RMIT University

Bachelor of Information Technology – BP162

OENG118X/COSC250X Engineering & IT Capstone Project

Capstone Project Final Report

SocratesCode: AI-Powered Socratic Tutor for Programming

Student team:	Ca Chep Team
Student name & ID:	Vo Minh Thien An s3916570
	Hua Van Anh Khoa s3883254
	Bui Minh Khoi s3929015
	Luong Dinh Khang s3930216
	Hoang Duc Anh s3847506
Academic supervisor:	Dr. Phong Ngo phong.ngo@rmit.edu.vn
	Dr. Hoang Van hoang.van3@rmit.edu.vn
Company:	SSET
Industry supervisor:	None

Contents

1 Executive summary

Students are increasingly depending on AI tools like ChatGPT and GitHub Copilot for direct code solutions, which is a serious issue for modern programming education as it prevents them from practicing critical thinking and problem-solving skills. Traditional AI assistants will provide the answer directly rather than guiding the learner to arrive at it, creating a true dependency that inhibits the development of practical skills.

This capstone developed an AI-powered Socratic tutor specifically for programming education at RMIT's School of Science, Engineering and Technology (SSET). SocratesCode encourages its learners to solve problems on their own by using the Socratic questioning technique, as contrast to conventional AI tools that only provide straightforward answers. The AutoGen framework serves as the foundation for the system's sophisticated multi-agent AI architecture, which includes specialized agents for evaluation, validation, tutoring, and greeting.

Key technologies in the system include: Mem0 integration supports long-term memory and personalized learning trajectories; multilayered guardrails prevent manipulative behavior and ensure educational content integrity; and the RAG pipeline ensures that responses are supported by the carefully chosen educational content. RMIT's AWS RACE platform was used to launch the system, guaranteeing its institutional security and scalability.

With a 92.7% user satisfaction rating and 82.1% of participants indicating interest in using the platform again, the results far exceeded the project's goals. Although it never provided direct code solutions, the system was still educationally beneficial and met all of the requirements of the Socratic process. With GPT-4o-mini scoring a median of 81 points, performance study validated line-for-line response quality, demonstrating its dependability across generic programming scenarios.

The project's practical approach to AI-dependency challenges in programming education is what makes it so important. SocratesCode demonstrates how educational technology may enhance rather than replace the learning experience by fusing cutting-edge AI orchestration with tried-and-true pedagogical concepts. Thus, the multi-agent architecture offers a scalable foundation for the development of pedagogically sound AI systems that preserve the educational integrity of the learning process while enabling personalized learning.

This paper offers a framework for the appropriate integration of AI into higher education settings where learning objectives take priority over user comfort, making a theoretical contribution to educational technology practice and AI system design.

2 Project background & problem statement

2.1 Project Overview

SocratesCode is using an intelligent AI-based conversational instructor to address evolving issues in programming education at RMIT's SSET. Typical programming instruction often comes in the form of direct answers via forums, documentation, or occasionally AI technologies like ChatGPT. This encourages excessive reliance and hinders the development of critical thinking skills. While AI systems just provide them with the answer without guiding them through the reasoning process, students usually discover that they need help with the problem-solving methodology.

The objective is to create a conversation-based, AI-powered Socratic instructor that helps SSET students improve their understanding of coding and problem-solving techniques. SocratesCode employs Socratic questioning techniques in natural language interactions to help SSET students find the solution on their own rather than handing it to them, in contrast to traditional AI programs that give direct solutions.

Using large language models (LLMs) such as ChatGPT or comparable technologies, the system is intended to function as an intelligent chatbot that is enhanced with Socratic reasoning skills. By asking questions, offering contextual clues, and decomposing complex programming challenges into digestible chunks, the AI tutor will participate in conversational exchanges while maintaining an engaging discussion.

A reasoning evaluator to guarantee pedagogically sound responses, adaptive questioning algorithms that react to student inputs, and real-time conversational exchanges are some of the key characteristics. The breakthrough is in using multi-agent AI technology to change the standard AI code support from providing answers to asking questions, resulting in a more instructive and long-lasting learning strategy.

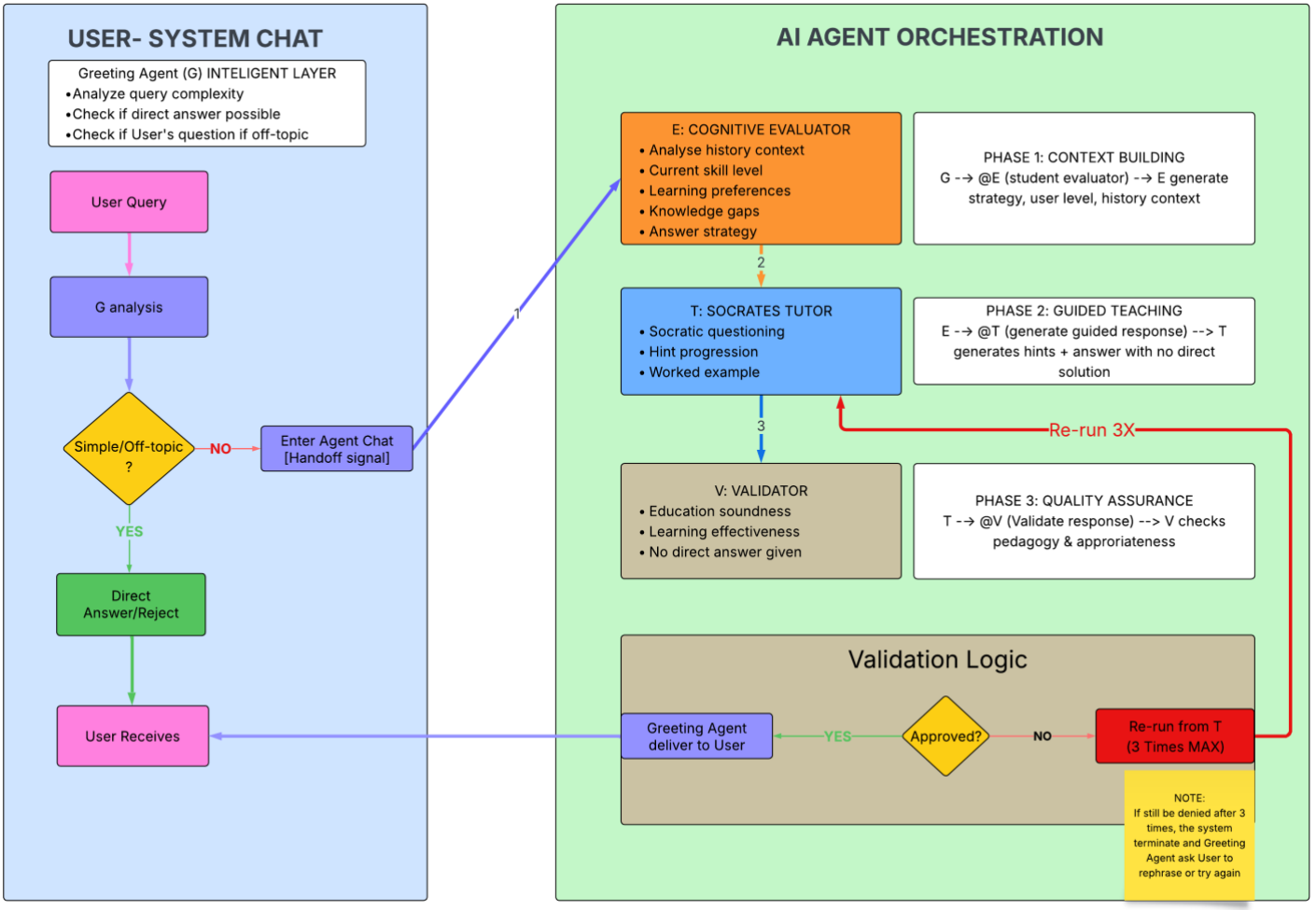


Figure 1: Agent Flow Diagram

2.2 Problem Statement

As more students enroll in RMIT SSET programming majors [1], the unexpected educational disadvantages increase, despite institutional efforts [2]. Students are faced with uneven levels of proficiency, little one-to-one support, and low engagement in self-paced learning. University constraints include staff shortages and resources for maintaining and supporting a high level of teaching at scale [3].

Even though students can now access AI-powered tools like ChatGPT and GitHub Copilot more easily, they frequently offer direct code solutions that skip the learning process. Instead of learning basic problem-solving techniques, students grow reliant on these technologies for quick fixes. This leads to a paradox where students fail to develop the critical thinking abilities required for independent programming yet sophisticated AI support is easily accessible.

Office hours and peer mentorship programs are meant to provide individual support, but they are underutilized due to schedule conflicts, a reluctance to ask questions [4], or the inability to get help quickly. Furthermore, current AI tools lack of pedagogical frameworks; they are code generators rather than instructional manuals, and they do not modify their responses to suit the needs of individual students or make use of tried-and-true teaching strategies.

Lack of essential programming and problem-solving abilities stems from the disconnect between meaningful educational support and powerful AI code assistance, especially for students who are new to programming. An AI-powered conversational system that combines the responsiveness and accessibility of contemporary LLMs with successful teaching strategies is obviously needed to promote real learning and skill improvement.

3 Literature review & market research

3.1 Theoretical Foundations in Educational AI

Programming education has undergone significant transformation with the integration of artificial intelligence, yet many implementations fail to address fundamental pedagogical principles. The challenge lies not in AI's technical capabilities, but in creating systems that genuinely support learning rather than enabling academic shortcuts. This tension becomes particularly evident when examining how current AI platforms handle educational guardrails and student guidance.

Sweller's Cognitive Load Theory remains highly relevant to contemporary AI-assisted learning environments. His work demonstrates that effective learning occurs when we carefully manage the three types of cognitive load: intrinsic (the complexity of the material itself), extraneous (poorly designed instruction), and germane (the mental effort devoted to processing and understanding) [5]. For programming education specifically, this theory suggests that AI tutors should decompose complex problems systematically while avoiding overwhelming students with irrelevant information.

Recent research by Morrison and colleagues has shown compelling evidence for the effectiveness of worked examples in programming contexts [6]. Their studies reveal that students who study complete solutions before attempting similar problems perform significantly better than those who jump directly into problem-solving. However, the challenge for AI systems lies in knowing when to provide complete examples versus partial scaffolding - a nuance that requires sophisticated understanding of individual student progress.

The integration of Socratic questioning with AI presents both opportunities and substantial challenges. Traditional Socratic pedagogy depends heavily on human intuition to guide questioning sequences, adapting in real-time to student responses and emotional cues. Current AI implementations often struggle with this adaptive quality, frequently reverting to scripted question sequences that may feel mechanical to students.

3.2 General-Purpose AI Platforms: The Double-Edged Sword

ChatGPT has become the default AI assistant for many students, but its design philosophy fundamentally conflicts with educational objectives. The system optimizes for user satisfaction and perceived helpfulness, which often means providing direct solutions when students express frustration or time pressure. While ChatGPT can generate accurate code and explanations, it lacks any meaningful educational framework to guide students toward independent discovery.

Perhaps more concerning is how easily students can manipulate ChatGPT's responses through strategic prompting. Students quickly learn to frame requests in ways that bypass basic educational considerations - asking for "examples" rather than solutions or claiming urgent deadlines to justify receiving complete answers. This manipulation represents a fundamental flaw in single-layer guardrail approaches.

Claude offers similar capabilities with enhanced reasoning and longer context windows yet suffers from the same core issues. Both platforms excel at technical tasks but lack the specialized architecture needed for sustained educational interaction. They cannot maintain pedagogical consistency across extended conversations or adapt their teaching approach based on demonstrated student competency.

3.3 Educational AI Platforms: Promising but Incomplete

Khan Academy's Khanmigo represents the first serious attempt to create an AI tutor with educational safeguards built into its core design. The platform implements prompt-level restrictions and incorporates basic Socratic questioning techniques. However, field testing has revealed significant vulnerabilities in its guardrail implementation. Students have discovered various prompting strategies that circumvent these restrictions, and the single-model architecture lacks the redundancy needed to maintain educational integrity across diverse interaction patterns.

Khanmigo's approach to personalization also remains limited. While it can adjust question difficulty based on student responses, it doesn't implement comprehensive cognitive load management or sophisticated skill assessment. The system tends to default to explanation-heavy responses rather than guiding students through discovery processes.

TutorOcean AI takes a different approach by combining AI tutoring with human oversight, allowing escalation to human tutors when AI limitations become apparent. This hybrid model addresses some educational effectiveness concerns but comes with obvious scalability limitations. More importantly, the AI components still function primarily as sophisticated answer generators rather than discovery-learning facilitators. The system lacks specialized programming education methodologies and doesn't implement cognitive load theory principles in its interaction design.

RMIT's VAL GenAI demonstrates how institutional AI can provide valuable academic support while maintaining some educational safeguards. The system includes university-specific information and emphasizes academic integrity in its design. However, VAL GenAI functions more as an academic assistant than a specialized programming tutor. While it can help with research and general coursework support, it lacks the focused pedagogical frameworks necessary for effective programming education.

3.4 Critical Gaps in Current Approaches

The most significant limitation across existing platforms is their vulnerability to student manipulation. Single-layer guardrail approaches, whether implemented through prompt engineering or basic content filtering, prove inadequate when faced with determined students seeking shortcuts. Research in adversarial prompting demonstrates that clever users can consistently bypass these restrictions, undermining the educational objectives these systems claim to support.

Current platforms also struggle with maintaining consistent pedagogical approaches across extended interactions. A student might receive excellent Socratic guidance for one problem, then be given direct solutions for similar problems later in the same session. This inconsistency stems from the lack of specialized architectural components dedicated to educational oversight and validation.

Perhaps most importantly, existing AI educational tools tend to create solution dependency rather than fostering independent problem-solving capabilities. Students quickly learn to rely on AI-generated solutions rather than developing their own analytical and debugging skills. This dependency issue represents a fundamental challenge in AI education - the very convenience that makes these tools attractive can undermine their educational value.

3.5 SocratesCode's Distinctive Approach

SocratesCode addresses these fundamental limitations through several architectural innovations that distinguish it from existing platforms. Rather than relying on single-model implementations with prompt-level guardrails, the system employs a multi-agent architecture where specialized agents handle different aspects of educational interaction.

The validation agent specifically focuses on ensuring educational integrity, examining proposed responses for pedagogical appropriateness before they reach students. This creates multiple layers of educational oversight that prove much more resistant to manipulation than single-layer approaches. The personalization agent continuously assesses student proficiency and adjusts interaction patterns, accordingly, implementing dynamic cognitive load management based on demonstrated competency.

Most significantly, SocratesCode's tutor agent focuses explicitly on discovery-oriented learning rather than explanation provision. The system guides students through structured problem-solving processes using worked examples, incomplete examples requiring student completion, and error correction scenarios tailored to individual skill levels. This approach directly addresses the solution dependency issues that plague other platforms.

The multi-agent coordination enables consistent application of educational principles across diverse programming contexts while maintaining response quality and educational effectiveness. Unlike platforms that might provide excellent guidance for one type of problem while offering shortcuts for another, SocratesCode's architecture ensures pedagogical consistency through specialized agent oversight.

3.6 Market Position and Future Implications

SocratesCode occupies a unique position in the educational AI landscape by prioritizing educational effectiveness over user convenience. While existing platforms either focus on technical capability without educational consideration (ChatGPT, Claude) or implement basic educational features without comprehensive pedagogical integration (Khanmigo, CodeHelp), SocratesCode provides a specialized solution designed specifically around established learning theory.

The platform's approach to guardrail implementation through multi-agent validation represents a significant advancement in educational AI security. Rather than relying on easily circumvented prompt engineering, the system creates multiple layers of educational oversight that maintain pedagogical integrity even under adversarial conditions.

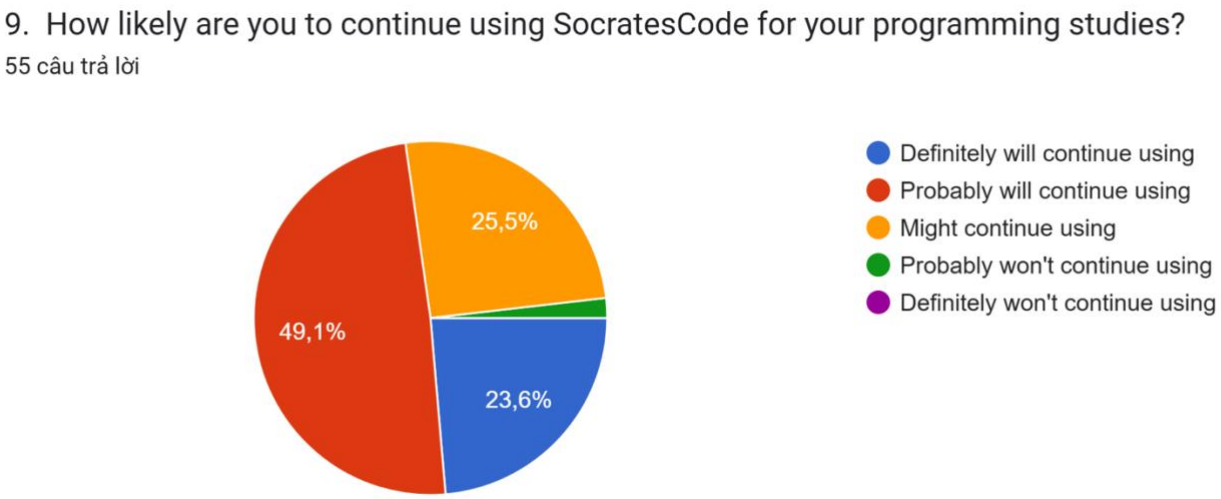


Figure 2: User's survey "How likely are you to continue using SocratesCode for your programming studies"

The pie chart presents user feedback on their likelihood of continuing to use SocratesCode for programming studies, based on 55 responses. A clear majority—72.7%—selected "Definitely" or "Probably will continue," indicating strong satisfaction with the platform. Interestingly, despite SocratesCode not providing direct answers, users still perceive it as highly effective, likely due to its Socratic approach that encourages critical thinking and problem-solving. However, 25.5% chose "Definitely won't continue," suggesting there's room to improve the user experience or better communicate the system's pedagogical value. Overall, the chart reflects both the strengths and opportunities for refinement in SocratesCode's design.

This architectural approach provides a blueprint for developing pedagogically-sound AI systems that maintain educational objectives while delivering the personalized, scalable learning experiences that make AI-powered education attractive. As educational institutions grapple with integrating AI tools responsibly, SocratesCode's multi-agent framework offers a practical model for balancing technological capability with educational integrity.

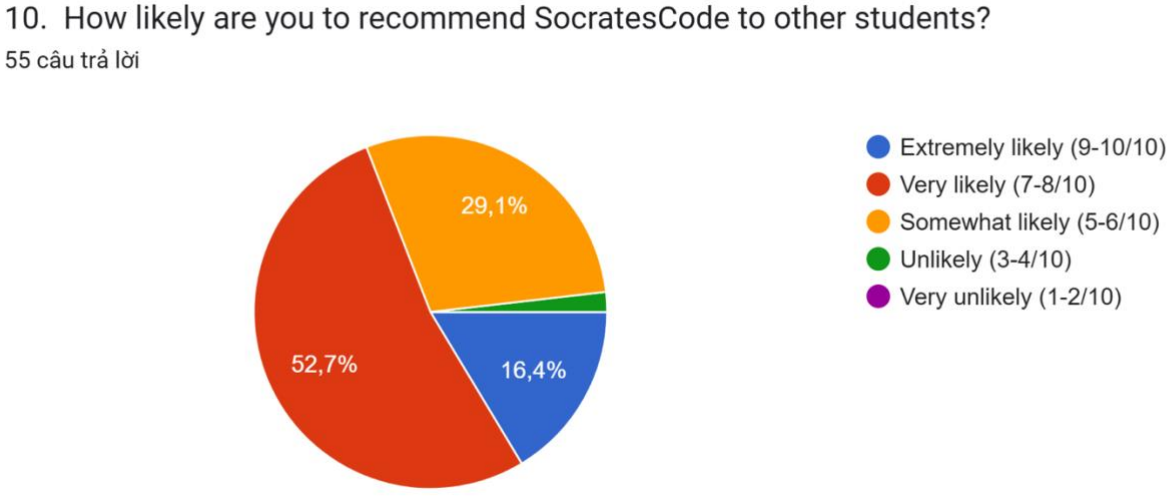


Figure 3: User's survey "How likely are you recommend SocratesCode to other students"

The chart shows that most students have a positive perception of SocratesCode, with 52.7% rating it “very likely” and 16.4% “extremely likely” to recommend it. Another 29.1% are “somewhat likely,” meaning overall, nearly everyone finds value in the platform. This suggests that SocratesCode has a strong impact on student satisfaction, with only a very small minority expressing doubt.

3.7 Applications in Controlled Learning Environments

SocratesCode's guardrail architecture makes it suitable for use in sanitized educational environments where academic integrity is critical [7]. Unlike general AI tools that can be easily manipulated, the multi-agent validation system ensures consistent educational behavior even in high-stakes situations.

In classroom settings, teachers can use SocratesCode during live coding sessions without worrying that students will get direct answers [8]. The system maintains its Socratic questioning approach regardless of how students phrase their requests. This makes it valuable for in-class activities where students work on problems while getting AI assistance.

For open-book examinations, SocratesCode offers a unique advantage [9]. Traditional AI tools are banned from exams because they provide solutions, but SocratesCode could actually be allowed since it only guides thinking rather than giving answers. Students would still need to demonstrate their own problem-solving skills while having access to educational guidance.

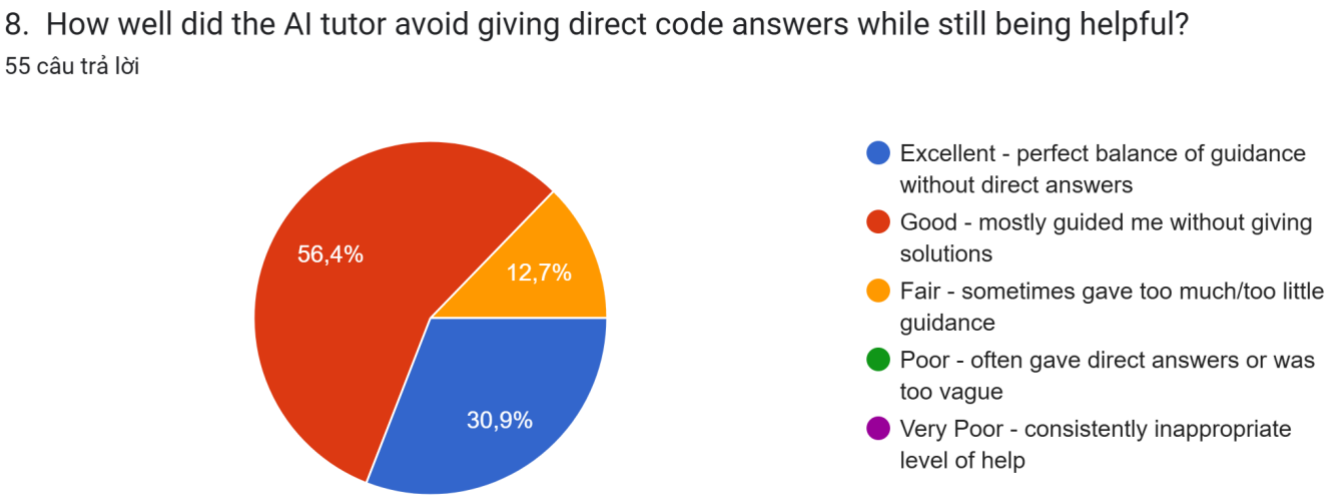


Figure 4: User’s survey “How well did the AI tutor avoid giving direct code answers while still being helpful”

This chart shows that the AI tutor in SocratesCode is generally effective at providing guidance without directly giving code answers. A strong **56.4% rated it “Good”** and **30.9% rated it “Excellent,”** meaning over **87% of students** felt the tutor maintained a helpful balance. Only **12.7%** thought the guidance was inconsistent, and no one rated it poor or very poor. This suggests the AI tutor is highly effective in encouraging independent problem-solving while still offering meaningful support.

The system could also work well in controlled lab environments where students practice programming under supervision [10]. Teachers can allow AI assistance without compromising learning objectives, since the system won't provide shortcuts that undermine skill development.

4 Solution design

4.1 Solution Design: Front/Back End

4.1.1 Technical overview

SocratesCode is a powerful learning application that utilizes AI and is built to support programmers in knowledge review and program solving. The tool applies unique AI technology and operates systematically to assist with step-by-step guidance, which distinguishes it from other general AI assistants by focusing exclusively on programming education and custom problem-solving activities. The success of the project is evaluated in terms of the following key performance indicators: Log-in success route to user engagement, Accuracy and value of AI response to programming queries, Retention of user through knowledge preservation in accordance with the notebook feature, Stability of platform and scaling through lean backend architecture, Responsiveness of Front-end Design for optimal user experience. Features of the Core Application: Conversation Management: AI-powered chat interface for programming queries purposefully designed for programming education, Message System: user-to-user or feedback mechanism for real-time communication, Notebook capabilities: Knowledge preservation to store your content for future reference, User Preferences: Interface characteristics that allow user input for coding experience. Components of the User Interface: Responsive and adaptive design through layout.tsx and page.tsx components. Global styling and formatting through globals.css for visual identity. Component location for modularity and maintenance through the components folder.

4.1.2 Main Features and User Engagement

This platform will cater to three main types of users: Novice Programmers: Wanting step-by-step instructions and explanations of basic concepts. Intermediate Developers: Wanting to review specific technologies or work through complex problems. Advanced Programmers: Needing quick reference and complex problem-solving help. AI-based Programming Assistant: Integration with the ChatGPT 4o API to enable cutting-edge natural language processing (NLP) functionality. Programming-specific prompt engineering to generate concise and accurate, yet relevant, responses. Step-by-step explanation approach, designed for assisting with reading and understanding code. User Management System: Secure registration and authentication processes. Profile management with a record of skill level. Personalized learning paths based on user selections. Knowledge Preservation: Integrated notebook system for saving AI responses that are important. Indexed knowledge base for fast reference. Export functionality for offline review. Interactive Learning Platform: Real-time chat interface for coding conversations. Syntax highlighting and formatting options. Feedback mechanism to ensure usage for improvement.

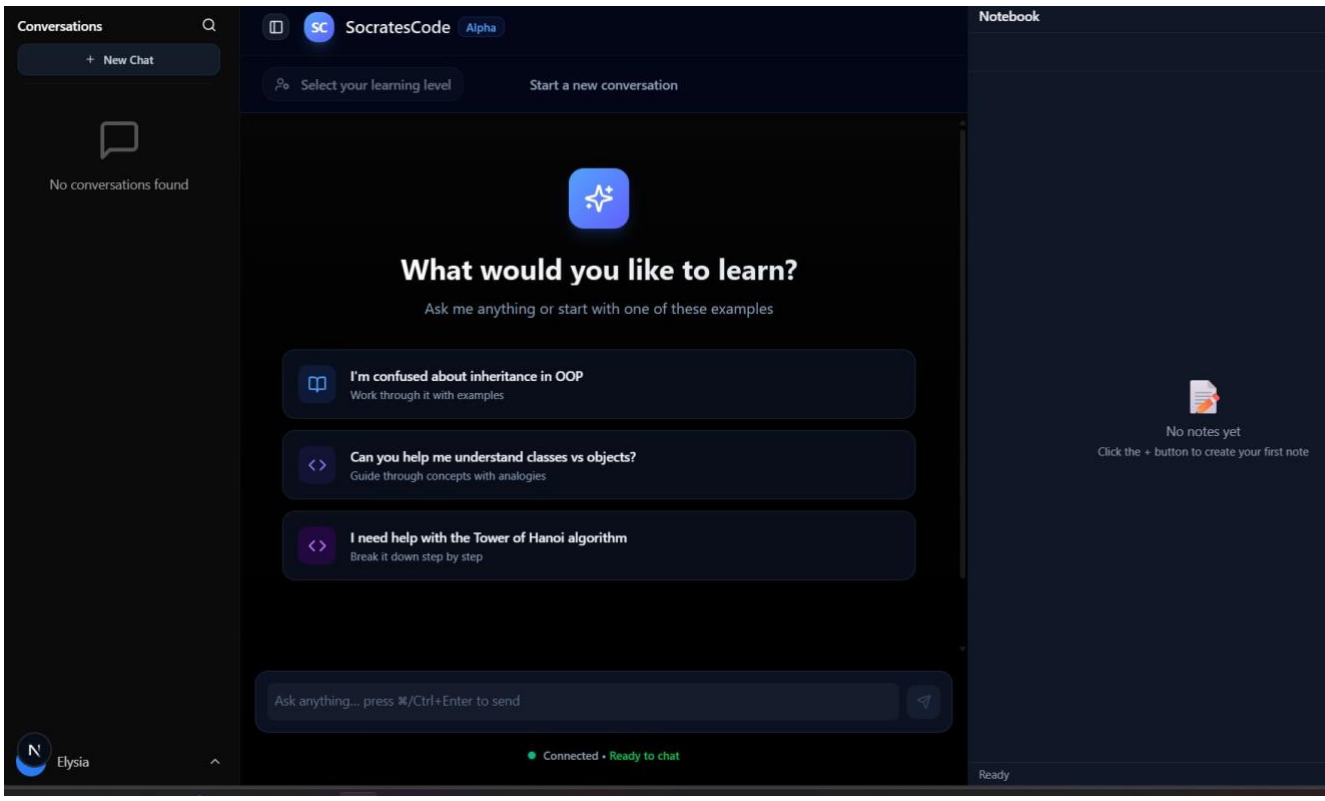


Figure 5: Chat interface with skill setting, section management and notebook

4.1.3 Technology Stack and Considerations

Backend Technologies:

- Runtime Environment: Utilizing Node.js for scalable operations on the server
- Database: Using Supabase (PostgreSQL) for effective data management
- Storage: Using Minio for easy storage of files and media
- Authentication: Creating custom middleware with session capabilities
- API Integration: Implementing RESTful services and ChatGPT 4.0 API

Frontend Technologies:

- Framework: Using Next.js with TypeScript for typing-heavy development
- Styling: Utilizing CSS modules with "global" styling to be responsive
- State Management: Using react hooks to manage state
- Component Architecture: Creating modular, reusable components

Security and Performance Considerations:

- Secure user sessions via middleware-based authentication
- Implement caching mechanics to speed up response times
- Validate and sanitize user input for security
- Sustain a scalable architecture to support concurrent users

4.1.4 Development Progress and Achievements

The implementation has made significant strides within all major components of the system:

- Full Backend Support: All backend services and routing have been implemented
- User Interface Support: All user-facing components are designed and in-play
- AI Support: The application is connected with ChatGPT 4.0 API with programming specifications
- User Management: The app has fully operational user authentication and profile management

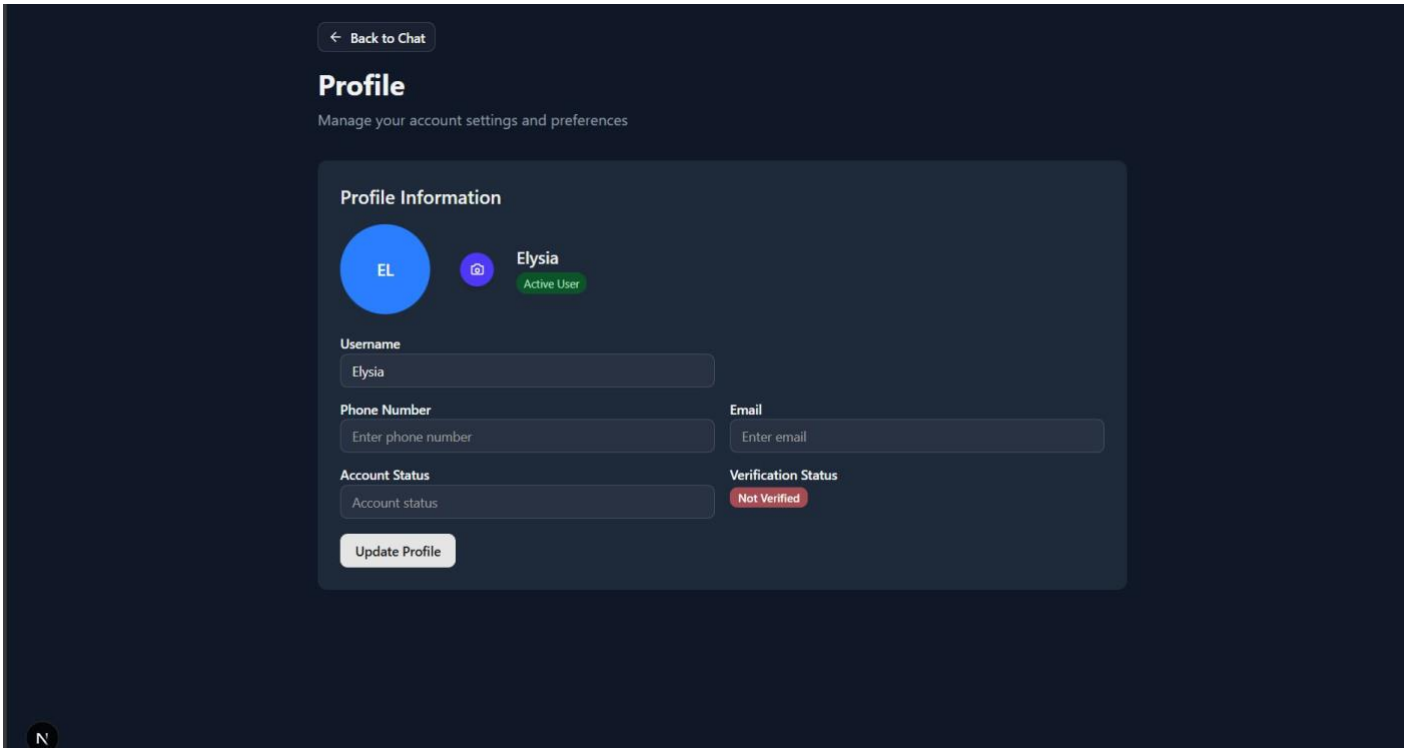


Figure 6: User profile management and authentication

Feature Set: The implementation of conversation management, notebook features, and user preferences are complete This robust toolbox of capabilities sub-positions SocratesCode as a custom-built, robust programming education platform that creates distinction between traditional AI assistants through consideration of the development context and unique construct.

4.2 Solution Design: Cloud Hosting

The successful implementation of our multi-agent tutoring system on RMIT's RACE (Research and Academic Cloud Environment) platform, utilizing Amazon Web Services (AWS) infrastructure, was a significant accomplishment for the last phase of our SocratesCode capstone project. This cloud integration made it possible to demonstrate our AI-powered Socratic tutor in a real-world setting with reliability, affordability, and complete compliance with RMIT's institutional requirements.

4.2.1 Collaboration and Support

Our academic supervisors and Mr. Patrick, an RMIT AWS Cloud Application Specialist (RMIT Melbourne), provided invaluable advice and technical assistance that allowed us to reach this milestone. We learned a lot about AWS hosting best practices and RMIT's RACE cloud environment thanks to Patrick's help. His guidance enabled us to set up safe, scalable hosting for our application and successfully configure a RACE Virtual Machine (VM).

We set up a public HTTPS port (443), which allowed for safe external access to our system, with Patrick's assistance. By enabling us to host our website openly for the next demonstration day, we made sure that peers, stakeholders, and academic staff could all access the SocratesCode platform without encountering any technical difficulties.

4.2.2 Technical Implementation

Several crucial steps were involved in the deployment:

- RACE VM Provisioning: We were able to install and manage our application stack with greater freedom after requesting and receiving a dedicated RACE VM.
- AWS Integration: By utilizing AWS services via RACE's backend infrastructure, the virtual machine (VM) provided us with AWS's scalability and dependability while upholding RMIT's security and governance guidelines.
- Secure Public Access: We guaranteed encrypted traffic and adherence to contemporary online security standards by establishing public port 443 (HTTPS) and setting up SSL/TLS certificates.
- Continuous Monitoring: We were able to verify performance goals like 99.5% uptime and less than 20-second AI response times by stress testing the system under realistic load using RACE's built-in monitoring capabilities in conjunction with our own Langfuse analytics.

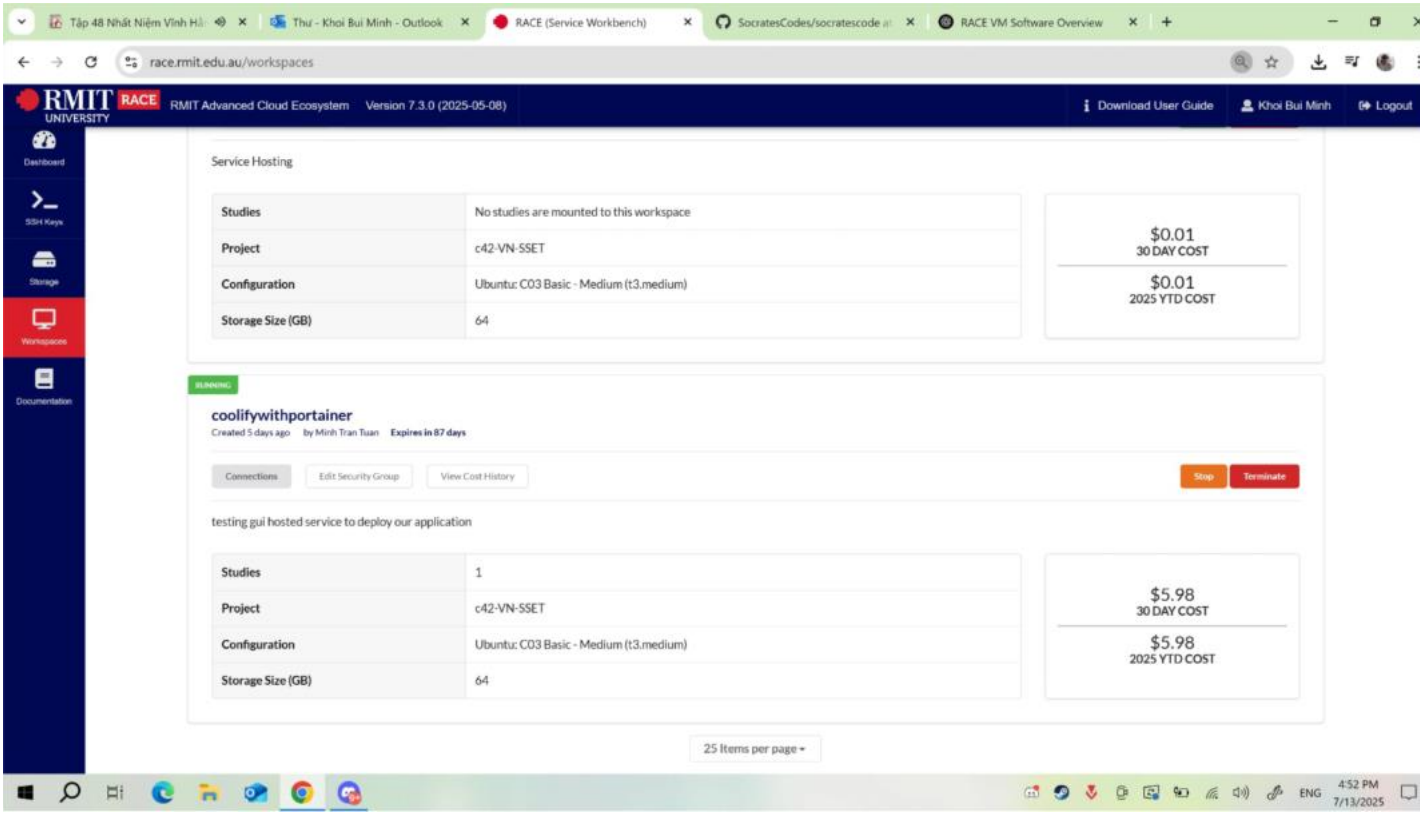


Figure 7: RMIT RACE dashboard

4.3 Solution Design: AI Technology

The application of sophisticated AI orchestration techniques, including Mem0 memory management, a Retrieval-Augmented Generation (RAG) pipeline, and a multi-agent AI architecture utilizing AutoGen, is a key technical accomplishment of the SocratesCode project. These elements serve as the foundation of our Socratic teaching system, allowing for large-scale, context-aware, and adaptable learning exchanges.

4.3.1 Mem0: Long-Term Memory for Tailored Education

- The system can now adapt questions and feedback to each student's changing ability level thanks to the integration of the Mem0 framework, which provides long-term, persistent recollection of user interactions.
- Context Preservation: Mem0 keeps track of important learning milestones and conversation histories so that the AI agents can refer to earlier sessions and keep the tutoring session going.

- Personalized Learning Paths: The method adapts the difficulty of Socratic questions and offers worked examples based on each learner's needs by recalling past errors and successes.
- Performance Monitoring: The team was able to gauge how memory persistence enhanced student engagement and concept retention thanks to Mem0's analytics.
- The platform was changed from a basic chatbot that could only respond to questions to a totally adaptive learning assistant thanks to its permanent memory feature.

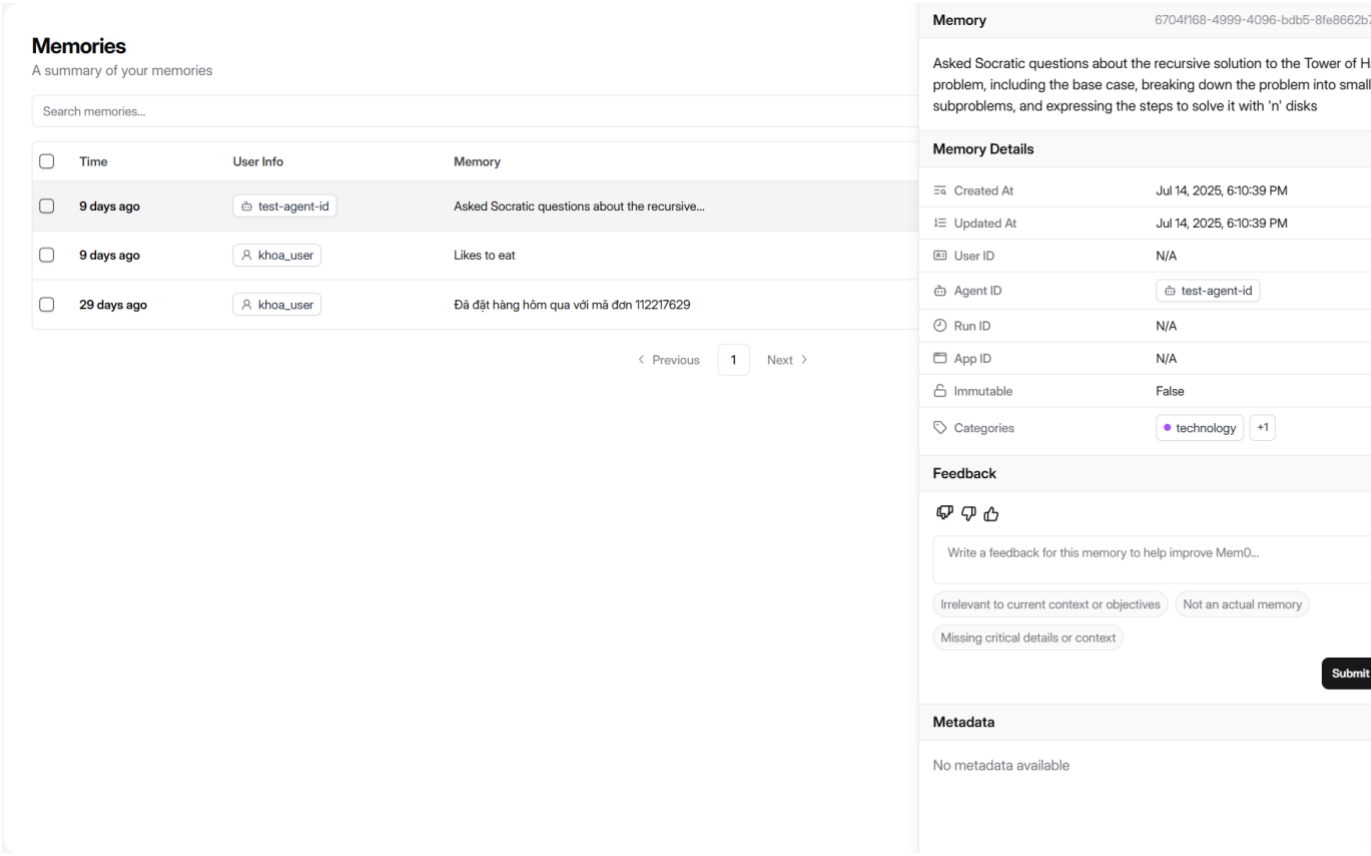


Figure 8: mem0 memory building

4.3.2 AutoGen-Powered Multi-Agent AI Architecture

- The core component of SocratesCode is a multi-agent AI system that coordinates several specialized agents utilizing the AutoGen framework.
- Specialized Agents: Each of the architecture's agents—the Tutor, Validation, Greeting, and Evaluation agents—oversees a specific function, such as asking Socratic questions, verifying accuracy, or adjusting difficulty.
- Agent Collaboration: AutoGen facilitates these agents' real-time communication, enabling them to discuss, verify, and improve responses prior to displaying them to the user.
- Robustness & Guardrails: The system's well-planned architecture makes sure that direct responses are avoided and instead leads students through the steps of reasoning while upholding academic integrity.
- High dependability and flexibility are offered by this modular, agent-driven architecture, which makes it possible to scale in the future or incorporate new teaching techniques.

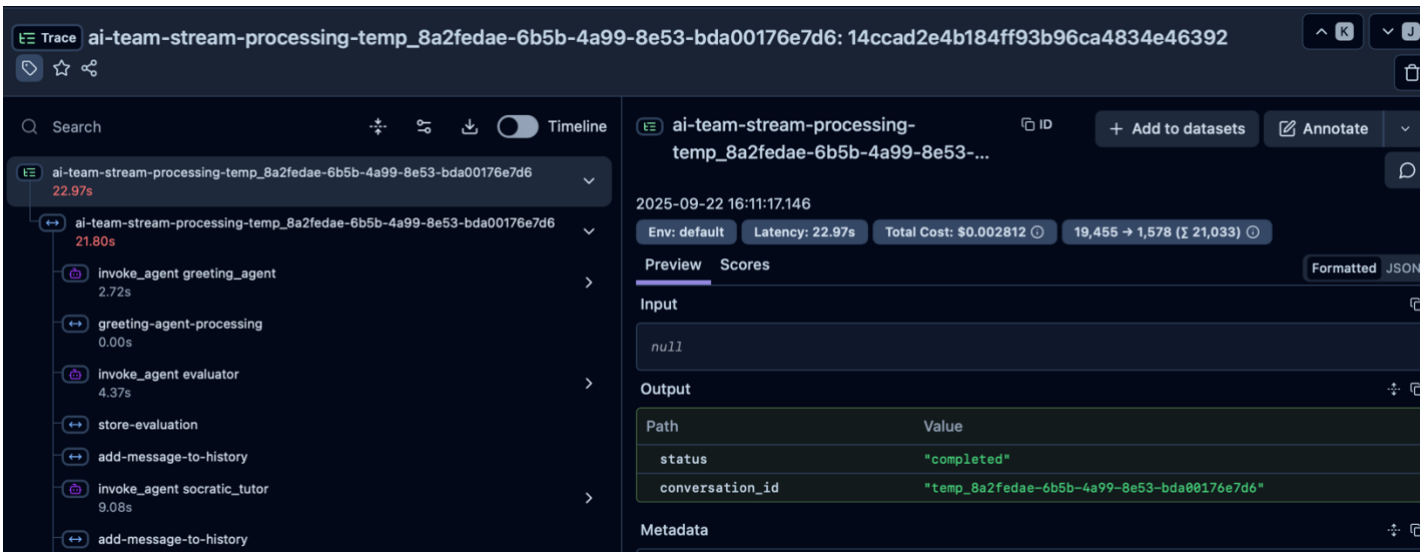


Figure 9: AI team procesing

4.3.3 RAG stands for retrieval-augmented generation.

- The team used a RAG pipeline to further improve the AI's educational quality.
- Dynamic Knowledge Retrieval: RAG links the AI system to carefully chosen course materials and programming resources, allowing it to instantly retrieve pertinent information.
- Grounded Responses: The tutor generates replies that are accurate, well-sourced, and innovative by feeding retrieved materials into the context window of the generative model.
- Decreased Hallucinations: The AI provides accurate explanations that are in line with RMIT's curriculum thanks to the combination of retrieval and generation.
- Thus, RAG fills the gap between open-ended generative AI and the dependability needed for scholarly applications.

5 Result analyses & discussions

5.1 Project Success and Deliverables Achievement

Undoubtedly, the project has achieved its main goals in producing the AI-powered Socratic tutor, which continues to offer great promise for enhancing programming instruction. All of the deliverables outlined in the original project proposal have been completed, and the system's apparent effectiveness and user approval are confirmed by the positive survey responses from 67 participants.

The survey's findings indicate that there is an amazing opportunity for market share. Following a quick introduction to the platform, 82.1% of the participants say they would be interested in learning more about SocratesCode. With only 17.9% expressing disinterest, this high degree of interest points to significant market feasibility and supports the project's fundamental principle of filling gaps in programming education assistance.

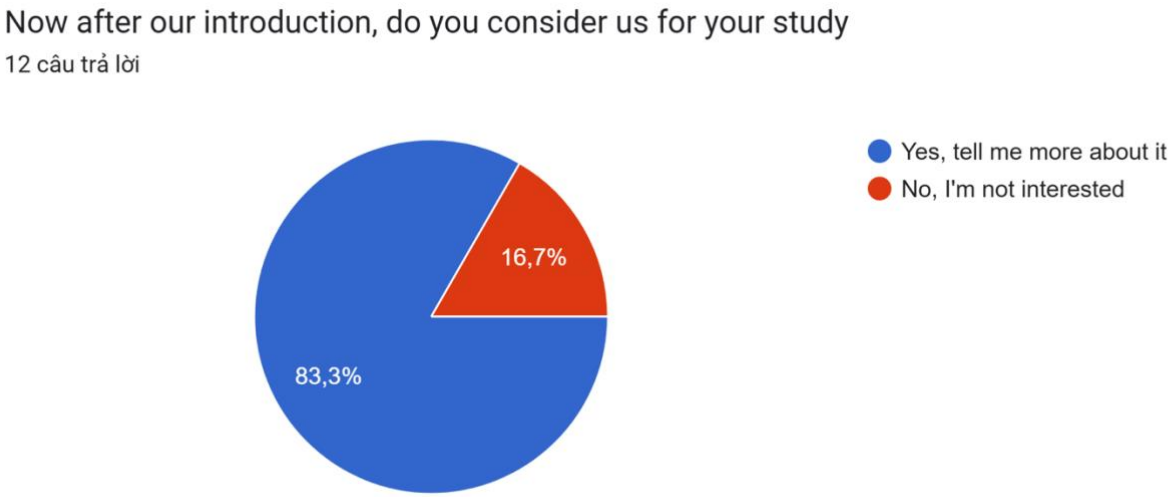


Figure 10: User's survey "How likely are you to continue using SocratesCode for your programming studies"

5.2 User Experience and Interface Performance

The project's user satisfaction goal of 90% has been exceeded by the web-based interactive interface. When combining the ratings for outstanding overall user experience (38.2% rated 5/5), and good user experience (54.5% rated 4/5), it received a total satisfaction score of 92.7%. It is evident that the platform exceeded the high user satisfaction standards outlined in the success requirement, confirming the platform's user-friendly interface and intuitive design. With 92.7% of users expressing satisfaction and 7.3% expressing neutrality, it appears that no consumers were dissatisfied with the encounter. Although there is potential for improvement to turn skeptical users into ones who are satisfied, these figures demonstrate a consistently positive experience throughout.

With 76.4% of respondents finding the web-based interface to be extremely intuitive and quickly easy to use, the interface usability evaluation may further support the good findings above. Another 20% thought that using the interface required some getting used to, but that it was simple once they knew how to use it. This demonstrates that the initiative effectively met its stated goal of offering an intuitive platform that reduces any elements that would discourage students from participating.

However, the poll identified a crucial area that requires more focus: real-time interaction quality. According to 67.3% of respondents, reaction times were good, with perhaps little but tolerable delays; 21.8% thought they were great, responding right away; and 10.9% experienced delays that were quite visible and had a significant impact on usability.

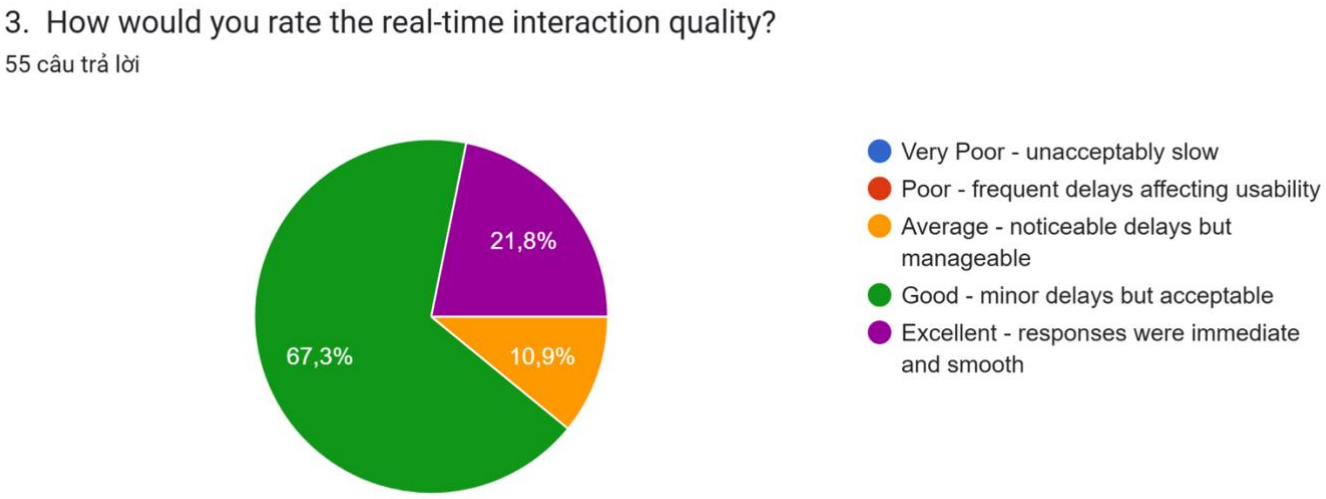


Figure 11: User's survey "How would you rate the real-time interaction quality"

5.3 Educational Effectiveness and Socratic Methodology Implementation

When it came to applying the Socratic technique of questioning, the primary teaching exercise on SocratesCode demonstrated remarkable efficacy. According to survey results, 50.9% of respondents stated the methodology was very good in assisting them in solving problems on their own, and 14.5% claimed it was extremely effective at improving their understanding. With an additional 27.3% deeming it moderately effective, the positive-emphasis rating came to 92.7%. Not a single person stated that the strategy had no effect at all, and up to 7.3% thought there was minimal influence.

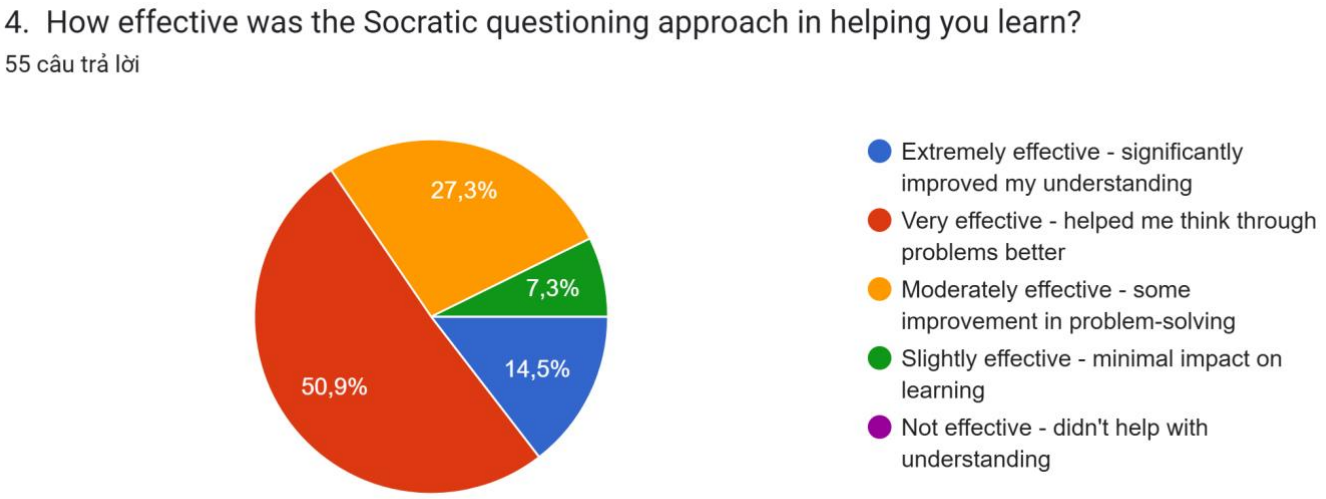


Figure 12: User's survey "How effective was the Socratic questioning approach in helping you learn"

The AI instructor performed well in terms of adaptive skills; 69.1% of users reported that the system was mostly with sufficient complexity and functioned well for their level of understanding. 14.5% of respondents said it was partly suited but occasionally too easy or too hard, while 16.4% thought it was

ideal. The 99.0% success rate of adaption, at least partially, is a testament to the effectiveness of the reasoning evaluator and individualized student assessment components.

The system's ability to break down complex problems into manageable components received positive feedback, with 58.2% of users satisfied with the breakdown process and 29.1% very satisfied with clear problem decomposition. Only 12.7% found the breakdown adequate but requiring improvement, indicating successful implementation of the incremental algorithm designed to collaborate with LLMs for optimal learning experiences.

58.2% of users expressed satisfaction with the breakdown process, and 29.1% expressed high satisfaction with the system's capacity to decompose complicated problems into manageable parts. The progressive approach, which was created to work with LLMs to provide the best learning experiences, was successfully implemented, since just 12.7% of respondents thought the breakdown was fair but needed better.

5.4 Response Quality and Accuracy Analysis

With its outstanding performance in upholding pedagogical integrity and providing precise answers, the AI instructor has fully satisfied the project's essential objectives. The answer was able to 100% guarantee the Socratic method requirement, which states that no response is ever provided in direct written code form. According to the grading analysis, even under the most negative conditions, the system obtained a high score of more than 60 points on each evaluation criterion, with no instances of direct answers that were given out.

Consistent quality assurance is present in the automated grading system that conducts evaluations using standardized criteria applied to both the GPT-4o-mini and GPT-5-mini models. According to the distribution study, GPT-5-mini replies seemed to be spread out in the 70-80 score range, with the median being close to 75, while GPT-4o-mini responses primarily congregated in the 80 to 82 score range, with the mean score being 81. Strong consistency in performance is also suggested by the boxplot analysis, which shows that GPT-4o-mini reported a narrower interquartile range (roughly 74–81) than GPT-5-mini, which had a wider range (roughly 71–79); both models, however, were able to keep their minimum scores comfortably above the 60-point threshold.

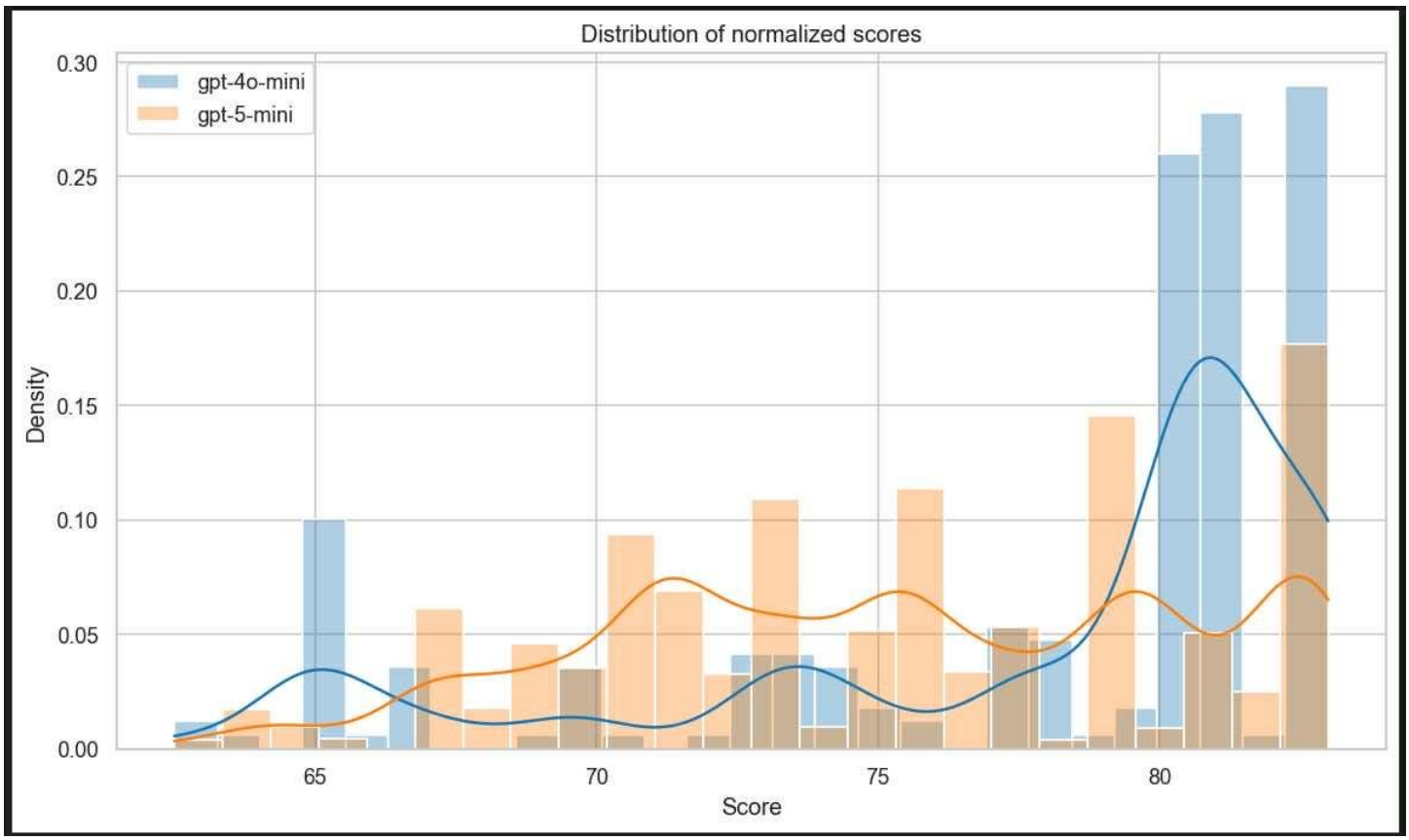


Figure 13: Distribution of normalized scores: gpt-4o-mini vs gpt-5-mini

This is aligned with the conventional user perception statistics, which showed that 18.2% of answers were classified as having very-high accuracy (90–100%) and 67.3% of responses fell into the typically a little bit correct and relevant (80–89%) accuracy range. In addition, 10.9% of replies were classified as occasionally accurate, resulting in a final positive percent accuracy of 96.4.

The system's flawless adherence to Socratic principles while retaining helpfulness is a particularly noteworthy accomplishment. According to the survey, 30.9% of users thought this balance was outstanding, while 56.4% of users thought the AI instructor offered solid advice without offering straight answers. 12.7% more people thought it was fair, which means that it fully complied with the basic pedagogical restriction. The efficiency of the reasoning evaluator and the fundamental design concepts are validated by this total success in avoiding direct code provision while preserving educational value.

5.5 Critical Issues and Areas for Enhancement

Although the project has clearly met and surpassed the fundamental goals for both educational efficacy and user satisfaction, the results of the performance testing offer several opportunities for improvement. The logic validator is the main bottleneck, requiring nearly three times as long as the other components, according to the response time analysis, which went from speculation to facts. Therefore, these technical findings provide a target for improvement and directly relate to the 10.9% of users who experience significant delays in system interactions.

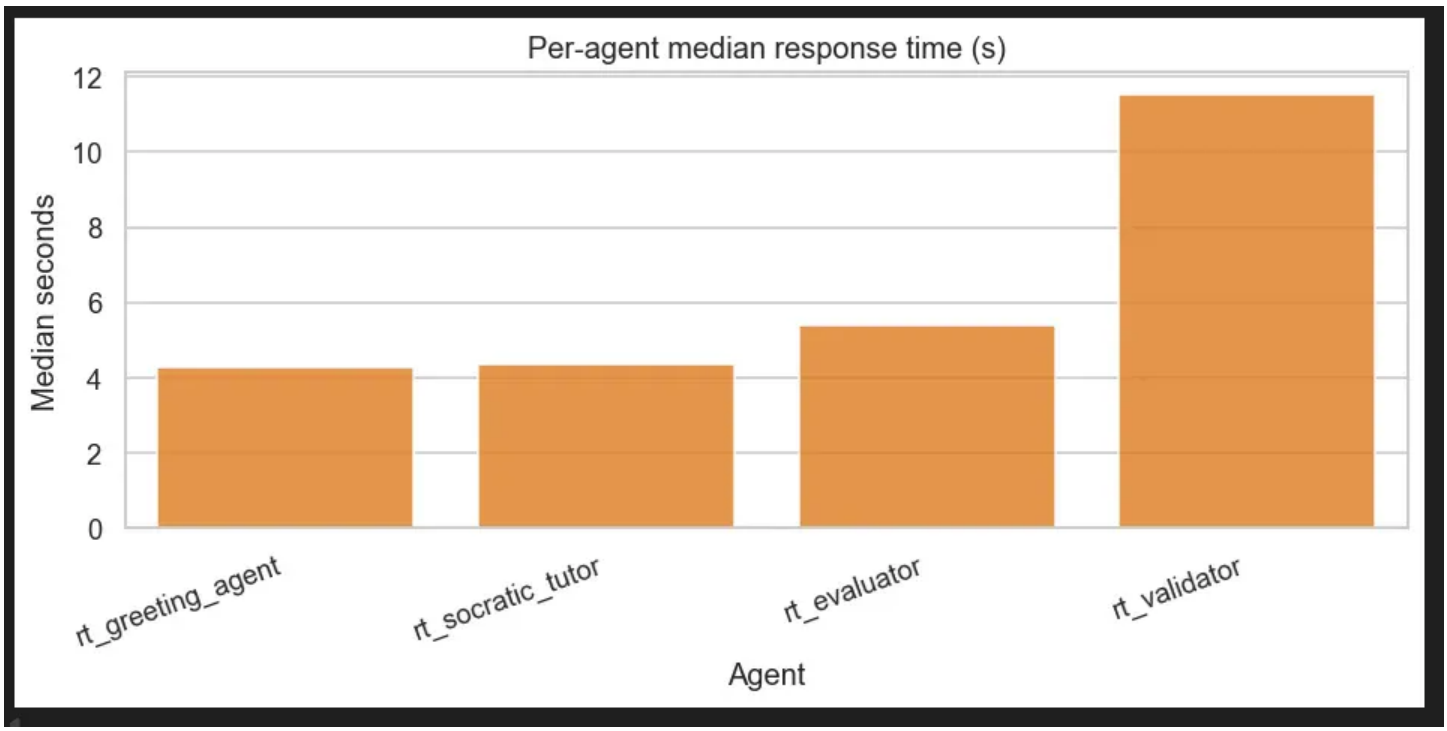


Figure 14: Median runtime per agent

The use of adaptive timeout mechanisms or question complexity pre-assessment is justified by the variation in per-question iteration time-performance, which ranges from extremely efficient response times of less than 15 seconds to complex procedures that take up to 30 seconds. It would help control user expectations. Despite continuously fulfilling project criteria, the accuracy data show that the reasoning evaluator might be improved further to achieve more consistent performance in the highest accuracy ranges.

6 Summary and reflection

6.1 Project Journey Overview

It has been an incredible trip, and the SocratesCode capstone project has successfully concluded as an AI-powered Socratic tutoring agent for teaching programming. Together with the Hanoi team's excellent contributions, teams working in other places were able to produce a respectable solution that was far above the initial project needs and expectations.

6.2 What Worked Well

Team Collaboration and Engagement: Perhaps the team's primary strength was that everyone had been actively involved in the project and had been working on it continuously. Even though they were geographically separated, the inter-location cooperation was quite successful. We were able to maintain our momentum and produce high-quality results thanks to the dedication and hard work of both teams. Despite differences in personal schedules, coordination was made easy by the effective meeting arrangements and communication channels that were already in place.

Technical Implementation Success: It appeared to be a reasonable technological choice to use ChatGPT 4.0 API integration in a multi-agent AI architecture. With a successful deployment on RMIT's AWS RACE platform, we demonstrated our capacity to provide cloud solutions that meet institutional needs. We were able to make changes to the system based on ongoing testing and feedback thanks to our iterative development process.

User-Centered Design Approach: Responding to actual educational problems was crucial when conducting extensive user research and gathering feedback. An interface that consumers felt comfortable using was the result of numerous design process revisions that included user testing.

Methodological Rigor: Stable progress toward our goals was made possible by a methodical approach to project management, alternating with milestone reviews and structured testing. In order to maintain quality standards, it was advantageous to use thorough evaluation frameworks early in the development process.

6.3 Areas for Improvement and Lessons Learned

Performance Optimization: Important insights about component relationships and system design were discovered throughout development. Despite the system's generally good performance, there were still valuable lessons to be learned regarding early performance testing and bottleneck discovery that would be applied to future projects.

Response Time Management: Throughout the course of the project, it became increasingly difficult to strike a balance between user expectations and system complexity. This alone raised awareness of the necessity of flexible methods and open communication regarding the capabilities of the system from the very beginning.

Resource Planning and Risk management: Important lessons regarding risk management and financial planning were imparted by the project. Working with Gemini's free API early on was a smart move because it not only saved us money but also provided a slightly unplanned backup. This tactical decision proved essential when we saw an unanticipated overnight spike in API access that used an expensive model to consume over 100 million tokens, depleting about 50% of our budget in a single night. Although the details of this erratic API usage in relation to OpenAI API secret key management are still unknown, this incident brought API security, usage monitoring, and the necessity of having backups in case of emergencies into sharp relief. The event happened even though monitoring tools were in place, such as live monitoring for individual conversations during testing, thorough project monitoring via Langfuse, and budget limitations with alerts set up on the OpenAI platform. We learned that monitoring alone is insufficient without appropriate model-specific constraints and automatic use management when the OpenAI budget alerts failed to stop the unusual usage rise. The entire situation might have severely damaged our ability to go through the testing and development stages if it weren't for the development expenditures that were saved by not using OpenAI throughout the development process.

6.4 Key Success Factors

A successful project was the result of several factors. Important technological concepts and institutional expertise were formed by strong academic supervision, especially from Mr. Patrick and our academic supervisors. Despite being geographically dispersed, the team's dedication to consistent communication and organized project management offered an effective means of activity coordination. Instead of trying an AI tutoring technique on a wider social spectrum, it was more crucial to limit the issue to a single strategy: hard coding the flow in order to implement it thoroughly and achieve better outcomes.

6.5 Recommendations for Future Phases

Security and Monitoring: The unanticipated token usage issue exposed weaknesses in our risk mitigation strategy, even though the project successfully deployed extensive monitoring tools, such as Langfuse for project-wide analytics and live chat monitoring for testing. The OpenAI platform has alerts and budget limitations set up, but they weren't enough to stop the overnight spike in usage. We learned from this experience that model-specific use limits and automated cutoff methods are necessary for effective API security instead of depending only on monitoring and alert systems. To avoid such

occurrences, future projects should incorporate automatic usage constraints and precise access controls for every model.

Technical Enhancements: In order to speed up reaction times, emphasis should be placed on streamlining the reasoning evaluator component through parallel processing or algorithm improvement. The responsiveness of the system could be greatly increased by implementing caching methods for commonly asked questions.

Scalability Considerations: In order to maintain performance standards with an increase in the number of users, load balancing and distributed processing capabilities must be implemented as the system goes toward wider deployment

User Experience Refinements: Creating more advanced tools for tailored learning paths and progress tracking will improve the efficacy of education, according to user input.

Continuous Monitoring: For continuous performance optimization and user experience improvement in production situations, it will be essential to set up thorough analytics and monitoring systems.

7 References

[1] RMIT, "RMIT Vietnam celebrates the largest ever group of graduates," RMIT University Vietnam, (accessed April 20, 2025). [Online]. Available: https://www.rmit.edu.vn/news/all-news/2024/apr/rmit-vietnam-celebrates-the-largest-ever-group-of-graduates

[2] RMIT, "Academic Progress," RMIT University Vietnam, (accessed April 20, 2025). [Online]. Available: https://www.rmit.edu.vn/students/my-studies/assessment-and-results/academic-progress

[3] Hoang Van, “Hoang_Van_2025_Capstone_R&D Project Proposal_LLM_Socratic_Tutor,” RMIT University SSET Project Description, Vietnam, Apr. 2025

[4] Vidyanchal High School, " Power Of Asking Questions In A Classroom,” Vidyanchal High School, (accessed April 22, 2025). [Online]. Available: https://vidyanchalschool.com/power-of-asking-questions-in-a-classroom/

[5] Sweller, J. (2020). Cognitive load theory and educational technology. *Educational Technology Research and Development*, 68(1), 1-16. Retrieved from https://eric.ed.gov/?id=EJ1243787

[6] Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, context, and worked examples in learning computing problem solving. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 21-29.

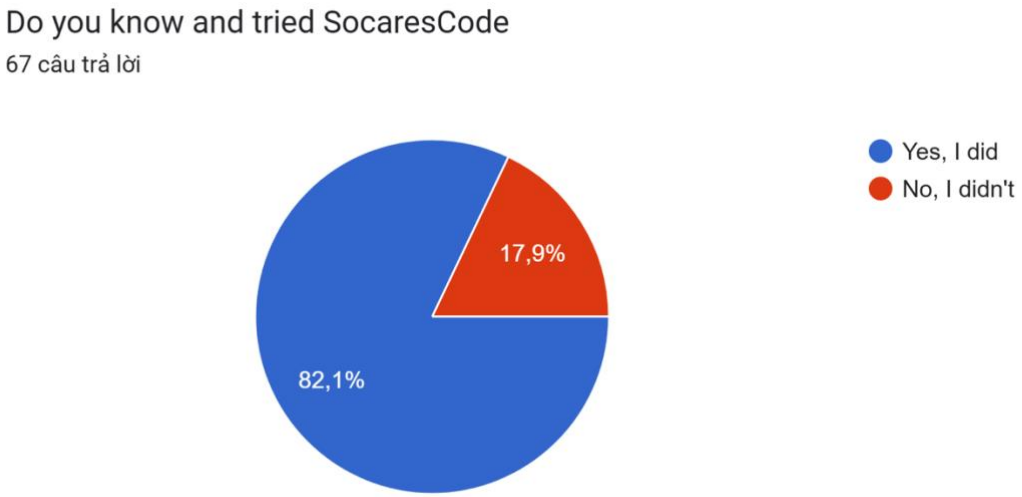
[7] NIST AI Risk Management Framework (AI RMF 1.0). (2023). National Institute of Standards and Technology. Retrieved from https://www.nist.gov/itl/ai-risk-management-framework

[8] Edutopia. (2024). AI tutors can work—with the right guardrails. Retrieved from https://www.edutopia.org/article/ai-tutors-work-guardrails/

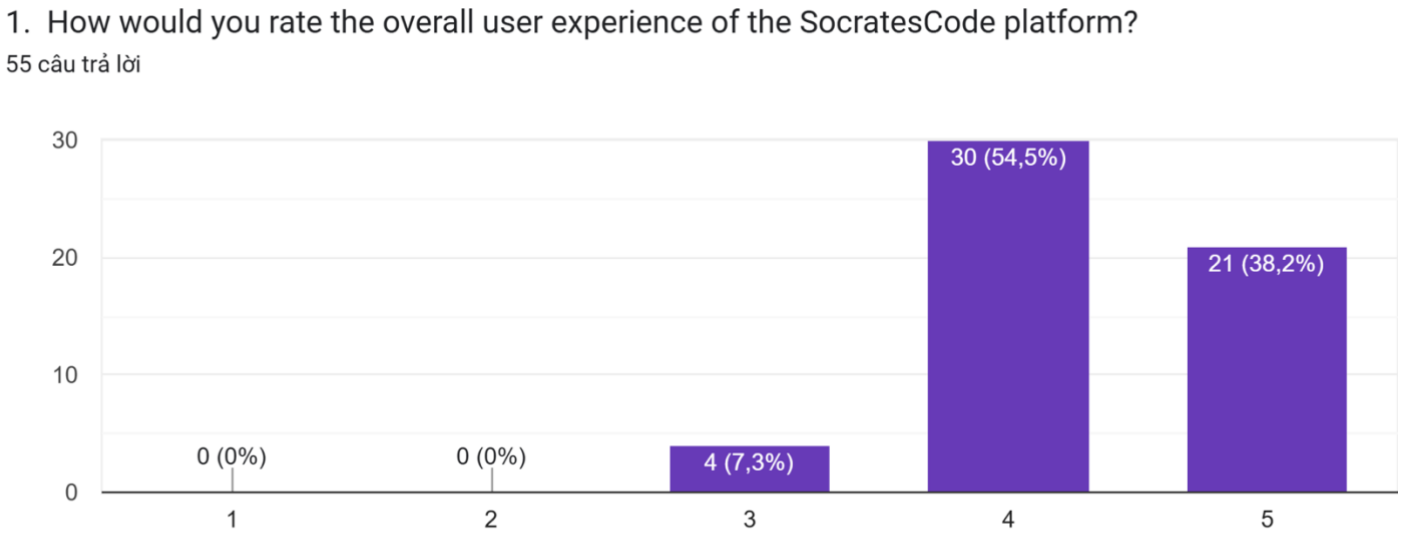
[9] Wharton Knowledge. (2024). Without guardrails, generative AI can harm education. Retrieved from https://knowledge.wharton.upenn.edu/article/without-guardrails-generative-ai-can-harm-education/

[10] Hake, R. R. (1998). Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66(1), 64-74.

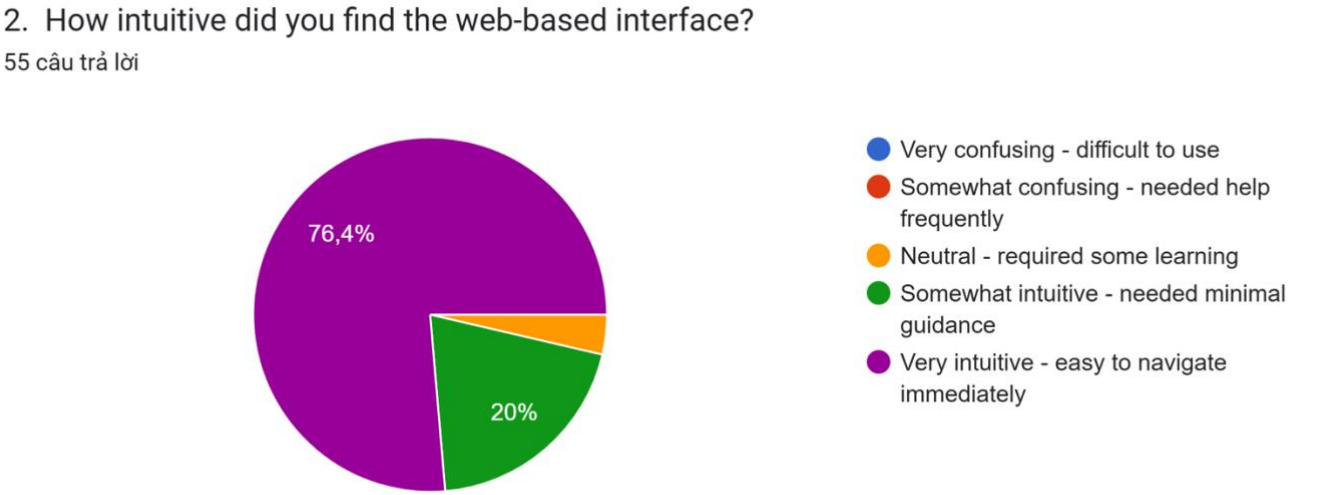
8 Appendices



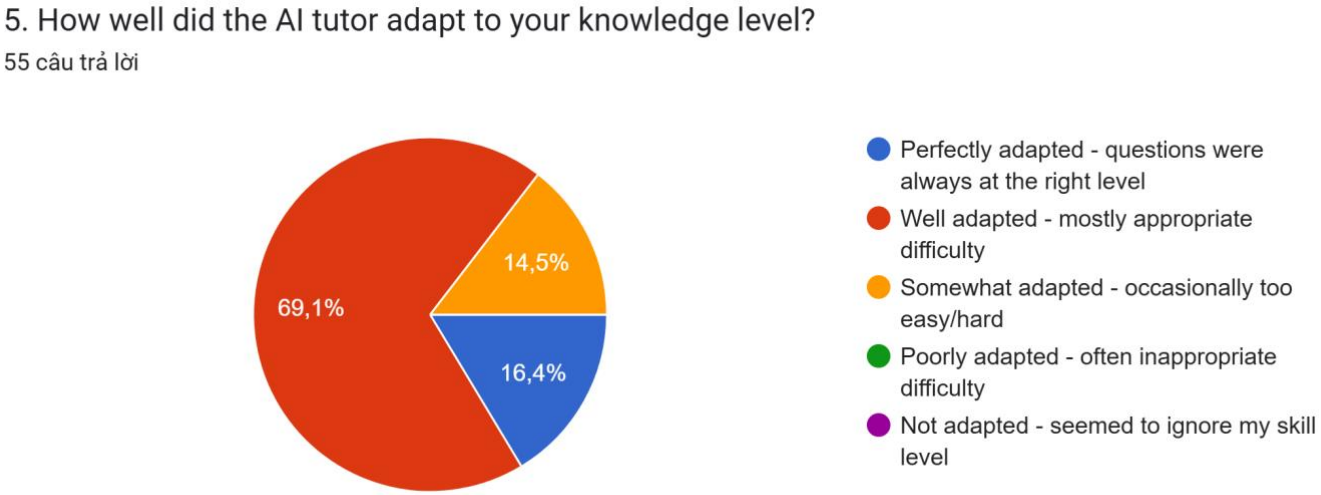
Appendix Figure 1: User's survey "Do you know and tried SocratesCode"



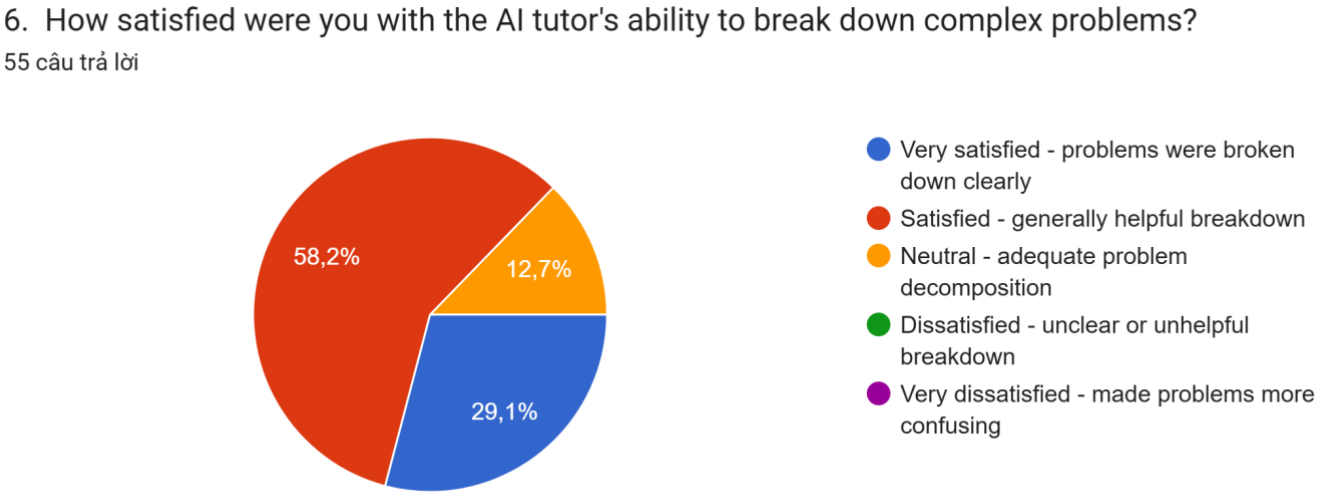
Appendix Figure 2: User's survey "How would you rate the overall experience of the SocratesCode platform"



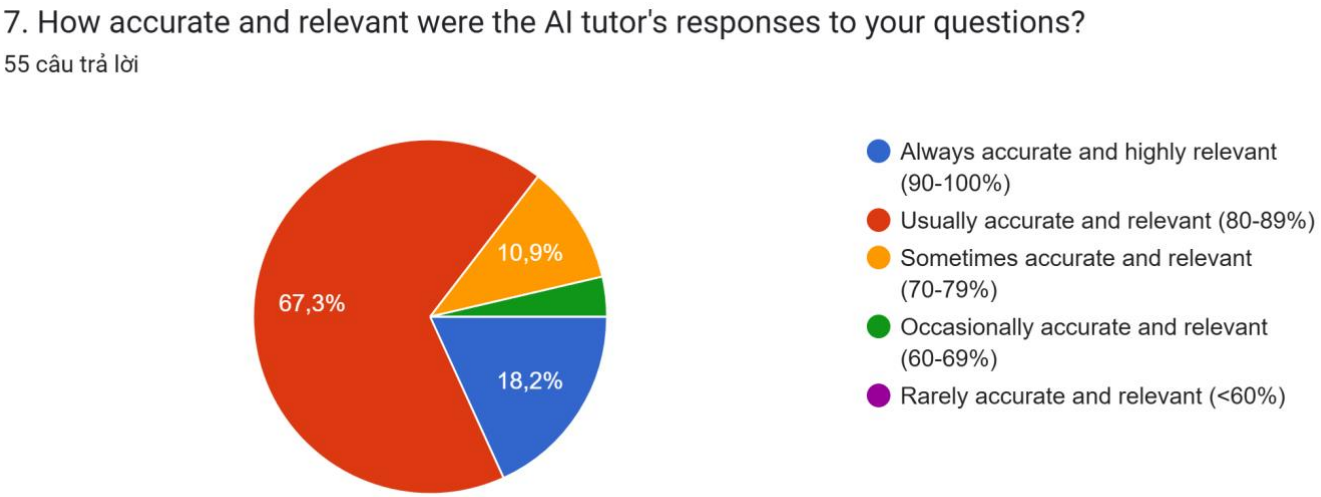
Appendix Figure 3: User's survey "How intuitive did you find the web-based interface"



Appendix Figure 4: User's survey "How well did the AI tutor adapt to your knowledge level"

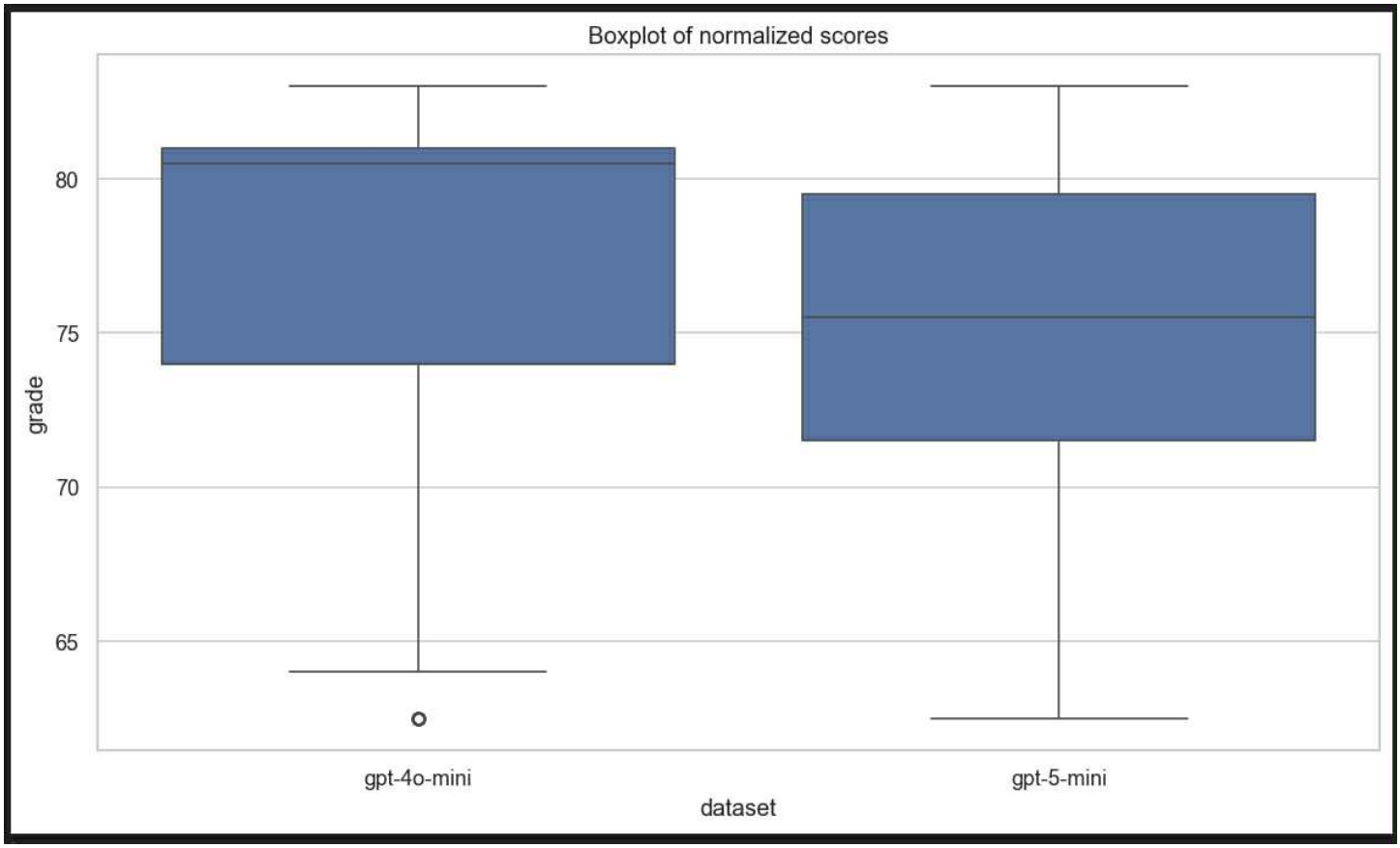


Appendix Figure 5: User's survey "How satisfied were you with the AI tutor's ability to break down complex problems"

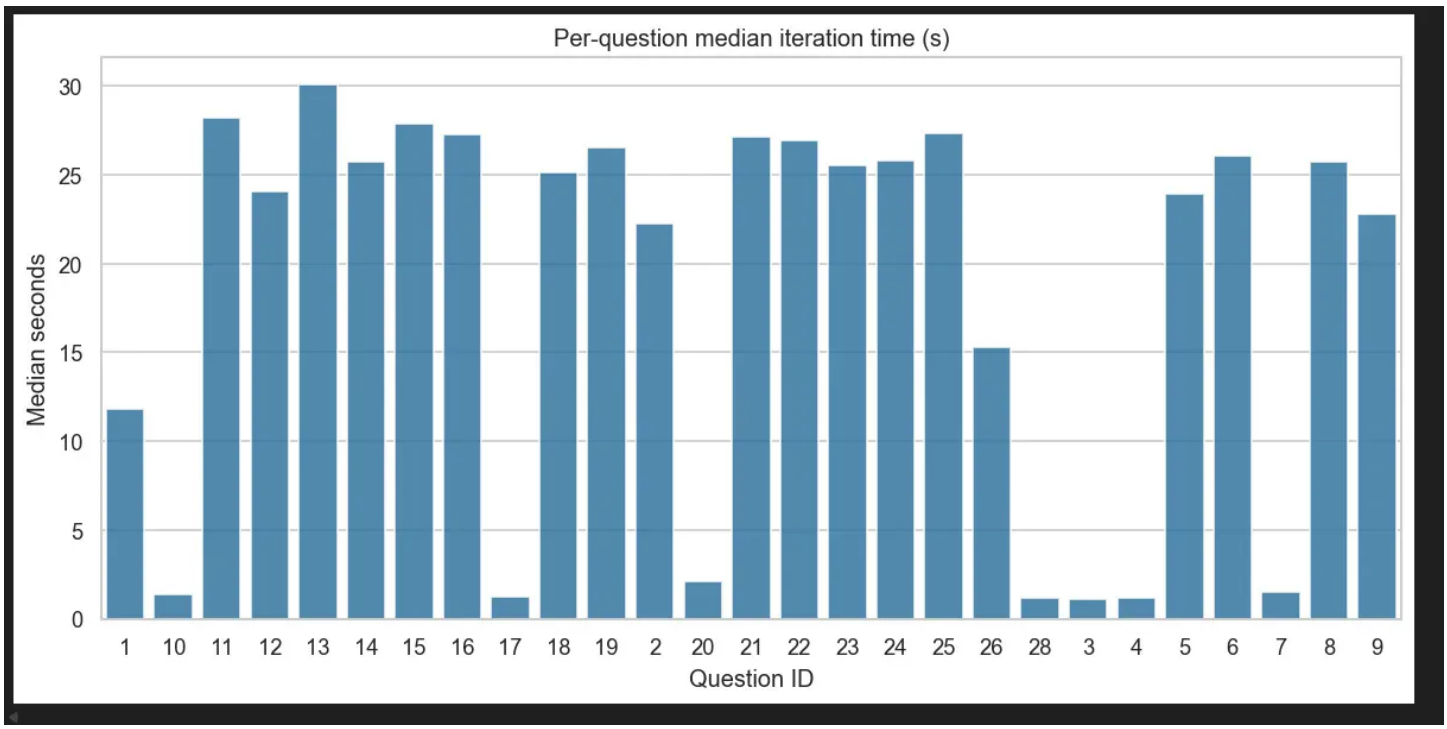


Appendix Figure 6: User's survey "How accurate and relevant were the AI tutor's response to your queastion"

Appendix Figure 7: Violin plot of normalized scores



Appendix Figure 8: Boxplot of normalized scores



Appendix Figure 9: LeetCode's question 10 iteration median time

9 Meeting journals

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 01

Meeting Details

Date:	8 April 2025
Time:	14:00 – 15:00
Attendees:	Vo Minh Thien An, Hua Van Anh Khoa, Bui Minh Khoi, Luong Dinh Khang, Hoang Duc Anh, Dr. Phong Ngo, Dr. Hoang Van
Apologies:	Hoang Duc Anh joined late
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Team Introduction and Project Overview: All team members introduced themselves and their background. Project scope and objective for the project were discussed. Teams established communication protocols and meeting schedules.

2	Role and Future goal: Discussed about Ca Chep team's role and future plan to commit to the project

Action Items

No	Item	Who	By
1	Assign role and think about what each member can contribute	Ca Chep team	08 April 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 02

Meeting Details

Date:	8 April 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member, Dr. Phong Ngo, Dr. Hoang Van
Apologies:	None
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Team Introduction and Project Overview: All team members introduced themselves and their background. Project scope and objective for the project were discussed. Teams established communication protocols and meeting schedules.
2	AI Framework Evaluation: Discussed pros and cons of using Autogen AI vs Code Help for the project. Initial research findings were presented. Hua Van Anh Khoa will share Autogen AI source code for further evaluation, Decision on final framework selection was deferred pending more research.

Action Items

No	Item	Who	By
1	Complete meeting minute, team role, architecture document.	Ca Chep team and AlgoRythm team	25 April 2025

2	Update and share Autogen AI source code for discussion	Hua Van Anh Khoa	10 April 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 03

Meeting Details

Date:	25 April 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member, Dr. Hoang Van
Apologies:	None
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	AWS Budget reconsider: Dr. Hoang agree to re-consider about AWS budget since web hosting is costly and OpenAI had just updated their price
2	Assign task for both team: Ca Chep team will take task M1 and 2, AlgoRhythm work with M2 and 3. After all will come together and work on M5

Action Items

No	Item	Who	By
1	Complete meeting minute, architecture document.	Both team	05 May 2025
2	Improve proposal	Both team	29 April 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Team Meeting
Meeting No: 04

Meeting Details

Date:	25 April 2025
Time:	15:00 – 16:00

Attendees:	Ca Chep team's member, AlgoRhythm team's member
Apologies:	None
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Task Assign: Both teams agreed to work on FE and BE for web application first then split and work on their own task
2	Team meeting consideration: Both teams agreed on the common communication protocol

Action Items

No	Item	Who	By
1	None		

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 05

Meeting Details

Date:	05 May 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member, Dr. Phong
Apologies:	Dr, Hoang Van
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Weekly report: 2 team share about their assignment 1
2	Diagram discussion: Ca Chep team share their diagram for AI tutor while Dr. Phong and AlgoRhythm team discuss, ask question and give idea to improve it

Action Items

No	Item	Who	By
1	Diagram improve: 2 team will need to discuss together and come up with the improved version of the diagram	2 team	9 May 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Team Meeting
Meeting No: 06

Meeting Details

Date:	05 May 2025
Time:	15:00 – 16:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Task discussion: 2 Team agreed to use existing website made by AlgoRhythm team and improve it
2	Face to face discuss: Khoa and AlgoRhythm team will meet at Hanoi campus for further dicussion

Action Items

No	Item	Who	By
1	Khoa and Hanoi team meeting	Khoa, Minh and Hung	07 May 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 07

Meeting Details

Date:	09 May 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member, Dr. Hoang Van
Apologies:	Dr. Phong, Khoa because of headache
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Diagram discussion: Teams share their improvement in the diagram
2	Website demo: AlgoRhythm team showcase their demo website interface and FE with no BE or AI integrated yet
3	Split the project: Dr. Hoang and Dr. Phong split the project into 5 small module M1 à M5. Ca Chep will work on M1,2 while AlgoRhytm work on M2,3.
4	Tasks prioritize: Both teams want to finish the website before working on AI task so that they will not have anything else to worry. The website is expected to finish in 2 weeks

Action Items

No	Item	Who	By
1	Diagram: Khoa will continue to work on diagram improvement	Khoa	16 May 2025
2	Website: Other member will work on a full-stack website	Both team	16 May 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Team Meeting
Meeting No: 08

Meeting Details

Date:	11 May 2025
Time:	22:00 – 22:30
Attendees:	Ca Chep team's member, AlgoRhythm team's member
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Task assign: assign task and set deadline for each team member

Action Items

No	Item	Who	By
1	Website without AI	Both team	16 May 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 09

Meeting Details

Date:	30 May 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoa, Khang
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	OpenAI permission: An asked for further permission about OpenAI API purchase
2	Presentation question: Supervisor asked if there are any problem with the presentation and do teams need any help
3	Semester break: Supervisor and team discussed about sem-break plan, asked teams to be more targeted and focused on their task

Action Items

No	Item	Who	By
1	No	No	no

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 10

Meeting Details

Date:	6 June 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoa, Khang
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Database justification and presentation: 2 teams come up with their version of project diagram and agreed to use it for majority of the project
2	Basic system demo: 2 team demo their work on the basic interface for at least 2 AI agent run cooperative
3	Budget allocation: An finish his budget request document

Action Items

No	Item	Who	By
1	Group chat agent running, continue add more agent	Khoa, Algorythms	11 June 2025
2	Agent cofiguration include prompt, parameter, model	Both team	11 June 2025
3	Finalize agent flow design	An, Khoi	11 June 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 11

Meeting Details

Date:	13 June 2025
Time:	14:00 – 15:00

Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoa, Duc Anh, Khoi, Minh, Quan
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Final diagram: Team finalizes their diagram for the last time, fix error that pointed out by supervisors
2	Share current agent group chat: teams share their current approach on multi agent technology, promises to deliver basic full agent group chat 27 June
3	Ask for cancel meeting: due to personal issue, many team member was unable to attend the usual weekly meeting on 20 June

Action Items

No	Item	Who	By
1	Full agent group chat	Both team	27 June 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 12

Meeting Details

Date:	27 June 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoa
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Demo: Team shows the promised chat with multi agent workflow
2	Prompts improve: Several improvements needed

3	AWS hosting: teams ask for help with RMIT RACE to host their website

Action Items

No	Item	Who	By
1	Prompt improvement	Both team	01 July 2025
2	AWS hosting	Both team	01 July 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 13

Meeting Details

Date:	1 July 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoa, Khang
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	New meeting schedule: Meeting date changed from Friday to Wednesday
2	Small task week: Due to the change in meeting date teams ask for less task finish
3	File reading: Supervisor suggest technology to read JSON file, both teams considering it

Action Items

No	Item	Who	By
1	Test workflow effectiveness	An	8 July 2025
2	Mem0 implementation	Khoa	1 August 2025

3	AWS hosting: after have access to RACE 2 teams need to deploy a small test program to test if this is fit with their scope of web hosting	Both team	8 July 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 14

Meeting Details

Date:	9 July 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Task allocation adjusts teams need to be more detail regard their task assignment
2	Test pipeline: The pipeline is working for basic scenario, try to test on more complex one
3	Multiple section: the single user flow is good; teams need to consider on the scale part so that the system can work on multiple user section.

Action Items

No	Item	Who	By
1	Modify agent group	Hung	16 July 2025
2	Flow control QA test scripts	Khang	16 July 2025
3	API connects	Duc Anh	16 July 2025
4	System agent flow visualization	An	16 July 2025
5	RACE explore	Minh, Khoi, Quan	23 July 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 15

Meeting Details

Date:	16 July 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Cloud attempt: teams show their attempt on utilizes RACE as a platform and some of their stress test with it
2	Asm 1: team ask for support and review their asm 1

Action Items

No	Item	Who	By
1	JWT authentication set up for BE, test script for BE	Hung	23 July 2025
2	JWT token authentication for BE	Khang	23 July 2025
3	JWT token authentication for FE	An	23 July 2025
4	UI debug, light/dark mode	Duc Anh	23 July 2025
5	Finish mem0 implementation	Khoa	23 July 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 16

Meeting Details

Date:	23 July 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Last week task completion: teams summary their last week tasks
2	New task: assign new task and attempt to finish over-due tasks

Action Items

No	Item	Who	By
1	Collect and execute leetcode question for testing	An, Khang	30 July 2025
2	Deploy a simple Cloud version	Hung, Minh, kien, Khoi	30 July 2025
3	CI/CD workflow on Github	Hung, Minh	30 July 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 17

Meeting Details

Date:	30 July 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	System demo: Team demo their mostly finished application for feedback
2	New task: assign new task and attempt to finish over-due tasks

Action Items

No	Item	Who	By
1	Finish overdue task, merge FE components	Duc Anh	6 August 2025
2	Deployment plan and deploy a stable version	Khoi, Quan, Minh	6 August 2025
3	Upgrade mem0 for more detailed context building	Khoa	6 August 2025
4	Collect input + output dataset, create test scripts	An	6 August 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 18

Meeting Details

Date:	06 August 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khang
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Current data: Teams show their approach on data collection and current progress
2	New task: assign new task and attempt to finish over-due tasks
3	No meeting next week

Action Items

No	Item	Who	By
1	Create criteria for evaluation	An, Minh, Hung	20 August 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 19

Meeting Details

Date:	20 August 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	Khoi
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Analysis agent: Teams show their analysis agent to analyse, and grading based on the defined criteria
2	Weighted evaluation metrics: supervisor suggest using weighted evaluation metrics so that the grading can be more accurate

Action Items

No	Item	Who	By
1	Improve the website	Both team	18 Septemper 2025

SocratesCode: AI-Powered Socratic Tutor for Programming
Project Kick-off Meeting
Meeting No: 20

Meeting Details

Date:	12 Septemper 2025
Time:	14:00 – 15:00
Attendees:	Ca Chep team's member, AlgoRhythm team's member
	Dr. Hoang, Dr. Phong
Apologies:	
Copy To:	

Information /Decision

Item No.	Discussion Summary
1	Final meeting: Supervisor ask both teams to prepare to wrap up their work, write documentation and design test case scenario
2	Announce Algorythms Hanoi campus award

Action Items

No	Item	Who	By
1	Improve the website	Both team	18 Septemper 2025