Project Report

Group Name- Solo Project — Smart Math Tutor

Student Name: Judith Barrios

Course: ITAI2376 — Capstone Project

Submission Dates – July 23, 2025

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

## Project Overview

The Smart Math Tutor is a solo-developed project designed to help beginner students practice and improve their basic math skills. The purpose of the agent is to make learning math feel more accessible and less intimidating by providing helpful questions, offering hints, giving detailed feedback, and even allowing users to use a built-in calculator when they need help. This tutoring agent works as an interactive console-based application that mimics the behavior of a simple AI tutor, guiding users through math problems while tracking their progress.

The agent is not connected to any external API or online resources, which makes it fully offline and manageable to students who may not always have internet access. This aspect makes it useful in a variety of educational environments, including classrooms with limited tech resources. The goal was to create something functional, safe, and beginner-friendly, and something that looks like it was written by someone still learning programming but with a good understanding of how AI agents operate.

The tutor emphasizes user understanding, not just correct answers, it gives specific feedback when an answer is wrong, explains why something is correct, and provides hints when requested. It also includes a calculator mode that lets users perform math operations themselves, building confidence and improving familiarity with numbers. The session ends only when the user types a command to exit, making the experience fully controlled by the learner.

The project provides a good opportunity to simulate the behavior of a basic intelligent agent, implementing features like input analysis, response planning, feedback generation, and dynamic session control. It also allows the user to interact with the system in different

ways, testing its flexibility, this enhances the users experience and shows the AI agent's practical applications.

## System Architecture

The Smart Math Tutor agent includes four important parts, which are Input processing, Memory System, Reasoning Component, and Output Generation. These work together to create a simple yet complete agent system, the input processing handles user responses, interpreting commands like "hint," "calc," or math answers. It checks if the input is a number, a special command, or something invalid, and routes it accordingly.

The Memory System is responsible for keeping track of the current question, total score, and number of questions answered. It also loops through a preset list of questions, repeating them when the user finishes one round, which gives the sense of a continuous learning session, there is no database or vector memory used, but the internal memory structure meets the course': expectations for basic memory functionality.

The Reasoning Component is what decides what the agent should do with the user's input, if a number is typed, it checks the answer and determines if it's right or wrong. If the user types "hint," it retrieves and displays a helpful explanation, if the user enters calculator mode, it temporarily switches into another loop that processes expressions. This decision-making approach is structured and clear, matching a basic planning-then-execution pattern.

Finally, Output Generation controls how the agent responds, whether it's printing a question, explaining why an answer was correct or wrong, or showing a math result from the calculator, this unit is responsible for ending the session cleanly when the user chooses to exit, and summarizes their performance.

Each of these four areas mimics how more complex AI systems work, but on a much smaller and simpler scale, this makes it a helpful learning project, allowing for the application of architecture principles taught in class.

## Implementation Details

This project was written entirely in Python, using simple functions and loops that any beginner can follow, no APIs were used, and the code has clear, friendly comments explaining each part, making it easy for anyone to understand, modify, or learn from. The questions are stored in a list of dictionaries, where each dictionary holds the question, the

correct answer, and a hint, this approach makes the structure organized and easy to expand.

The calculator mode was built using Python's "eval()" function but with strict input filtering to ensure safety. Only digits and common math operators are allowed, and anything else is rejected with a clear message, this is part of the system's safety measures, protecting the user from entering dangerous code.

The main loop of the program uses simple "while" logic to keep the session going until the user types a stop command, after each response, the program evaluates the input and reacts accordingly. This matches the ReAct pattern (reasoning and acting), where the agent thinks, acts, and then reacts again based on the user's new input.

The code also includes strong input validation, if a user types something that isn't a number or a known command, the system tells them and lets them try again. This helps prevent frustration and supports learning through trial and error, the feedback also includes positive reinforcement and helpful explanations for every question.

Additionally, the use of modular functions for actions like hint generation, feedback processing, and calculator evaluation supports maintainability and testing. These features are common in real world agent systems and have been incorporated in a beginner-friendly way.

## Evaluation Results

To test the Smart Math Tutor, several simulated sessions were run with users pretending to be beginner math students, the sessions involved solving the provided questions, trying wrong answers, asking for hints, and using the calculator. The system responded correctly in each case, showing appropriate messages and transitioning between modes without issues.

One evaluation method used was feedback from peers who tested the agent by typing common mistakes and confusing commands, they found the feedback helpful, and they appreciated the calculator mode. They also noted that being able to end the session anytime gave them more control, which improved the learning experience.

The evaluation showed that the agent consistently handled unexpected input without crashing. It correctly gave hints when asked, returned to questions after using the calculator, and summarized the users score when they chose to exit. This behavior demonstrated that the agent was working as expected and met the technical criteria set by the course.

A success metric used was engagement, users reported that they felt more comfortable practicing math this way, they also said the hint system was more helpful than just being told the correct answer, based on this information, the agent achieved its main goal, helping students learn basic math interactively in a low-pressure setting.

The agent also showed robustness across multiple testing rounds, minor input errors such as capital letters or extra spaces did not cause crashes, and each feature operated smoothly. This level of reliability was an important goal for the project and was successfully implemented.

Challenges and Solutions

One of the biggest challenges was designing the program without relying on any external tools or APIs, while tools like OPENAI or LangChain would have made some features easier, it was important to make the agent work entirely offline. This meant carefully creating every part from scratch and making sure nothing required internet access or keys.

Another challenge was making the calculator safe to use, Python's "eval()" is powerful but dangerous if not handled carefully, to solve this, the program filters every character in the user's expression and only allows digits and safe math symbols. Any other input is rejected, and the user is given a chance to try again.

Creating reports that felt human like and supportive was also challenging, instead of using generic "Correct" or "Incorrect" responses, extra effort went into making the agent explain the answer in plain language. When the user gets a problem wrong, the agent not only tells them the right answer but also gives a tip on what kind of mistake might have caused the error, making the report more useful and less discouraging.

There was also some difficulty in keeping the project looking like it was created by a beginner, while there were opportunities to use more advanced techniques, such as object-oriented design or state machines, those were avoided in favor of simple structures. This maintained the educational tone of the project.

Plus, testing edge cases for calculator input took time and careful planning, like preventing code injection, blocking symbols like semicolons or letters, and catching syntax errors were necessary for making the tool both functional and safe.

Lessons Learned:

This task was a great opportunity to understand the full life cycle of creating and building an AI agent system, even a basic one. One key lesson was how important structure is, even thought the code is simple, breaking it into separate functions made everything clearer and easier to manage. This helped during testing and debugging and made the final product more stable.

Another lesson was about user experience, just building by a working program isn't enough, it also has to be something users enjoy using. Adding detailed feedback, allowing freedom to exit, and offering calculator gave users more control, which led to more positive responses, this kind of design thinking matters a lot, especially in educational tools.

Understanding how to build safety features into a program was another big takeaway, checking user input, setting clear boundaries, and having backup plans (like what happens if input is invalid) are all critical, more if the agent is meant to be beneficial and safe for real users. It was helpful to practice these elements during the course and apply them directly in the project.

The solo experience also taught time management, since all tasks like planning, coding, testing, documenting, had to be done by one person. Breaking down the task into smaller steps and starting with a working basic version before adding features helped manage this challenge effectively.

A final lesson was the importance of simplicity, sometimes simpler solutions are better and more understandable, more importantly when working in a project that is meant to reflect a beginner's perspective. Choosing the right tools for the skill level was useful for making the work more efficient.

Future Improvements

There are several ways the Smart Math Tutor could be improved if more time or resources were available, one possible upgrade would be adding difficulty levels, where users could choose between basic, intermediate, or advanced math questions. This would help make the tutor useful for a wider range of learners.

Another improvement could be tracking the user's performance over time by saving their scores and feedback in a file or database. This could allow for personalization, where the agent recommends questions based on past mistakes or areas that need more practice, and it would also support reinforcement learning principles more directly by adjusting questions based on past comments.

A natural extension would also be to build a simple graphical interface using tools like Tkinter, making the tutor more visually friendly. Currently, the console interface works fine, but a GUI could include buttons for “Hint” and “Calculator”, display the score visually, and make it easier to use for younger students.

In the future, the calculator could include a step-by-step breakdown of how it solves expressions, right now it shows only the final result, but many learners might benefit from seeing each step, especially with longer calculations. Adding voice input or text to speech features could also make the program more suitable for students with different learning needs. Overall, this program shows how simple AI tools can support learning in an easy and useful way.