Proposal for Solo Capstone Project

Group Name: Solo Project — Simple Math Tutor

Participant: Judith Barrios

Course: ITAI2376 – Capstone Project

Project Title: Beginner-Friendly Math Tutor with Calculator Support

GitHub Link to SmartMathTutor - https://github.com/JudithBarrios/Math-Tutor-AI-Agent.git

Problem Statement

Many students who are just beginning to learn math often face challenges because the support they need is not always available. Whether it is due to limited classroom time, large class sizes, or not having access to tutors at home, some learners fall behind simply because they do not get enough practice. There are tools online, but they are sometimes too advanced, filled with features that make them hard to use, or rely on a constant internet connection.

The problem I want to solve with this agent is to give students a basic, supportive way to practice simple math without needing fancy devices, online tools, subscriptions. It is focused on beginners who need practice with things like addition, subtraction, multiplication, and division.

For my project, I chose Option 4: Interactive Learning Companion, this choice lets me build something that interacts with the user in a helpful and kind way, offers guidance, and even allows them to explore math expressions on their own through a calculator tool. I felt this way best fit the goal of making learning easier for people who may be starting from nothing.

Agent Design

The agent is made using an uncomplicated design that includes all the important parts of a smart system. First, there is Input Processing, where the agent reads the user's response, whether it is a number, a request for help, a calculator command, or a way to end the session. This is how it understands what the user wants.

Next is the Memory System, which keeps track of how many questions the user has answered and how many they got right, allowing the agent to give a summary at the end so users know how they did overall.

Then we have the Reasoning Component, which is where the agent checks if the answer is correct, incorrect, or valid, and decides how to respond. It also knows when to give a hint or activate the calculator. Finally, there is the Output Generation, which is the agent's way of communicating, this includes giving feedback that explains what the correct answer is and why.

I used the Planning-then-Execution method, where the agent decides what to do before it takes any action, this helps the system stay organized and predictable for the user.

Tool Selection and Development Plan

The agent uses two tools: a basic Calculator Mode and a Hint System, the calculator lets the user type in math expressions to solve, which is helpful if they want to double check their work or just experiment. It only allows safe characters like numbers and symbols used in math, and it checks before solving anything.

The hint tool is tied to each question and gives it a brief explanation of how to approach solving it, without giving the answers away, these tools help the user feel like they have support instead of just being tested.

Here is my step-by-step development plan:

- Week 1: Brainstorm ideas, design the interface, and pick the questions
- Week 2: Build the question system and add hints
- Week 3: Add the calculator, text the functions, and build in error protection
- Week 4: Write helpful feedback messages, improve user instructions, and clean up the code

Evaluation, Resources and Risks

To evaluate this agent, I ran different test sessions where I answered questions correctly and incorrectly, asked for hints, used the calculator, and tested the exit command. The goal was to make sure it gave exact feedback, did not crash, and handled all kinds of user input without problems.

Since the project was built using Google Colab and basic Python, it does not require any expensive tools or paid APIs. That makes it more accessible and easier to test for other students or teachers who want to try it out.

Some risks included users typing in strange calculator expressions or giving answers that could not be understood, I solved this by adding input filters and clear messages. Another risk was users not understanding how to interact, so I added clear explanations at the start of the session. In the future, I hope to expand the agent's abilities to include more types of questions, a learning path, or even small games that help build math skills.

Additional Considerations

Beyond the immediate technical and timeline concerns, it is important to consider the user experience carefully, especially for a learning tool intended for beginners. Ensuring that the agent provides clear, encouraging feedback will be essential for maintaining motivation and engagement throughout the learning sessions. This requires iterative testing with real users to refine the interaction style and messaging.

Moreover, planning for scalability will be beneficial if this project is to be expanded in the future. Designing the codebase and architecture to be modular and adaptable will make it easier to incorporate added topics, more complex problem types, or advanced features such as speech input and output.

Finally, maintaining detailed documentation from the start is crucial, clear and accessible documentation not only aids current development and testing but also supports future contributors who may build on or maintain the agent. This includes documentation decisions, met challenges, and how they were overcome.

In the process of creating a solo project, time management also became an important part of the overall plan. Since every part from design to testing had to be handled by one person, breaking the workload unto manageable weekly goals helped prevent burnout and ensured that steady progress was made without feeling overwhelmed. This pacing also allowed time to go back and revise features or improve the feedback system based on how it performed in practice.

Another consideration was the importance of making the tool accessible for a wide range of users, because the goal was to help people improve their math skills in a beginner friendly way, the interface was designed to be clear and text-based, avoiding overwhelming visuals or technical language. This helped ensure that users of all ages or backgrounds could feel

comfortable interacting with the tutor, regardless of their prior experience with technology or math tools.