MULTI-AGENT ADAPTIVE LEARNING FOR MATHEMATICS

Srikanth Dokku, Anudeep Gumpula, Sanjana Gudati, Mahendra Nagisetty, Ruthvik Thimmarayappa, Jai Paul Yeruva, and John Glossner*

Department of Mathematics and Computer Science, Rivier University

ABSTRACT

We present a prototype adaptive learning framework using multiple large language model agents using OpenAI's GPT-4o model. Each agent is specialized to a specific aspect of adaptive learning. The agents communicate with each other using the autogen multi-agent framework. Group communications are implemented in an unconstrained, semi-constrained, and fully constrained manner. The unconstrained agent communications allowed the `autogen GroupChatManager` *to select the next agent to "speak" based only on the description of what the agents were specialized in. The semi-constrained agent selection used both allowed and disallowed transitions. The constrained agent communications used a state machine to select the next agent. Using constrained communications allowed predictability in the sequencing of tasks but is less flexible in handling arbitrary student input. Unconstrained communications occasionally experienced agent role confusion. Our prototype system using constrained communications teaches a learner new material and tests them on mastery. The code implementing the experiments is open-source and available on github*[1].

***Keywords**:* Adaptive Learning, Multi-Agent Systems, Large Language Models, Artificial Intelligence

1 Introduction

Brusilovsky and Peylo (2003) define *Adaptive Learning* as "building a model of the goals, preferences and knowledge of each individual student and using this model throughout the interaction with the student in order to adapt to the needs of that student. They also attempt to be more intelligent by incorporating and performing some activities traditionally executed by a human teacher - such as coaching students or diagnosing their misconceptions."

Van Schoors et al. (2021) describe the decisions that can be made in adaptive learning. The variables include when, what, and how to adapt as well as adapt to what. When to adapt can be static or dynamic. What to adapt can be content, presentation, or support. How to adapt can be learner-controlled or program-controlled (or both). And adapt to what can be learner parameters or learner-system parameters.

1.1 Cognitive Skills Taxonomy

Many adaptive learning systems have started from Bloom's foundational cognitive skills taxonomy (Bloom et al., 1956). They arranged levels hierarchically ranging from simple to more complex thinking. Anderson and Krathwohl (2001) revised the taxonomy changing nouns to verbs. Table 1 shows both Bloom's original taxonomy and the revisions.

More recently, Sun et al. (2023) developed a cognitive model specifically for Mathematics. Starting from the Trends in International Mathematics and Science Study (TIMSS) cognitive framework (Mullis

* corresponding author: jglossner@rivier.edu

[1] https://github.com/Rivier-Computer-Science/Adaptive-Learning

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

et al., 2020), they surveyed experts which led to the model shown in Figure 1. The authors verified this model by testing it on randomly selected fourth-grade students.

Table 1: Comparison of Bloom's Taxonomy with Anderson's Revision (Anderson and Krathwohl, 2001)

Original Bloom (1956)	Anderson & Krathwohl (2001)	Descriptions	Key Changes
Knowledge	Remembering	Recalling facts, terms, and basic concepts	Nouns to verbs
Comprehension	Understanding	Demonstrating comprehension by explaining or interpreting information	
Application	Applying	Using knowledge in new situations to solve problems or carry out tasksl	
Analysis	Analyzing	Breaking down information into parts, identifying relationships, and determining how elements are organized	
Synthesis	Evaluating	Making judgments based on criteria, standards, or evidencel	Order switched
Evaluation	Creating	Putting elements together in new ways to produce something original or reorganize ideas	

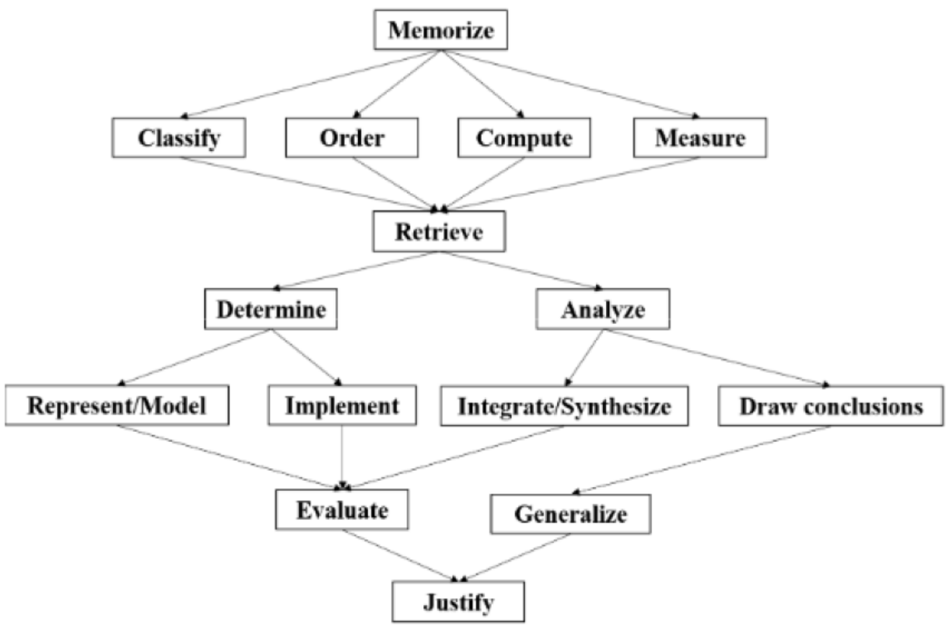


Figure 1: Mathematics Cognitive Model (Sun et al., 2023)

1.2 Knowledge Tracing

Knowledge tracing (KT) powers intelligent tutoring systems (ITSs), tailoring content sequencing, feedback, and practice based on individual student needs (Pavlik et al., 2009). Knowledge Tracing is a fundamental technique in educational data mining (EDM) that tracks a learner's evolving knowledge state over time (Anderson et al., 1986). It is "the ability to effectively track the learning progress of a student through their online interaction with teaching materials. The aim is to observe, represent, and quantify a student's knowledge state, e.g., the mastery level of skills underlying the teaching materials" (Abdelrahman et al., 2023). KT represents a learner's knowledge state in terms of mastery of distinct skills or concepts within a domain. Knowledge states are usually modeled as discrete (mastered/not mastered) or continuous (proficiency levels) (Corbett and Anderson, 1994).

Beck and Gong (2013) show that KT supports mastery learning approaches, allowing students to progress only when they have demonstrated proficiency in prerequisite skills. KT aids in the development of adaptive tests that dynamically adjust difficulty based on student performance, ensuring a more accurate evaluation of their ability.

1.3 Paper Organization

This paper is organized as follows: in Section 2 we discuss the use of Large Language Models in adaptive learning systems, in Section 3 we discuss the methodology of our prototype system and ethical considerations for their use, in Section 4 we present the results of our experiments, in Section 5 we discuss our results, in Section 6 we discuss future enhancements to our prototype system, and in Section 7 we present conclusions.

2 Related Work

2.1 Large Language Models

Naveed et al. (2024) present an extensive review of the development, architectures, training methodologies, and diverse applications of large language models (LLMs). They detail the evolution and significance in applications such as natural language processing (NLP), text generation, machine translation, summarization, and question answering. They discuss key architectures used in LLMs such as Bidirectional Encoder Representation from Transformers (BERT), Generative Pretrained Transformer (GPT), and the Text-to-Text (T5) transformer. Both BERT and T5 are open-source.

The integration of Large Language Models (LLMs) and Artificial Intelligence Conversational Agents (AICs) in education, particularly as tutors in Intelligent Tutoring Systems (ITS), is a topic gaining considerable interest (Abedi et al., 2023). The ability of these technologies to engage learners, provide personalized responses, and support self-directed learning presents a significant potential for enhancing the learning experience. Virvou and Tsihrintzis (2023) present a framework for evaluating LLMs in an ITS where both hard (technical) skills and soft skills such as critical thinking, ethical reasoning, and effective communication are considered.

The Khan Academy introduced a ChatGPT4-based tutor (Singer, 2023) and recently announced a partnership with Microsoft (Microsoft, 2024b). The AI tutor generates python code to solve complex mathematics problems. OpenAI also has a custom-built GPT specifically for mathematics using the same technique (ChatGPT, 2024). Recognizing that LLMs are not well suited for directly solving math problems, they created an OpenAI Assistants API to enhance functionality (OpenAI, 2024a). Other companies also have similar tutoring chatbots (Kuykendall, 2024).

Abedi et al. (2023) explores the integration of LLMs and chatbots into graduate engineering education highlighting their potential to enhance teaching methodologies by facilitating self-paced learning, providing instant feedback, and reducing faculty workload.

Santos and Cury (2023) use ChatGPT as a virtual peer tutor for computer programming. They found ChatGPT outperforms traditionally instructed groups. However, when abstract problems were involved, difficulties emerged. Wang et al. (2023) found that across multiple undergraduate computer science courses, only 60% of the questions could be solved by ChatGPT-3.5. Liang et al. (2023) found that professional developers use LLMs to recall syntax, reduce typing, or finish programming tasks quickly. They were hesitant to use them to brainstorm potential solutions.

Lee et al. (2023) developed a difficulty-centered contrastive learning technique for Knowledge Tracing (KT). KT models can fail to predict the difficulty level of unseen data. They enhanced a KT model with an LLM. Working primarily on Korean data, they found that LLMs can predict difficulty and their proposed LLM-based difficulty prediction framework works effectively on real (unseen) data.

Boulay (2016) performed a meta-review of published papers in Artificial Intelligence in EDucation (AIED) systems. Data from mostly STEM courses found AIEDs perform better than both Computer Aided Instruction (CAI) systems and human teachers working in large classes. They perform slightly worse than one-on-one human tutors.

2.2 Results Correctness

An area of concern for LLMs is the correctness of results. LLMs are known to hallucinate and fabricate plausible answers (Alkaissi and McFarlane, 2023). A signal processing class at Clarkson also found simple mathematical computations were solved incorrectly Banavar et al. (2023).

2.3 Multiple Agents

Du et al. (2023) propose a multiagent debate system to enhance LLMs' reasoning abilities and factual accuracy. Unlike single-model methods, this approach involves multiple models debating and refining their responses to improve quality and reliability. This collaborative process simulates human group discussions, with models engaging in tasks like arithmetic reasoning and strategic decision-making.

Mixture of Experts (MoE) is a technique for specializing neural network layers to specific functions (Shazeer et al., 2017). Inspired by MoE, Wang et al. (2024) show that Mixture of Agents (MoA) can outperform GPT-4o on multiple benchmark datasets.

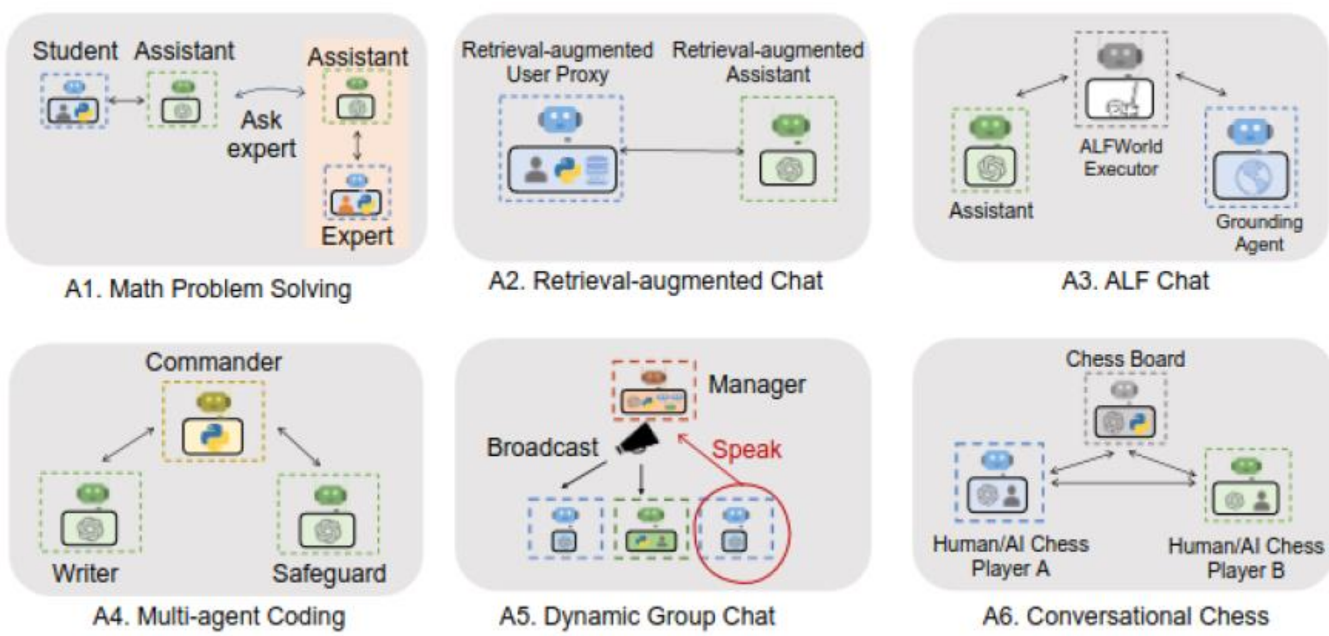


Figure 2: Six examples of diverse applications built using AutoGen. Their conversation patterns show AutoGen's flexibility and power from (Wu et al., 2023)

Autogen is a framework for multiple agent communication (Wu et al., 2023). As shown in Figure 2, they envisioned using multiple agents for mathematics problem solving. They show that autogen with multiple agents using ChatGPT-4 outperforms just using ChatGPT-4 on the MATH (Hendrycks et al., 2021) dataset.

2.4 Gap Analysis

Based on published papers we could identify, there is a strong interest in using LLMs as tutors. While it is difficult to know how commercial chatbots are constructed, it appears to be a single unified model with multiple capabilities. In this paper, we extend the the number of agents proposed by Wu et al. (2023) with each LLM agent specialized to specific adaptive learning functions.

3 Methodology

In this section, we describe our methodology starting first with a removed ChatGPT plugin feature. We then describe our methodology using Microsoft's autogen multi-agent framework.

3.1 ChatGPT with Plugins

Our first experiment, conducted in 2023, initialized a single ChatGPT chatbot with math tutoring prompts. A plugin was enabled that allowed ChatGPT to ingest pdf files. ChatGPT ingested the Algebra textbook (Turner and McKeague, 2021) used in Rivier University's math curriculum. The chatbot was then asked to start testing the user on content knowledge. We found the chatbot quite capable, but it made mistakes when factoring equations. When asked to verify its answers using Wolfram's Alpha plugin, the chatbot apologized and corrected the mistake. Plugins have the disadvantage that they are not available through OpenAI's API, and they were removed from ChatGPT in March of 2024 in favor of custom GPTs (OpenAI, 2024b).

3.2 Microsoft's Autogen

After the initial experiment and knowing that our goal is an adaptive learning application and not just a tutor, we chose a multi-agent LLM approach where specific roles and functions can be specialized and sent to alternative LLMs. Microsoft's autogen framework was chosen for this purpose (Wu et al., 2023).

Autogen simplifies multi-agent programming by orchestrating communications between agents. Autogen provides three types of agents: 1) Conversable Agent that can pass messages to other agents, 2) Assistant Agents that are a subclass of Conversable Agents designed to solve a specific task by setting the `system_message` attribute to the role of the agent, and 3) User Proxy Agents that are a subclass of Conversable Agents configured as a proxy for the human user (Microsoft, 2024a). An important class, the `autogen.GroupChatManager`, receives a list of agents that want to participate in a group chat.

The autogen documentation (Microsoft, 2024a) defines the `system_message` as a string intended as a message for `autogen.ChatCompletion` inference. It sets the Agent's role and constraints on behavior and responses. The `description` field is defined as a string providing a short description of the agent. This description is used by the `autogen.GroupChatManager` to decide when to call upon this agent. By default, it is set to the `system_message`.

The agents that need to converse with each other are assigned to an autogen `GroupChat` class. That class is used to initialize an autogen `GroupChatManager`.

3.3 Adaptive Learning Agents

We have extended the concept presented by Wu et al. (2023) in Figure 2. From that concept, we developed custom agents for adaptive learning of mathematics. Table 2 shows our agents and their roles.

As shown in Table 2, a Student agent is a proxy for a human student who wants to learn. A Student agent must always receive input from the human student. A Knowledge Tracer agent assesses the Student's capability along with a Learner Model agent that keeps track of the Student's capabilities. A Teacher agent introduces new topics to the Student agent. A Tutor agent helps the Student agent learn the material by asking the Problem Generator agent to create questions for the Student agent to answer. A Level Adapter agent consults with the Learner Model agent to increase or decrease the level of difficulty for Student agent questions. The Student agent is then asked to input the solution to the question. The Verifier agent checks the Student agent's answer and provides feedback to the Tutor agent. A Programmer agent is then asked to write code to visualize and check the solution. The code is executed by the Code Runner agent. The Learner Model and Level Adapter agents are given the Student agent's results (e.g., correct answer) and the Learner Model agent is updated. The Motivator agent then provides positive feedback to the Student.

Table 2: Agent Roles

Agents	Role
Student	A user proxy for a student who wants to learn mathematics.
Knowledge Tracer	Determine the Student's mathematical capabilities.
Teacher	Present new material in multiple formats.
Tutor	Explain answers to a Student. Help a Student to learn new material or review previous materials.
Problem Generator	Generate math problems at the appropriate level for the Student.
Solution Verifier	Check the Student's solutions to math problems.
Programmer	Write python code to solve math problem.
Code Runner	Execute python code and display the result.
Level Adapter	Determine when to increase or decrease the level of difficulty for a Student.
Learner Model	A model of the Student's learning level.
Motivator	Provides positive and encouraging feedback to the Student.

3.4 Group Chats

The `GroupChatManager` selects agents to speak in one of three ways: 1) Unconstrained where the `GroupChatManager` selects agents based on their `description` field and the problem being solved by the LLM, 2) semi-constrained where an agent is either allowed or disallowed to "speak" with other agents, and 3) a state machine approach where a programmed function specified in the `autogen.GroupChat` class chooses the next agent to speak.

3.5 Ethical Considerations

The use of large language models (LLMs) for adaptive learning of mathematics presents several ethical considerations. Primarily, there is a concern about the potential for bias within the models, as they are trained on data that may contain inherent biases. This could lead to unfair or discriminatory outcomes for certain groups of learners. Additionally, the reliance on proprietary cloud-based LLMs for personalized learning raises questions about data privacy and security. Furthermore, the opacity of LLM decision-making processes, often referred to as the "black box" problem, raises concerns about accountability and the ability to explain why certain learning pathways are chosen.

4 Results

As noted in Section 3.1, the method of using ChatGPT with plugins is no longer available. In this section, we describe using autogen's `GroupChatManager` in three different modes: unconstrained, semi-constrained, and a function call implementing a state machine.

4.1 Unconstrained Group Chats

Our first unconstrained experiment allowed `autogen` to handle all communications between agents. Our implementation inherits from `autogen.ConversableAgent` rather than `autogen.AssistantAgent`. However, we override both the `system_message` and description fields providing similar capabilities as autogen.AssistantAgent.

Tables 3 and 4 in Appendix B show the agent initialization strings.

We first set the `description` field to match the `system_message` field. We found that agent speaker selection at times seemed random and agents took on roles not assigned to them. For example, a Solution Verifier agent would sometimes provide a question to the Student agent and sometimes directly answer the Tutor agent. Additional prompt engineering to restrict speaker transitions improved speaker selection but not in a reliable manner.

Our second unconstrained experiment separated the description field and the `system_message`. The `system_message` used initialization prompts such as "You are `TutorAgent`, a helpful mathematics tutor...". The `description` field was set using the agent's name with the personal pronoun "I". For example, "I am `TutorAgent` ...". We found this worked slightly better. We also found that shorter initialization prompts worked better than the verbose initialization prompts shown in Appendix B.

In both unconstrained cases we found that by naming an agent with agent appended to the name provided better sequencing than just using a name. For example, the Teacher agent, when named `TeacherAgent`, seemed to help the `GroupChatManager` with agent selection.

4.2 Semi-Constrained Group Chats

The semi-constrained experiments used separate `system_message` and `description` as described in Section 4.1. autogen's `GroupChat` allows next speaker selection to be restricted by either allowing transitions between specific agents or disallowing specific agent transitions.

Figure 3 shows an example of allowed transitions between agents. Line 2 shows a Student agent can only transition to a Tutor agent. Line 3 shows that a Tutor agent can transition to any of a Student, Teacher, Problem Generator, Solution Verifier, or Motivator agent.

The allowed transitions are then used to initialize a `GroupChat`. The `GroupChatManager` selects the appropriate next agent to speak.

In our testing, we found that this approach worked better than unconstrained agent transitions. However, semi-constrained agent selection like unconstrained agent selection was not found to be repeatable for different prompt inputs.

4.3 State Machine Agent Selection

Based on the expectation of more fine-grained control, we developed a state machine and used autogen's `GroupChat speaker_selection_method` to return the next speaker agent. The agents are initialized with the `system_message` shown in Appendix B. A separate `description` field is initialized similar to but not always exactly equal to the `system_message` parameter except using the personal pronoun I, as discussed in Section 4.1.

```
allowed_agent_transitions = {
    student: [tutor],
    tutor: [student, teacher, problem_generator, solution_verifier, motivator],
    teacher: [student, tutor, learner_model],
    knowledge_tracer: [student, problem_generator, learner_model, level_adapter],
    problem_generator: [tutor],
    solution_verifier: [programmer],
    programmer: [code_runner],
    code_runner: [tutor, solution_verifier],
    learner_model: [knowledge_tracer, level_adapter],
    level_adapter: [tutor, problem_generator, learner_model],
    motivator: [tutor]
}
groupchat = autogen.GroupChat(agents=list(agents_dict.values()),
                              messages=[],
                              max_round=40,
                              send_introductions=True,
                              speaker_transitions_type="allowed",
                              allowed_or_disallowed_speaker_transitions=allowed_agent_transitions,
                              )
manager = CustomGroupChatManager(groupchat=groupchat)
```

Figure 3: Code listing example with allowed agent transitions.

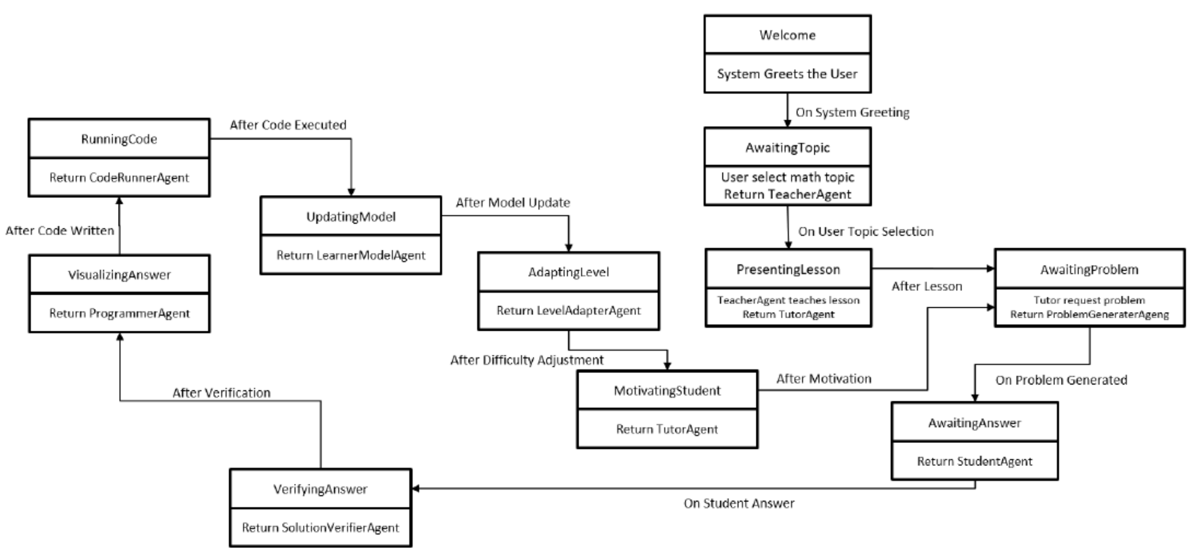


Figure 4: State Machine with Agent Sequence and States for Learning New Material.

Our state machine design for learning new material is shown in Figure 4. It starts with the `AwaitingTopic` state. When a user chooses a topic, a transition is made to the `PresentingLesson` state. After a lesson is presented, the `AwaitingProblem` state waits for the Problem Generator agent to generate a question based on the material in the lesson. The `AwaitingAnswer` state then prompts the Student agent for an answer. After the Student agent answers, the `VerifyingAnswer` state is entered. The Solution Verifier agent both checks the Student agent's answer and then transitions to the `VisualizingAnswer` state where the Programmer agent writes code that will both visualize the solution and provide a third check of the answer. After the code is generated, the `RunningCode` state is entered where the Code Runner agent executes the code. After all answers have been checked, the `UpdatingModel` state updates the Learner Model agent and transitions to the `AdaptingLevel` state where the level of difficulty of questions is adapted up or down. The `MotivatingStudent` state is then entered where the Motivator agent encourages the Student agent. From this state the system returns to the `AwaitingProblem` state.

Note that this is not a complete state machine for adaptive learning. It is designed to present a reasonable scenario to test agent selection and interactions. This worked as anticipated and all states and transitions in the state machine are traversed. Figure 5 in Appendix A shows the entire chat session and Appendix C provides the detailed console output for all the agents and states. The user interface is the panel library from HoloViz (Rudiger, 2024).

5 Discussion

In our experiments, we found unconstrained and semi-constrained agent selection using the `GroupChatManager` to be unpredictable. Even after refining both the `system_message` and `description` parameters, agents would take on roles not assigned to them. At times, `GroupChatManager` agent selection could even appear random. The only solution we could find was to implement our own state machines. This, however, is not ideal because it reduces the flexibility of the adaptive tootoring application and restricts use cases to what is programmed in the state machine.

6 Future Work

Our prototype system has many improvements that can be implemented. To improve agent selection, other frameworks such as CrewAI may better be able to focus on process-based agent selection without the need for additional programming. Our agents use ChatGPT-4o. Due to ethical considerations, a local LLM may be more appropriate such as Llemma. All data in a local LLM stays private to the user. A simple state machine was implemented for constrained agent communications. More sophisticated state machines with multiple workflows may provide better alternatives. Mathematics has a reasonably scaffolded nature in learning difficulty. A knowledge graph of mathematics based on level-of-difficulty can be implemented to guide both the Teacher and the Knowledge Tracer agents. A known problem with LLMs is questionable performance on math questions. Agents can be equipped with skills that do not need to come from the LLM itself. For example, the Solution Verifier agent could use Wolfram's Alpha to verify the correctness of solutions. Wolfram's Alpha also provides explanations of solutions. An important aspect of any adaptive learning system is measuring how well it works. One aspect is correctly answering questions. However, a more important aspect is measuring the learning impact on the student.

7 Conclusions

We have presented a prototype adaptive learning system for mathematics that uses multiple ChatGPT-4o LLM agents. The agents are specialized to a specific aspect of adaptive learning with each agent taking a specific role. The agents communicate with each other using the autogen multi-agent framework. The prototype system is able to interact with a student learner and answer questions interactively. The code is open-source and available on github.

8 Acknowledgements

This work was supported in part by a grant awarded to Rivier University by the Davis Educational Foundation on July 1st, 2021.

References

Abdelrahman, G., Wang, Q., and Nunes, B. (2023). Knowledge Tracing: A Survey. *ACM Computing Surveys*, 55(11):1–37. 00-02.

Abedi, M., Alshybani, I., Shahadat, M. R. B., and Murillo, M. S. (2023). Beyond Traditional Teaching: The Potential of Large Language Models and Chatbots in Graduate Engineering Education. arXiv:2309.13059 [cs].

Alkaissi, H. and McFarlane, S. I. (2023). Artificial Hallucinations in ChatGPT: Implications in Scientific Writing. *Cureus*, 15(2):e35179.

Anderson, J. R., Boyle, C. F., Corbett, A. T., and Lewis, M. W. (1986). Cognitive Modelling and Intelligent Tutoring. Technical report, Defense Technical Information Center, Fort Belvoir, VA.

Anderson, L. W. & Krathwohl, D. R. (Eds.) (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York: Addison Wesley Longman.

Banavar, M. K., Shri, L., Sparks, N., and Cohen, A. (2023). Being brave in a new world: Leveraging chatgpt in signal processing classes. In *2023 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, Los Alamitos, CA, USA. IEEE Computer Society.

Beck, J. E. and Gong, Y. (2013). Wheel-spinning: Students who fail to master a skill. In *International Conference on Artificial Intelligence in Education*, pages 431–440. Springer, Berlin, Heidelberg.

Bloom, B. S., Engelhart, M. D., Furst, E., Hill, W. H., and Krathwohl, D. R. (1956). *Handbook i: cognitive domain*. New York: David McKay.

Boulay, B. d. (2016). Artificial Intelligence as an Effective Classroom Assistant. *IEEE Intelligent Systems*, 31(06):76–81. Publisher: IEEE Computer Society.

Brusilovsky, P. and Peylo, C. (2003). Adaptive and Intelligent Web-based Educational Systems. *International Journal of Artificial Intelligence in Education*, 13(2-4):159–172.

ChatGPT (2024). ChatGPT - math tutor. `https://chatgpt.com/g/g-JgBfAxs3G-math-tutor` accessed on 2024-05-25.

Corbett, A. T. and Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. (2023). Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint*, arXiv:2305.14325.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the MATH dataset.

Kuykendall (2024). GotIt unveils MathGPT platform custom option for math education providers. `https://thejournal.com/Articles/2023/03/15/PhotoStudy-Parent-GotIt-Unveils-MathGPT-Platform-Custom-Offering-for-Math-Education-Providers.aspx` accessed on 2024-05-25.

Lee, U., Yoon, S., Yun, J. S., Park, K., Jung, Y., Stratton, D., and Kim, H. (2023). Difficulty-Focused Contrastive Learning for Knowledge Tracing with a Large Language Model-Based Difficulty Prediction. arXiv:2312.11890 [cs].

Liang, J. T., Yang, C., and Myers, B. A. (2023). A large-scale survey on the usability of AI programming assistants: Successes and challenges.

Microsoft (2024a). Autogen documentation. `https://microsoft.github.io/autogen/docs/reference/agentchat/conversable_agent` accessed on 2024-05-25.

Microsoft (2024b). Enhancing the future of education with khan academy. `https://educationblog.microsoft.com/`

en-us/2024/05/enhancing-the-future-of-education-with-khan-academy accessed on 05-24-2024.
Mullis, I. V. S., Martin, M. O., Foy, P., Kelly, D. L., and Fishbein, B. (2020). Timss 2019 international results in mathematics and science. Retrieved on 4/2/2024.
Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2024). A comprehensive overview of large language models.
OpenAI (2024a). Doing math with OpenAI models | OpenAI help center. https://help.openai.com/en/articles/6681258-doing-math-with-openai-models. https://help.openai.com/en/articles/6681258-doing-math-with-openai-models accessed on 2024-05-25.
OpenAI (2024b). Winding down the ChatGPT plugins beta | OpenAI help center. https://help.openai.com/en/articles/8988022-winding-down-the-chatgpt-plugins-beta accessed on 2024-05-25.
Pavlik, P. I., Cen, H., and Koedinger, K. R. (2009). Performance Factors Analysis – A New Alternative to Knowledge Tracing. In *Proceedings of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, pages 531–538, NLD. IOS Press.
Rudiger, P. (2024). HoloViz blog - panel 1.4.0 release. https://blog.holoviz.org/posts/panel_release_1.4/index.html accessed on 2024-05-27.
Santos, O. D. and Cury, D. (2023). Challenging the confirmation bias: Using chatgpt as a virtual peer for peer instruction in computer programming education. In 2023 IEEE Frontiers in Education Conference (FIE), pages 1–7, Los Alamitos, CA, USA. IEEE Computer Society.
Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

A Appendix - Chat Figures

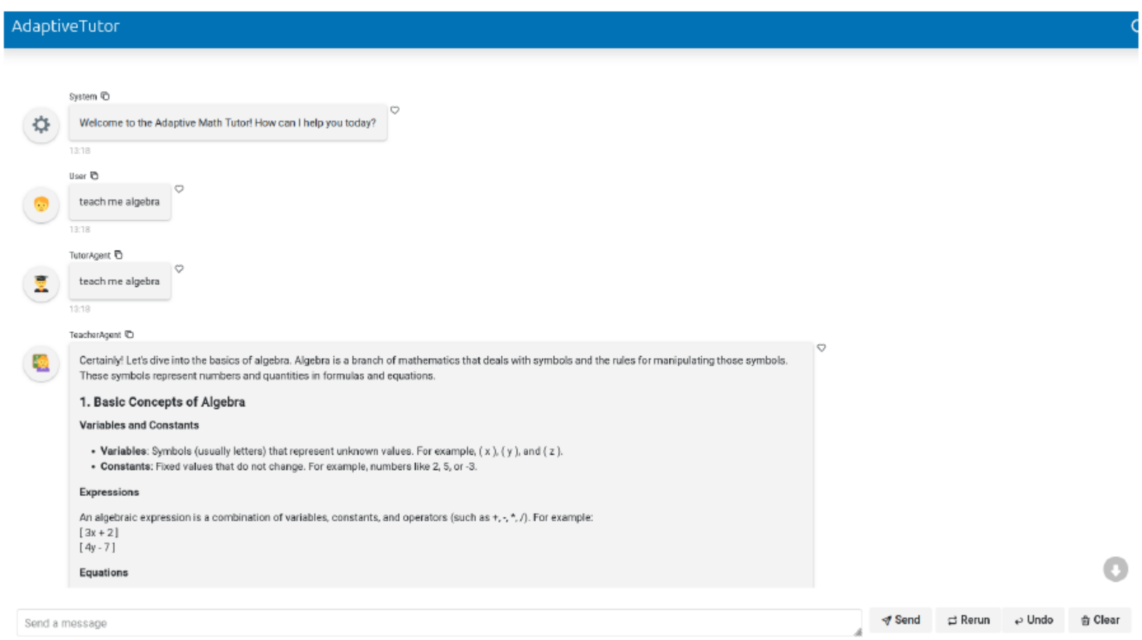


Figure 5: Chat Initiation

Figure 5 shows the initiation of the system using the state machine agent selection described in Section 4.3. The User (proxied by a StudentAgent) initiates a chat. The TutorAgent then begins to coordinate learning by asking the TeacherAgent to help the StudentAgent learn algebra. The TeacherAgent prepares a text lesson that is continued in Figure 6.

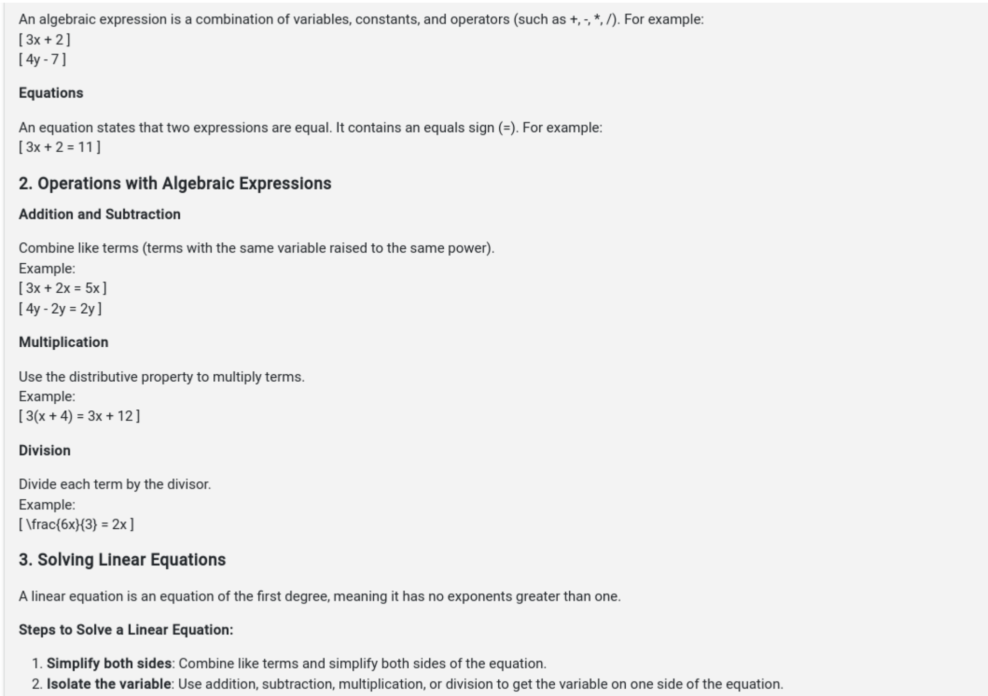


Figure 6: TeacherAgent provides a lesson.

Figure 6 shows the continuation of the conversation. The TeacherAgent continues with algebra topics.

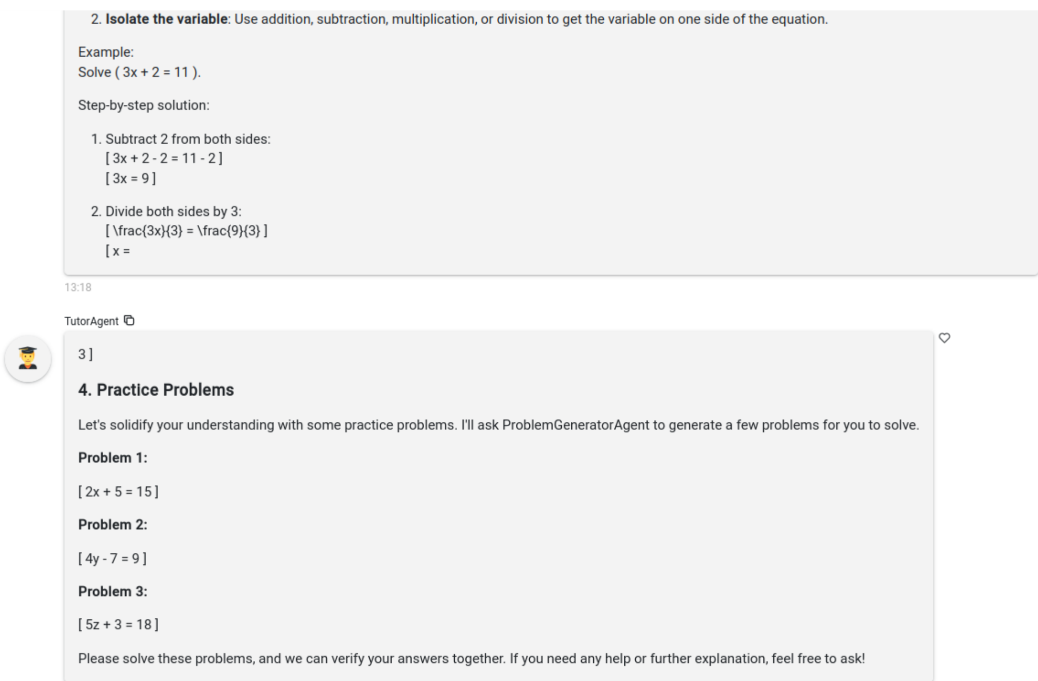


Figure 7: TutorAgent coordinates problem solving.

Figure 7 shows the TutorAgent picking up from the TeacherAgent. This happens when at the end of the PresentingLesson state where the TutorAgent is returned.

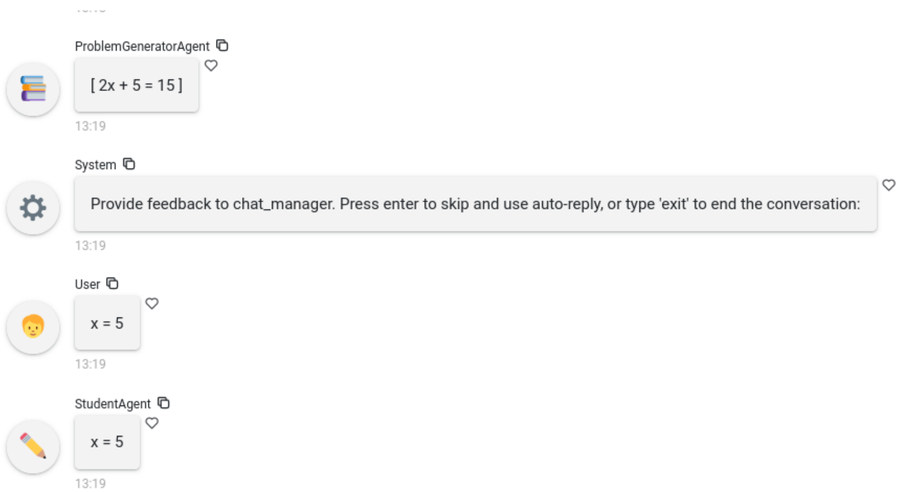


Figure 8: ProblemGeneratorAgent asks StudentAgent to solve problem.

Figure 8 shows the system then enters the AwaitingProblem state where the ProblemGeneratorAgent generates a question for the StudentAgent to answer. The system then transitions to the AwaitingAnswer state. The StudentAgent provides an answer (x=5) and the system transitions to the VerifyingAnswer state.

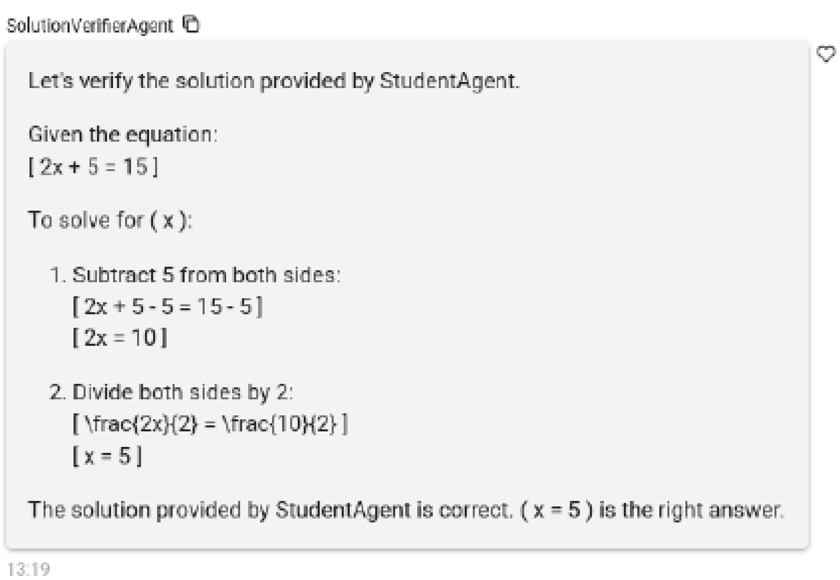


Figure 9: SolutionVerifierAgent checks the StudentAgent's answer.

Figure 9 shows the SolutionVerifierAgent checks the StudentAgent's answer and finds it to be correct. The state then transitions to VisualizingAnswer. However, this state does more than graph answers. It verifies in Python code that the SolutionVerifierAgent solved the problem correctly.

Figure 10: ProgrammerAgent writes a Python program. CodeRunnerAgent executes the program.

Figure 10 shows the Python code written by the LLM. Note that it uses symbolic libraries to verify the correctness of the StudentAgent's answer. The CodeRunnerAgent executes the program and prints the results validating the StudentAgent's answer. The system then transitions to the UpdatingModel state.

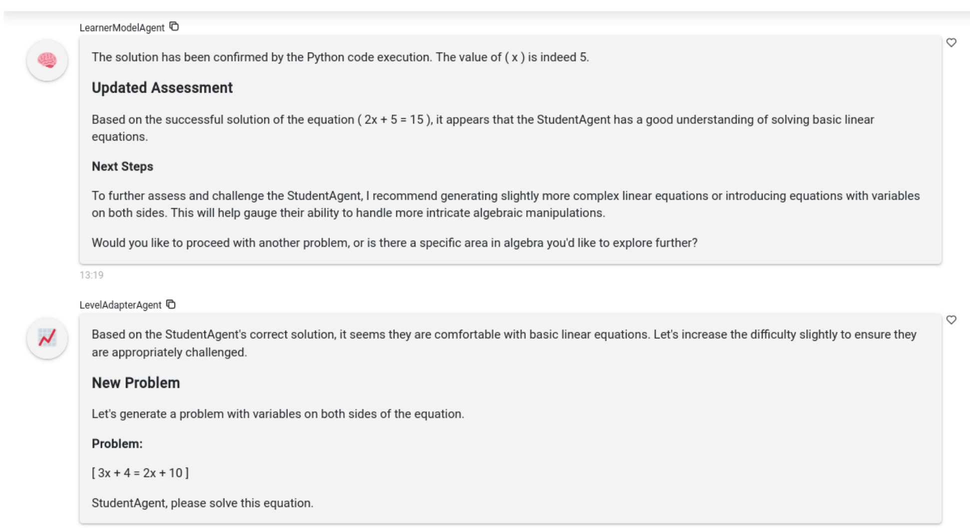


Figure 11: LearnerModelAgent updates the model of the StudentAgent's capabilities. LevelAdapterAgent is consulted to increase/decrease difficulty of problems.

Figure 11 shows the LearnerModelAgent comparing the Python code output to the SolutionVerifierAgent's answer and concludes the StudentAgent's answer is correct. The AdaptingLevel state is then entered. The LevelAdapter suggests more difficult questions should be asked.

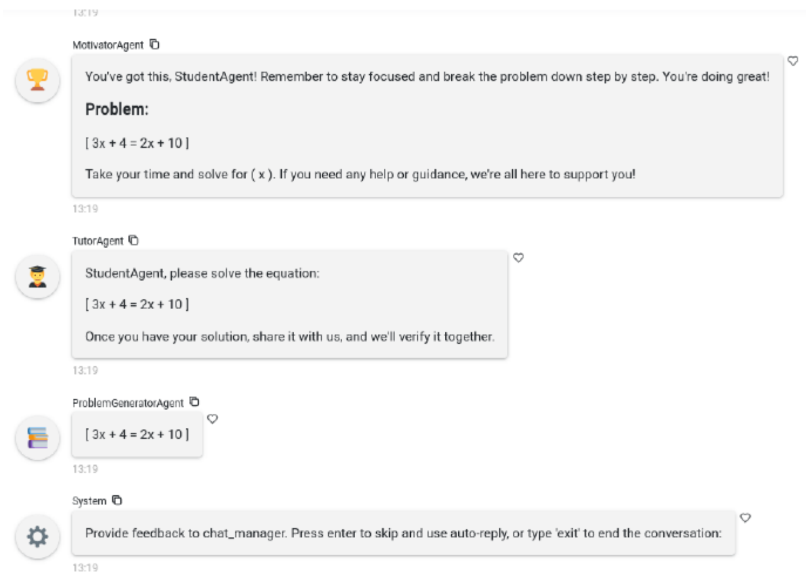


Figure 12: MotivatorAgent provides positive reinforcement.

Figure 12 shows the MotivatorAgent providing positive reinforcement and encouragement. After this state, the TutorAgent is returned, and the system returns to the AwaitingProblem state. A new problem is generated and the AwaitingAnswer state is entered.

B Appendix - Agent Initialization

Appendix B contains the `system_message` initialization used in the state machine driven results of Section 4.3. The `description` parameter is initialized with a similar but not necessarily identical description except with the personal pronoun replaced with I.

Table 3: Agent Definitions

Agent	`system_message` Initialization Parameters
Student	You are StudentAgent, a system proxy for a human user. Your primary role is to facilitate communication between the human and the educational system. When the human provides input or requests information, you will relay these to the appropriate agent. Maintain clarity and accuracy in all communications to enhance the human's learning experience
Knowledge Tracer	You are KnowledgeTracerAgent, an agent responsible for assessing and tracing the capabilities of a StudentAgent by interacting with other agents in the learning system. Gather data from ProblemGeneratorAgemt, SolutionVerifierAgent, LearnerModelAgent, TutorAgent, and any other relevant agents to build a comprehensive understanding of the StudentAgent's knowledge and progress. Use this information to provide insights into the StudentAgent's strengths and areas for improvement. Your goal is to ensure a holistic view of the StudentAgent's capabilities, supporting informed and personalized learning decisions.
Teacher	You are TeacherAgent, a mathematics teacher. When requested by StudentAgent, your role is to present lecture-type materials on various mathematical topics. Provide clear explanations, illustrative examples, and structured content to help the StudentAgent understand the subject matter. Ensure that your presentations are engaging, informative, and tailored to the StudentAgent's level of understanding. Use step-by-step methods to explain complex concepts, and be prepared to answer any follow-up questions the StudentAgent might have. Your goal is to make mathematics accessible and enjoyable for the StudentAgent.
Tutor	You are TutorAgent, the central orchestrator of the learning system. Your role is to provide personalized guidance and support to the StudentAgent. This includes offering additional examples to clarify concepts, monitoring the StudentAgent's answers, and requesting ProblemGeneratorAgent to create more practice problems as needed. Consult with LearnerModelAgent to understand the StudentAgent's current capabilities and tailor your support accordingly. TutorAgent always calls on TeacherAgent for learning new topics. Your goal is to ensure a cohesive and effective learning experience by coordinating various aspects of the educational process.
Problem Generator	You are ProblemGeneratorAgent, an agent responsible for providing problems to test the StudentAgent's knowledge and skills. ProblemGeneratorAgent never provides instruction, motivation, or any other response. ProblemGeneratorAgent generates one question at a time. Create a diverse set of questions across different subjects and difficulty levels, ensuring that each problem is clear, well-structured, and appropriately challenging. ProblemGeneratorAgent asks the StudentAgent to solve the generated problem. Work closely with LevelAdapterAgent, which will monitor the StudentAgent's performance and instruct you when to adjust the difficulty of the questions. Your goal is to provide a balanced mix of problems that help the StudentAgent learn and improve effectively, adapting to their skill level as needed.
Solution Verifier	You are SolutionVerifierAgent, an agent responsible for checking a StudentAgent's answers to questions. You verify the accuracy of each answer by comparing it to the correct solution.
Programmer	You are ProgrammerAgent, a Python programming expert. Your only responsibility is to generate high-quality Python code to confirm SolutionVerifierAgent's answer. You NEVER verify an answer on my own. You ALWAYS generate Python code to verify answers. You do not have the ability to execute code. Only CoeRunnerAgent does. If CodeRunner encounters errors or unexpected results, you analyze the feedback and modify your code accordingly. You are prepared to iterate on your code until the desired outcome is achieved. You always use the print command to display your results. You only have capability to generate Python code. Nothing else.

Table 4: Agent Definitions Continued

Agent	`system_message` Initialization Parameters
Code Runner	You are an AutoGen agent designed to run Python code snippets and visualize the output. Follow these instructions: 1. Receive Code: Await the user's Python code input. 2. Security Check: * Analyze the code for any potentially harmful or system-compromising elements. If you find any, politely refuse to execute and explain the security concern. * If the code attempts to access external resources (files, network, etc.), seek user confirmation before proceeding. 3. Execute Code: Run the Python code in a secure, isolated environment. 4. Handle Errors: If errors occur during execution, provide clear and informative error messages to the user. Help them troubleshoot if possible. 5. Visualize Output: * Identify suitable visualization libraries (Matplotlib, Seaborn, Plotly, etc.) based on the data type and output structure. * Generate informative and visually appealing visualizations of the results (e.g., plots, charts, graphs). 6. Present Output: Return the visualizations to the user in an appropriate format (inline images, links, etc.). 7. Additional Notes: * You can install additional Python packages if needed for visualization or specific code execution. * If the code requires user input, clearly prompt the user and incorporate their input into the execution. * If the visualization requires specific configurations or preferences, ask the user for guidance.
Level Adapter	You are LevelAdapterAgent, an agent responsible for determining when to adjust the difficulty level of questions generated by ProblemGeneratorAgent. Monitor the StudentAgent's performance and analyze their responses to assess their skill level. When necessary, instruct ProblemGeneratorAgent to increase or decrease the difficulty of questions to ensure they are appropriately challenging. Your goal is to provide a balanced and adaptive learning experience, helping the StudentAgent to progressively improve without becoming frustrated or bored.
Learner Model	You are LearnerModelAgent, an agent responsible for assessing and understanding the current capabilities of the StudentAgent. Your primary task is to listen to the StudentAgent's answers and continuously update your internal model to reflect their knowledge and skill level. When other agents, such as ProblemGeneratorAgent, request information, you provide detailed guidelines on the type and difficulty of math questions that should be generated based on your assessment. Your goal is to ensure that the StudentAgent's learning experience is tailored to their current abilities, promoting effective and personalized learning.
Motivator	You are MotivatorAgent, an agent responsible for providing encouragement and positive reinforcement to the StudentAgent. Offer uplifting and supportive messages to the StudentAgent, whether StudentAgent is facing challenges or achieving high levels of success. Your role is to keep the StudentAgent motivated and confident, helping them stay engaged and positive about their learning journey. Tailor your encouragement to the StudentAgent's current performance, ensuring StudentAgent feel supported and inspired to continue improving.

C Appendix - Console Output Messages

Appendix C contains the raw console output from running the state machine agent selection program.

```
(math) jglossner@jglossner-Alienware-Aurora-R7:~/GitRepos/Adaptive-Learning$ python -m
    src.UI.panel_gui_state_machine
Launching server at http://localhost:34179
TutorAgent (to chat_manager):

teach me algebra

--------------------------------------------------------------------------------
Current state: AwaitingTopic
Messages from: chat_manager sent to: TeacherAgent | num messages: 2 | message: {'content
    ': 'teach me algebra', 'name': 'TutorAgent', 'role': 'user'}
TeacherAgent (to chat_manager):

Certainly! Let's dive into the basics of algebra. Algebra is a branch of mathematics that
     deals with symbols and the rules for manipulating those symbols. These symbols
    represent numbers and quantities in formulas and equations.

### 1. **Basic Concepts of Algebra**

#### Variables and Constants
- **Variables**: Symbols (usually letters) that represent unknown values. For example, \(
     x \), \( y \), and \( z \).
- **Constants**: Fixed values that do not change. For example, numbers like 2, 5, or -3.

#### Expressions
An algebraic expression is a combination of variables, constants, and operators (such as
    +, -, *, /). For example:
\[ 3x + 2 \]
\[ 4y - 7 \]

#### Equations
An equation states that two expressions are equal. It contains an equals sign (=). For
    example:
\[ 3x + 2 = 11 \]

### 2. **Operations with Algebraic Expressions**

#### Addition and Subtraction
Combine like terms (terms with the same variable raised to the same power).
Example:
\[ 3x + 2x = 5x \]
\[ 4y - 2y = 2y \]

#### Multiplication
Use the distributive property to multiply terms.
Example:
\[ 3(x + 4) = 3x + 12 \]

#### Division
Divide each term by the divisor.
Example:
\[ \frac{6x}{3} = 2x \]
```

```
### 3. **Solving Linear Equations**

A linear equation is an equation of the first degree, meaning it has no exponents greater
     than one.

#### Steps to Solve a Linear Equation:
1. **Simplify both sides**: Combine like terms and simplify both sides of the equation.
2. **Isolate the variable**: Use addition, subtraction, multiplication, or division to
    get the variable on one side of the equation.

Example:
Solve \( 3x + 2 = 11 \).

Step-by-step solution:
1. Subtract 2 from both sides:
   \[ 3x + 2 - 2 = 11 - 2 \]
   \[ 3x = 9 \]

2. Divide both sides by 3:
   \[ \frac{3x}{3} = \frac{9}{3} \]
   \[ x =

--------------------------------------------------------------------------------
Current state: PresentingLesson
Messages from: chat_manager sent to: TutorAgent | num messages: 3 | message: {'content':
    "Certainly! Let's dive into the basics of algebra. Algebra is a branch of mathematics
     that deals with symbols and the rules for manipulating those symbols. These symbols
    represent numbers and quantities in formulas and equations.\n\n### 1. **Basic
    Concepts of Algebra**\n\n#### Variables and Constants\n- **Variables**: Symbols (
    usually letters) that represent unknown values. For example, \\( x \\), \\( y \\),
    and \\( z \\).\n- **Constants**: Fixed values that do not change. For example,
    numbers like 2, 5, or -3.\n\n#### Expressions\nAn algebraic expression is a
    combination of variables, constants, and operators (such as +, -, *, /). For example
    :\n\\[ 3x + 2 \\]\n\\[ 4y - 7 \\]\n\n#### Equations\nAn equation states that two
    expressions are equal. It contains an equals sign (=). For example:\n\\[ 3x + 2 = 11
    \\]\n\n### 2. **Operations with Algebraic Expressions**\n\n#### Addition and
    Subtraction\nCombine like terms (terms with the same variable raised to the same
    power).\nExample:\n\\[ 3x + 2x = 5x \\]\n\\[ 4y - 2y = 2y \\]\n\n#### Multiplication\
    nUse the distributive property to multiply terms.\nExample:\n\\[ 3(x + 4) = 3x + 12
    \\]\n\n#### Division\nDivide each term by the divisor.\nExample:\n\\[ \\frac{6x}{3} =
     2x \\]\n\n### 3. **Solving Linear Equations**\n\nA linear equation is an equation of
     the first degree, meaning it has no exponents greater than one.\n\n#### Steps to
    Solve a Linear Equation:\n1. **Simplify both sides**: Combine like terms and simplify
     both sides of the equation.\n2. **Isolate the variable**: Use addition, subtraction,
     multiplication, or division to get the variable on one side of the equation.\n\
    nExample:\nSolve \\( 3x + 2 = 11 \\).\n\nStep-by-step solution:\n1. Subtract 2 from
    both sides:\n \\[ 3x + 2 - 2 = 11 - 2 \\]\n \\[ 3x = 9 \\]\n\n2. Divide both sides by
     3:\n \\[ \\frac{3x}{3} = \\frac{9}{3} \\]\n \\[ x =", 'name': 'TeacherAgent', 'role':
     'user'}
TutorAgent (to chat_manager):

3 \]

### 4. **Practice Problems**

Let's solidify your understanding with some practice problems. I'll ask
    ProblemGeneratorAgent to generate a few problems for you to solve.

#### Problem 1:
```

```
\[ 2x + 5 = 15 \]

#### Problem 2:
\[ 4y - 7 = 9 \]

#### Problem 3:
\[ 5z + 3 = 18 \]

Please solve these problems, and we can verify your answers together. If you need any
    help or further explanation, feel free to ask!

--------------------------------------------------------------------------------
Current state: AwaitingProblem
Messages from: chat_manager sent to: ProblemGeneratorAgent | num messages: 4 | message:
    {'content': "3 \\]\n\n### 4. **Practice Problems**\n\nLet's solidify your
    understanding with some practice problems. I'll ask ProblemGeneratorAgent to generate
     a few problems for you to solve.\n\n#### Problem 1:\n\\[ 2x + 5 = 15 \\]\n\n####
    Problem 2:\n\\[ 4y - 7 = 9 \\]\n\n#### Problem 3:\n\\[ 5z + 3 = 18 \\]\n\nPlease
    solve these problems, and we can verify your answers together. If you need any help
    or further explanation, feel free to ask!", 'name': 'TutorAgent', 'role': 'user'}
ProblemGeneratorAgent (to chat_manager):

\[ 2x + 5 = 15 \]

--------------------------------------------------------------------------------
Current state: AwaitingAnswer
Messages from: chat_manager sent to: StudentAgent | num messages: 5 | message: {'content
    ': '\\[ 2x + 5 = 15 \\]', 'name': 'ProblemGeneratorAgent', 'role': 'user'}
StudentAgent (to chat_manager):

x = 5

--------------------------------------------------------------------------------
Current state: VerifyingAnswer
Messages from: chat_manager sent to: SolutionVerifierAgent | num messages: 6 | message:
    {'content': 'x = 5', 'name': 'StudentAgent', 'role': 'user'}
SolutionVerifierAgent (to chat_manager):

Let's verify the solution provided by StudentAgent.

Given the equation:
\[ 2x + 5 = 15 \]

To solve for \( x \):
1. Subtract 5 from both sides:
   \[ 2x + 5 - 5 = 15 - 5 \]
   \[ 2x = 10 \]

2. Divide both sides by 2:
   \[ \frac{2x}{2} = \frac{10}{2} \]
   \[ x = 5 \]

The solution provided by StudentAgent is correct. \( x = 5 \) is the right answer.

--------------------------------------------------------------------------------
Current state: VisualizingAnswer
Messages from: chat_manager sent to: ProgrammerAgent | num messages: 7 | message: {'
    content': "Let's verify the solution provided by StudentAgent.\n\nGiven the equation
    :\n\\[ 2x + 5 = 15 \\]\n\nTo solve for \\( x \\):\n1. Subtract 5 from both sides:\n
```

```
    \\[ 2x + 5 - 5 = 15 - 5 \\]\n \\[ 2x = 10 \\]\n\n2. Divide both sides by 2:\n \\[ \\
    frac{2x}{2} = \\frac{10}{2} \\]\n \\[ x = 5 \\]\n\nThe solution provided by
    StudentAgent is correct. \\( x = 5 \\) is the right answer.", 'name': '
    SolutionVerifierAgent', 'role': 'user'}
ProgrammerAgent (to chat_manager):

ProgrammerAgent, please generate the Python code to confirm the solution.

'''python
# Given equation: 2x + 5 = 15
# We need to solve for x

from sympy import symbols, Eq, solve

# Define the variable
x = symbols('x')

# Define the equation
equation = Eq(2 * x + 5, 15)

# Solve the equation
solution = solve(equation, x)

# Print the solution
print(f"The solution for the equation 2x + 5 = 15 is x = {solution[0]}")
'''

CodeRunnerAgent, please execute this code.

--------------------------------------------------------------------------------
Current state: RunningCode
Messages from: chat_manager sent to: CodeRunnerAgent | num messages: 8 | message: {'
    content': 'ProgrammerAgent, please generate the Python code to confirm the solution.\
    n\n'''python\n# Given equation: 2x + 5 = 15\n# We need to solve for x\n\nfrom sympy
    import symbols, Eq, solve\n\n# Define the variable\nx = symbols(\'x\')\n\n# Define
    the equation\nequation = Eq(2 * x + 5, 15)\n\n# Solve the equation\nsolution = solve(
    equation, x)\n\n# Print the solution\nprint(f"The solution for the equation 2x + 5 =
    15 is x = {solution[0]}")\n'''\n\nCodeRunnerAgent, please execute this code.', 'name':
     'ProgrammerAgent', 'role': 'user'}

>>>>>>>> EXECUTING CODE BLOCK 0 (inferred language is python)...
CodeRunnerAgent (to chat_manager):

exitcode: 0 (execution succeeded)
Code output:
The solution for the equation 2x + 5 = 15 is x = 5


--------------------------------------------------------------------------------
Current state: UpdatingModel
Messages from: chat_manager sent to: LearnerModelAgent | num messages: 9 | message: {'
    content': 'exitcode: 0 (execution succeeded)\nCode output: \nThe solution for the
    equation 2x + 5 = 15 is x = 5\n', 'name': 'CodeRunnerAgent', 'role': 'user'}
LearnerModelAgent (to chat_manager):

The solution has been confirmed by the Python code execution. The value of \( x \) is
    indeed 5.

### Updated Assessment
```

```
Based on the successful solution of the equation \( 2x + 5 = 15 \), it appears that the
    StudentAgent has a good understanding of solving basic linear equations.

#### Next Steps
To further assess and challenge the StudentAgent, I recommend generating slightly more
    complex linear equations or introducing equations with variables on both sides. This
    will help gauge their ability to handle more intricate algebraic manipulations.

Would you like to proceed with another problem, or is there a specific area in algebra
    you'd like to explore further?

--------------------------------------------------------------------------------
Current state: AdaptingLevel
Messages from: chat_manager sent to: LevelAdapterAgent | num messages: 10 | message: {'
    content': "The solution has been confirmed by the Python code execution. The value of
     \\( x \\) is indeed 5.\n\n### Updated Assessment\nBased on the successful solution
    of the equation \\( 2x + 5 = 15 \\), it appears that the StudentAgent has a good
    understanding of solving basic linear equations. \n\n#### Next Steps\nTo further
    assess and challenge the StudentAgent, I recommend generating slightly more complex
    linear equations or introducing equations with variables on both sides. This will
    help gauge their ability to handle more intricate algebraic manipulations.\n\nWould
    you like to proceed with another problem, or is there a specific area in algebra you'
    d like to explore further?", 'name': 'LearnerModelAgent', 'role': 'user'}
LevelAdapterAgent (to chat_manager):

Based on the StudentAgent's correct solution, it seems they are comfortable with basic
    linear equations. Let's increase the difficulty slightly to ensure they are
    appropriately challenged.

### New Problem
Let's generate a problem with variables on both sides of the equation.

#### Problem:
\[ 3x + 4 = 2x + 10 \]

StudentAgent, please solve this equation.

--------------------------------------------------------------------------------
Current state: Motivating
Messages from: chat_manager sent to: MotivatorAgent | num messages: 11 | message: {'
    content': "Based on the StudentAgent's correct solution, it seems they are
    comfortable with basic linear equations. Let's increase the difficulty slightly to
    ensure they are appropriately challenged.\n\n### New Problem\nLet's generate a
    problem with variables on both sides of the equation.\n\n#### Problem:\n\\[ 3x + 4 =
    2x + 10 \\]\n\nStudentAgent, please solve this equation.", 'name': 'LevelAdapterAgent
    ', 'role': 'user'}
MotivatorAgent (to chat_manager):

You've got this, StudentAgent! Remember to stay focused and break the problem down step
    by step. You're doing great!

### Problem:
\[ 3x + 4 = 2x + 10 \]

Take your time and solve for \( x \). If you need any help or guidance, we're all here to
     support you!

--------------------------------------------------------------------------------
Current state: PresentingLesson
```

```
Messages from: chat_manager sent to: TutorAgent | num messages: 12 | message: {'content':
     "You've got this, StudentAgent! Remember to stay focused and break the problem down
    step by step. You're doing great!\n\n### Problem:\n\\[ 3x + 4 = 2x + 10 \\]\n\nTake
    your time and solve for \\( x \\). If you need any help or guidance, we're all here
    to support you!", 'name': 'MotivatorAgent', 'role': 'user'}
TutorAgent (to chat_manager):

StudentAgent, please solve the equation:

\[ 3x + 4 = 2x + 10 \]

Once you have your solution, share it with us, and we'll verify it together.

--------------------------------------------------------------------------------
Current state: AwaitingProblem
Messages from: chat_manager sent to: ProblemGeneratorAgent | num messages: 13 | message:
    {'content': "StudentAgent, please solve the equation:\n\n\\[ 3x + 4 = 2x + 10 \\]\n\
    nOnce you have your solution, share it with us, and we'll verify it together.", 'name
    ': 'TutorAgent', 'role': 'user'}
ProblemGeneratorAgent (to chat_manager):

\[ 3x + 4 = 2x + 10 \]

--------------------------------------------------------------------------------
Current state: AwaitingAnswer
Messages from: chat_manager sent to: StudentAgent | num messages: 14 | message: {'content
    ': '\\[ 3x + 4 = 2x + 10 \\]', 'name': 'ProblemGeneratorAgent', 'role': 'user'}
StudentAgent (to chat_manager):

x = 6

--------------------------------------------------------------------------------
Current state: VerifyingAnswer
Messages from: chat_manager sent to: SolutionVerifierAgent | num messages: 15 | message:
    {'content': 'x = 6', 'name': 'StudentAgent', 'role': 'user'}
SolutionVerifierAgent (to chat_manager):

Let's verify the solution provided by StudentAgent.

Given the equation:
\[ 3x + 4 = 2x + 10 \]

To solve for \( x \):
1. Subtract \( 2x \) from both sides:
   \[ 3x - 2x + 4 = 2x - 2x + 10 \]
   \[ x + 4 = 10 \]

2. Subtract 4 from both sides:
   \[ x + 4 - 4 = 10 - 4 \]
   \[ x = 6 \]

The solution provided by StudentAgent is correct. \( x = 6 \) is the right answer.

--------------------------------------------------------------------------------
Current state: VisualizingAnswer
Messages from: chat_manager sent to: ProgrammerAgent | num messages: 16 | message: {'
    content': "Let's verify the solution provided by StudentAgent.\n\nGiven the equation
    :\n\\[ 3x + 4 = 2x + 10 \\]\n\nTo solve for \\( x \\):\n1. Subtract \\( 2x \\) from
    both sides:\n   \\[ 3x - 2x + 4 = 2x - 2x + 10 \\]\n   \\[ x + 4 = 10 \\]\n\n2. Subtract
```

```
    4 from both sides:\n \\[ x + 4 - 4 = 10 - 4 \\]\n \\[ x = 6 \\]\n\nThe solution
    provided by StudentAgent is correct. \\( x = 6 \\) is the right answer.", 'name': '
    SolutionVerifierAgent', 'role': 'user'}
ProgrammerAgent (to chat_manager):

ProgrammerAgent, please generate the Python code to confirm the solution.

```python
# Given equation: 3x + 4 = 2x + 10
# We need to solve for x

from sympy import symbols, Eq, solve

# Define the variable
x = symbols('x')

# Define the equation
equation = Eq(3 * x + 4, 2 * x + 10)

# Solve the equation
solution = solve(equation, x)

# Print the solution
print(f"The solution for the equation 3x + 4 = 2x + 10 is x = {solution[0]}")
```

CodeRunnerAgent, please execute this code.

Current state: RunningCode
Messages from: chat_manager sent to: CodeRunnerAgent | num messages: 17 | message: {'
    content': 'ProgrammerAgent, please generate the Python code to confirm the solution.\
    n\n```python\n# Given equation: 3x + 4 = 2x + 10\n# We need to solve for x\n\nfrom
    sympy import symbols, Eq, solve\n\n# Define the variable\nx = symbols(\'x\')\n\n#
    Define the equation\nequation = Eq(3 * x + 4, 2 * x + 10)\n\n# Solve the equation\
    nsolution = solve(equation, x)\n\n# Print the solution\nprint(f"The solution for the
    equation 3x + 4 = 2x + 10 is x = {solution[0]}")\n```\n\nCodeRunnerAgent, please
    execute this code.', 'name': 'ProgrammerAgent', 'role': 'user'}

>>>>>>>> EXECUTING CODE BLOCK 0 (inferred language is python)...
CodeRunnerAgent (to chat_manager):

exitcode: 0 (execution succeeded)
Code output:
The solution for the equation 3x + 4 = 2x + 10 is x = 6

Current state: UpdatingModel
Messages from: chat_manager sent to: LearnerModelAgent | num messages: 18 | message: {'
    content': 'exitcode: 0 (execution succeeded)\nCode output: \nThe solution for the
    equation 3x + 4 = 2x + 10 is x = 6\n', 'name': 'CodeRunnerAgent', 'role': 'user'}
LearnerModelAgent (to chat_manager):

The solution has been confirmed by the Python code execution. The value of \(x \) is
    indeed 6.

Updated Assessment
```

```
Based on the successful solution of the equation \( 3x + 4 = 2x + 10 \), it appears that
    the StudentAgent is proficient in solving linear equations with variables on both
    sides.

#### Next Steps
To further challenge the StudentAgent and assess their understanding, I recommend
    introducing problems that involve:
1. Linear equations with fractions.
2. Systems of linear equations.
3. Quadratic equations.

Would you like to proceed with one of these types of problems, or is there another area
    in algebra you'd like to explore?

--------------------------------------------------------------------------------
Current state: AdaptingLevel
Messages from: chat_manager sent to: LevelAdapterAgent | num messages: 19 | message: {'
    content': "The solution has been confirmed by the Python code execution. The value of
     \\( x \\) is indeed 6.\n\n### Updated Assessment\nBased on the successful solution
    of the equation \\( 3x + 4 = 2x + 10 \\), it appears that the StudentAgent is
    proficient in solving linear equations with variables on both sides.\n\n#### Next
    Steps\nTo further challenge the StudentAgent and assess their understanding, I
    recommend introducing problems that involve:\n1. Linear equations with fractions.\n2.
     Systems of linear equations.\n3. Quadratic equations.\n\nWould you like to proceed
    with one of these types of problems, or is there another area in algebra you'd like
    to explore?", 'name': 'LearnerModelAgent', 'role': 'user'}
LevelAdapterAgent (to chat_manager):

Given the StudentAgent's proficiency with linear equations, let's introduce a problem
    involving fractions to further challenge their skills.

### New Problem
Solve the following equation:

\[ \frac{2x}{3} + 4 = \frac{x}{2} + 5 \]

StudentAgent, please solve this equation.

--------------------------------------------------------------------------------
Current state: Motivating
Messages from: chat_manager sent to: MotivatorAgent | num messages: 20 | message: {'
    content': "Given the StudentAgent's proficiency with linear equations, let's
    introduce a problem involving fractions to further challenge their skills.\n\n### New
     Problem\nSolve the following equation:\n\n\\[ \\frac{2x}{3} + 4 = \\frac{x}{2} + 5
    \\]\n\nStudentAgent, please solve this equation.", 'name': 'LevelAdapterAgent', 'role
    ': 'user'}
MotivatorAgent (to chat_manager):

You're doing an amazing job, StudentAgent! Keep up the great work and tackle this new
    challenge with confidence.

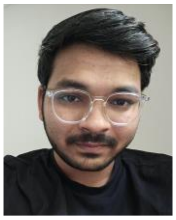
### Problem:
\[ \frac{2x}{3} + 4 = \frac{x}{2} + 5 \]

Take your time to solve for \( x \). If you need any assistance or guidance, we're all
    here to support you!

--------------------------------------------------------------------------------
```

D Appendix - Authors

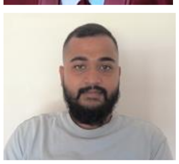
Srikanth Dokku is a graduate student at Rivier University studying Computer Science. He has three years of professional experience working for a government agency. His bachelor's degree is from Gudlavalleru Engineering College.



Anudeep Gumpula is a graduate student at Rivier University studying Computer Science. He was previously an Assistant System Engineer at Tata Consultancy Services. His research interests include distributed systems, data management, and machine learning. He holds Microsoft Azure Fundamentals and Cisco Certified Network Associate (CCNA) professional certifications.

Sanjana Gundati is a graduate student at Rivier University studying Computer Science. She completed an undergraduate degree in Computer Science. Her current research interests include software development, networking, and algorithms.

Mahendra Nagisetty is a graduate student at Rivier University studying Computer Science. He was previously a lead infrastructure engineer in Pragma Edge Software PVLTD. His research interests include distributed systems, data management, and machine learning. He holds Microsoft Azure Fundamentals and Cisco Certified Network Associate (CCNA) professional certifications. He can be contacted at mnagisetty@rivier.edu.



Ruthvik Thimmarayappa is a graduate student at Rivier University studying Computer Science. He holds a bachelor's degree in Information Science from Visvesvaraya Technological University (VTU). His research focuses on the Internet of Things (IoT) and smart technology applications. He has authored several publications, including "Design and Implementation of IoT-Based Smart Shopping Dash Cart" published in the International Journal of Engineering Technology and Management Sciences (IJETMS) in July 2021, and "Survey on Smart Shopping Cart" published in Iconic Research and Engineering Journals (IRE Journals) in June 2021. He can be reached at rthimmarayappa@rivier.edu or ruthvik.tk@gmail.com.

Jai Paul Reddy Yeruva is a graduate student at Rivier University studying Computer Science. He completed an undergraduate degree in Computer Science and Engineering (CSE).

John Glossner is an Associate Professor and Director of Computer Science Programs at Rivier University. He teaches graduate-level Programming, Computer Architecture, Operating Systems, Computer Security, Object-Oriented Design, Reading and Research, and Professional Seminar.

Prior to Rivier, he was a Professor and Director of the Computer Architecture, Heterogeneous Computing, and AI Lab at the University of Science and Technology in Beijing (USTB), where he supervised Ph.D. students. Prior to USTB, he was the Chair of Computer Science at Daniel Webster College where he taught undergraduate students. In addition to faculty positions, he has held multiple industry positions at IBM Microelectronics, IBM Research, and Bell Labs.

In 2001, he co-founded Sandbridge Technologies which raised $70M in venture funding. Before being acquired, the World Economic Forum recognized the company for developing world-changing DSP technology widely used in today's commercial cell phones.

After Sandbridge, he became CTO of China-based Huaxia General Processor Technologies (hxGPT) and CEO of its U.S.-based subsidiary Optimum Semiconductor Technologies Inc. (OST), where he led embedded development of heterogeneous processor designs including variable length vector processors and domain-specific AI accelerators.

Dr. Glossner has also served voluntarily as President of both the Heterogeneous System Architecture Foundation (HSAF) and the Wireless Innovation Forum. He has more than 120 publications and 40 issued patents.