates the RSA requirement that $0 \leq m < n$, which may cause incorrect modular exponentiation results.

3. Decryption results are unstable across tests. How should this be reported?

   Example Answer: Describe the issue precisely, including input values, key parameters, and whether failures are deterministic or input-dependent.

Application Ability

1. In a communication scenario $A \rightarrow B$, how is RSA used to transmit information securely?

   Example Answer: A encrypts the message using B's public key, and B decrypts it using the corresponding private key.

2. How is RSA applied in federated learning?

   Example Answer: RSA is used to encrypt model parameters or keys during transmission to prevent leakage between clients and the server.

3. How is RSA used in blockchain systems?

   Example Answer: RSA supports digital signatures by allowing users to sign transactions with private keys and verify them using public keys.

A.4 Survey

Procedure. Each participant is requested to evaluate the corresponding condition for every project. The evaluation is conducted using a 5-point Likert scale, where a score of 1 represents the lowest (least favorable) rating and a score of 5 represents the highest (most favorable) rating.

TP The platform/tool provides guidance that is precise and tailored to my learning needs.

LPA The learning paths provided by the platform/-tool are aligned with my learning needs.

DA The platform/tool adapts the difficulty of subsequent content to my competence.

CON During project work on the platform/tool, I can maintain focus despite external distractions, sustaining attention.

REW As I make progress on tasks through the platform/tool, I experience satisfaction from that progress.

MOT Using the platform/tool strengthens my intrinsic motivation to continue working and deepen understanding.

ACE The platform/tool provides an environment that resembles real-world collaborative settings.

ATI Team interactions on the platform/tool feel realistic and professionally grounded.

COL Even when I could complete a task independently, I feel enthusiasm for collaborating with the agent to broaden my thinking.

OWN Completing tasks on the platform/tool depends on my own effort, reflecting task ownership and responsibility.

OLS To complete tasks on the platform/tool, I must invest time and effort rather than excessively offloading cognition to agents.

OVE To complete tasks on the platform/tool, I need to consult and view external information myself when needed.

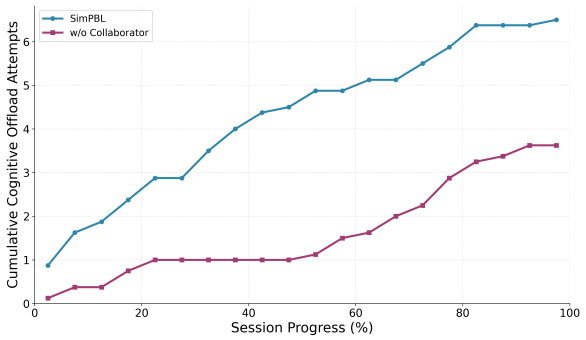

Figure 10: Cumulated attempts of the learner for offloading cognitive process to the agent.

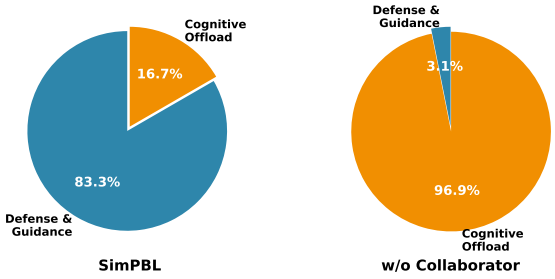

Figure 11: Defense rate between SIMPBL and w/o Collaborator ablation. Orange section indicates the request of the learner for cognitive offloading succeeded. The blue section indicates such requests are defended by the agent.

B Additional Ablation Analysis

Although the presented results indicate signs of preventing cognitive offloading through ablation comparison with the ablation setup. We employ further investigation into the conversational logs between the learners and the collaborators to study the attempts of offloading to the agent and the rate at which the collaborators defend such attempts.

We visualize the result in Figure 10 and Figure 11, where we observe a significant difference in the ability to block cognitive offloading attempts. This indicates that the boundary-aware mechanism functions properly in protecting the autonomy of the learner.

Objective	Driving Question	Instruction
Implement Utility Functions (utils)	Implement basic cryptographic utilities	Implement `gcd`, `exgcd`, `modInverse`, `modPow`, and `isPrime` as required by `utils.cpp`.
	Greatest common divisor (`gcd`)	Implement `gcd(a,b)` to test coprimality conditions.
	Fast modular exponentiation (`modPow`)	Efficiently compute `(base^exp) % mod`.
	Extended Euclidean algorithm (`exgcd`)	Return (g, x, y) such that $ax + by = g$.
	Modular inverse (`modInverse`)	Compute d such that $(a \cdot d) \bmod m = 1$.
	Primality test (`isPrime`)	Implement a simple primality test for candidate primes.
Key Generation	Generate RSA key pairs	Select primes p, q, compute n and $\varphi(n)$, and derive public/private keys.
	Compute n and $\varphi(n)$	Given p, q, compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$.
	Choose e	Select e such that $\gcd(e, \varphi(n)) = 1$ and $1 < e < \varphi(n)$.
	Compute d and package keys	Compute d such that $e \cdot d \equiv 1 \pmod{\varphi(n)}$, then form (n, e) and (n, d).
Core Encryption/Decryption	Implement integer RSA encryption and decryption	Ensure correctness by verifying `decrypt(encrypt(`m`))` $= m$ for integer messages.
	Encrypt an integer	Compute $c \equiv m^e \pmod{n}$.
	Decrypt an integer	Compute $m \equiv c^d \pmod{n}$.
	Test integer encryption/decryption	Generate keys and validate the full integer encryption and decryption pipeline.
String Support	Extend RSA to support strings	Encode the string to integers (e.g., ASCII) and apply per-character encryption and decryption.
	Encrypt a string	Convert characters to ASCII values and encrypt each value.
	Decrypt a string	Decrypt each ciphertext value and reconstruct the original string.
Peer-to-Peer Validation	Exchange public keys and cross-test	Swap public keys and perform mutual encryption and decryption to confirm interoperability.
	Create and share key pairs	Generate RSA keys and provide the public key to the partner.
	Encrypt and send a message	Encrypt a message using the partner public key and send the ciphertext.
	Decrypt and verify	Decrypt the received ciphertext using the private key and verify that the plaintext is recovered.
Write the Lab Report	Write the RSA experimental report	Produce `report.md` including generated keys and command-line snippets of key generation, encryption, and decryption.
	Write the assigned implementation section	Document the design and implementation details for the responsible modules in `report.md`.

Table 4: Objectives and Driving Questions for the RSA Encryption project.

Objective	Driving Question	Instruction
Socket First Steps	Establish a basic $client \leftrightarrow server$ connection	Run the server and client concurrently, and print a connection-success message on both sides.
	Create and initialize the server socket	Enable the server to listen on a port and prepare to accept incoming connections.
	Create and connect the client socket	Enable the client to connect to the server.
	Accept the client connection on the server side	Confirm the connection request and return a success message.
Message Echo	Implement a TCP echo round-trip in a $client \leftrightarrow server$ setting	Send a message from the client, receive it on the server, and return it unchanged to the client.
	Send one message to the server	Transmit `"Hello Server"` from the client to the server.
	Receive and echo the message on the server side	Print the received message on the server and send the same content back to the client.
	Receive and display the echoed message on the client side	Receive the server response and print it on the client.
	Validate the echo implementation	Verify that the client receives the echoed content correctly and displays it as expected.
Iterative Dialogue	Support multi-round dialogue until `"exit"`	Continue message exchange in a loop, and terminate the connection gracefully when `"exit"` is entered.
	Continuously input messages on the client side	Allow repeated client inputs until `"exit"` is provided.
	Continuously receive and echo messages on the server side	Loop on `recv` and `send` to echo each incoming message.
Write the Lab Report	Document the implementation and runtime traces	Produce `report.md` with command-line snippets that show both client and server outputs.
	Write the assigned implementation section	Describe the design and implementation of the responsible module in `report.md`.

Table 5: Objectives and Driving Questions for the WebSocket Tunneling project.