AI-Based Adaptive Tutoring System with Conditional Memory

Md Al Amin
Department of Electrical and Computer Engineering, North South University,
mdal.amin5@northsouth.edu
Mohaimen Azad
Department of Electrical and Computer Engineering, North South University,
mohaimen.azad@northsouth.edu
Aminul Islam Arafat
Department of Electrical and Computer Engineering, North South University,
aminul.arafat@northsouth.edu

Abstract

Traditional classroom settings face challenges in providing personalized, step-by-step guidance for every student. This paper presents the development and evaluation of three distinct AI-driven math tutoring systems designed to enhance student learning: a RAG-Based Math Topic Teacher, an Adaptive Problem-Solving Tutor with Conditional Memory, and a Bilingual(Bangla and English) Step-by-Step Problem Solver. The RAG-based system leverages retrieval-augmented generation to teach mathematical concepts from provided documents, mimicking a human teacher's persona and pedagogical strategies. The Adaptive Tutor employs response analysis and dynamic prompt generation based on student performance and memory to deliver personalized hints and guide students through problems without providing direct solutions. The Bilingual Tutor offers accessible step-by-step problem-solving in both English and Bangla. We detail the architecture, implementation, and unique features of each system, highlighting their contributions to creating more engaging, effective, and accessible AI math education tools.

CCS Concepts ● Artificial Intelligence ● Intelligent Tutoring System ● Large Language Model Application

1. Introduction

Mathematics education is foundational to academic and professional success. However, students often struggle with grasping complex concepts or solving problems independently. While human tutors provide invaluable personalized support, they are not always scalable or accessible to all students. AI-powered tutoring systems have emerged as potential solutions, offering automated assistance, immediate feedback, and tailored learning experiences.

Developing effective AI tutors presents several challenges, including:

- **Content Coverage:** Ensuring comprehensive and accurate information across various topics and difficulty levels.
- **Pedagogical Approach:** Mimicking effective teaching strategies (e.g., breaking down concepts, scaffolding, checking understanding).

- **Adaptability:** Adjusting instruction based on individual student progress, misconceptions, and learning styles.
- **Engagement:** Maintaining student motivation and active participation.
- **Accessibility:** Supporting diverse linguistic needs and learning environments.

This project explores these challenges by developing three distinct AI math tutoring systems, each focusing on a different aspect of the tutoring process:

1. Teaching mathematical topics and concepts (RAG-Based Math Teacher).
2. Providing adaptive, hint-based guidance for problem-solving (Adaptive Problem-Solving Tutor).
3. Offering accessible step-by-step solutions in multiple languages (Bilingual Step-by-Step Tutor).

This paper describes the design, technical implementation, and operational flow of each system.

2. System Architecture and Implementation

All three systems utilize large language models (LLMs) orchestrated by the LangChain framework and hosted using Streamlit for the frontend interface. They leverage Groq's high-performance inference API for fast and responsive inference.

2.1 RAG-Based Math Topic Teacher

The RAG-Based Math Topic Teacher is designed to act as a comprehensive math instructor, capable of teaching concepts and explaining theories based on a corpus of educational documents. Its core mechanism relies on Retrieval-Augmented Generation (RAG) to fetch relevant information before generating a response.

Architecture:

The system follows a standard RAG pipeline enhanced with specific components tailored for educational content and a teacher persona. The architecture is illustrated in Figure 1.

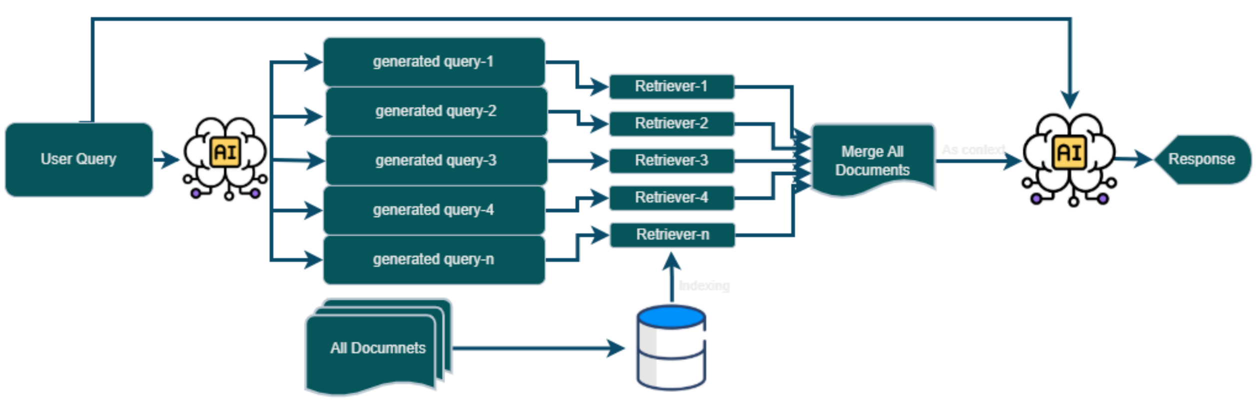

Figure 1: High-level overview of the RAG pipeline used in the project.

Implementation Details:

- **Data Loading and Processing:** The system loads mathematical content from PDF files stored in a local /Data directory using DirectoryLoader and PyPDFLoader.
- **Custom Chunking:** A crucial component is the MathTeacherChunker. Unlike generic text splitters, this custom chunker identifies and separates specific pedagogical sections within documents, such as "Overview," "Prerequisites," "Concepts" (marked by ## headers), "Practice Problems," and "Teaching Strategies." This allows the retriever to fetch chunks specifically relevant to the student's need (e.g., retrieving prerequisite knowledge when a student struggles).
- **Embedding and Vector Store:** HuggingFace embeddings (all-MiniLM-L6-v2) is used to convert text chunks into vector representations. These vectors are stored and indexed in a ChromaDB instance, persisted locally in the /chroma_db directory for efficient retrieval.
- **Retrieval:** The system employs a combination of retrieval strategies:
    - A base vector store retriever (vector_store.as_retriever).
    - A MultiQueryRetriever to generate multiple search queries from a single user input, improving the chances of finding relevant documents, especially for complex questions.
    - A HistoryAwareRetriever which rephrases the user's query based on the chat history, allowing the system to maintain context across turns.
- **LLM and Chain Orchestration:** A ChatGroq model (meta-llama/llama-4-scout-17b-16e-instruct) serves as the generation engine. LangChain is used to create the create_retrieval_chain, which combines the history-aware retrieval step with a document combination step (create_stuff_documents_chain).
- **Prompt Engineering:** The system utilizes a sophisticated, multi-part prompt (get_teacher_prompt) to define the AI's persona and teaching strategy. This prompt includes:
    - A base teacher identity emphasizing clear, real-world explanations and addressing misconceptions.
    - Pedagogical principles like starting with context, scaffolding, using multiple representations, and checking understanding.
    - An "Interaction pattern" outlining a classroom-like sequence (opening, activation, instruction, modeling, guided practice, checking, independent practice, summary, preview).
    - *Topic-specific* guidance (e.g., for Algebra, Geometry, Trigonometry, Functions) that injects relevant teaching tips based on the detected topic of the user's query.
    - A RAG instruction specifically guiding the LLM on how to use the retrieved context to answer the {input} while adhering to the teacher persona and pedagogical steps.
- **Topic Detection:** A simple keyword-based function (detect_topic) attempts to identify the mathematical domain (algebra, geometry, trigonometry, functions) from the user's query to inform the dynamic prompt selection.
- **Frontend:** Streamlit provides the conversational chat interface, allowing users to interact with the teacher persona and visualizing the chat history. Options are available to view retrieved documents for transparency.

This system's strength lies in its ability to provide context-rich, pedagogically sound explanations derived from a curated knowledge base, simulating a knowledgeable human teacher.

2.2 Adaptive Problem-Solving Tutor

The Adaptive Problem-Solving Tutor is designed to guide students through solving mathematical problems interactively, providing only hints and asking questions, never full solutions. It adapts its hint strategy based on the student's responses, incorporating a form of conditional memory.

Architecture:

This system's architecture focuses on analyzing student input and dynamically adjusting the AI's behavior via prompt modifications. The architecture is depicted in Figure 2.

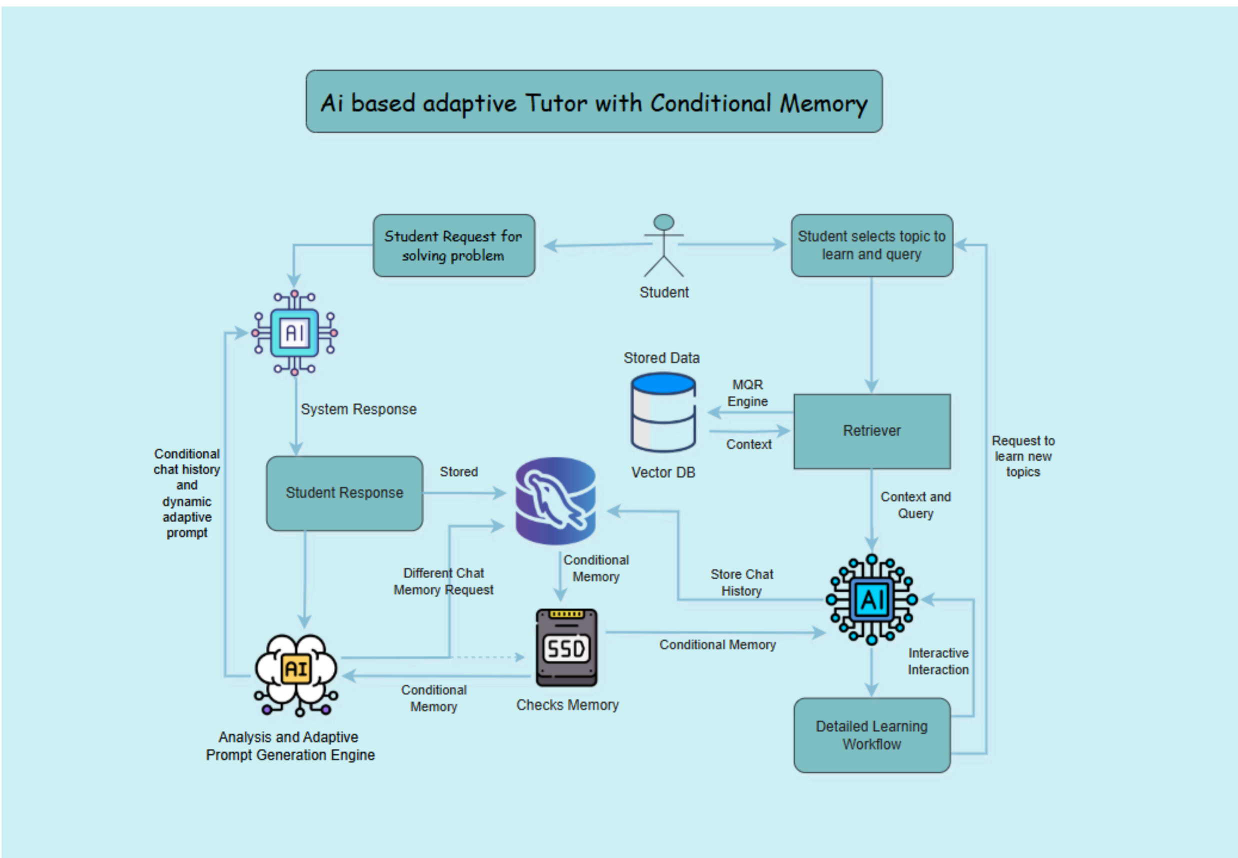

Figure 2: AI-Based Adaptive Tutor with Conditional Memory System Design

Implementation Details:

- **LLM:** A ChatGroq model (qwen-qwq-32b) is used for its performance in generating structured outputs and following complex instructions.
- **Student Response Analysis:** The system analyzes each student response to the previous hint using a separate LLM chain (sentiment_chain). It classifies the student's response into one of three categories: correct, partially_correct, or incorrect. Pydantic models (StuOutputCorrectness)

and a custom RobustOutputParser are used to ensure reliable extraction of this classification from the LLM's output, even if the output contains extraneous text.

- **Conditional Memory:** The system tracks the number of consecutive_incorrect responses in the Streamlit session state.
- **Adaptive Prompt Generation:** Based on the sentiment classification and the conditional memory, a dedicated LLM chain (adaptive_prompt_generator) dynamically generates an *adaptive system prompt*.
  - If correct: Prompt guides the tutor to move to the next step and provide a hint for *that* step.
  - If partially_correct: Prompt guides the tutor to re-explain the *current* step or provide a more targeted hint for it.
  - If incorrect: Prompt guides the tutor to provide a more basic hint, break the *current* step into smaller sub-steps, or check for prerequisite understanding.
  - If consecutive_incorrect >= 2: Additional guidance is added to the adaptive prompt, instructing the tutor to provide more explicit hints, break down steps further, or consider prerequisite gaps, while still forbidding full solutions.

    Pydantic (AdaptivePrompt) and the robust parser are used for structured adaptive prompt extraction.
- **Main Tutor Chain:** The primary tutor interaction uses a LangChain RunnableWithMessageHistory. The *system prompt* for this runnable is dynamically constructed by combining a static base_prompt (defining the core hint-only, step-by-step policy) with the dynamically generated adaptive_prompt. This allows the system's behavior to shift subtly based on student performance.
- **Structured Tutor Response:** The main tutor chain is instructed to output its response in a specific JSON format (TutorResponse Pydantic model) containing a "hint" and a "reasoning" field. The RobustOutputParser extracts this, and the "reasoning" is shown in an expandable section in the UI.
- **Hint Enforcement:** A post-processing step (enforce_hint_only_approach) is implemented to programmatically check the generated "hint" for patterns indicative of providing a full solution. If detected, the hint is replaced with a generic "hint-only" message, reinforcing the core policy.
- **Frontend:** Streamlit provides the chat interface. The sidebar dynamically updates with the sentiment classification, generated adaptive prompt, and current hint level (based on consecutive incorrect responses), offering transparency into the system's internal state.

This system excels at providing personalized guidance during problem-solving, actively preventing the student from simply copying solutions and encouraging critical thinking through targeted hints and adaptive difficulty.

Example Interaction (Adaptive Tutor):

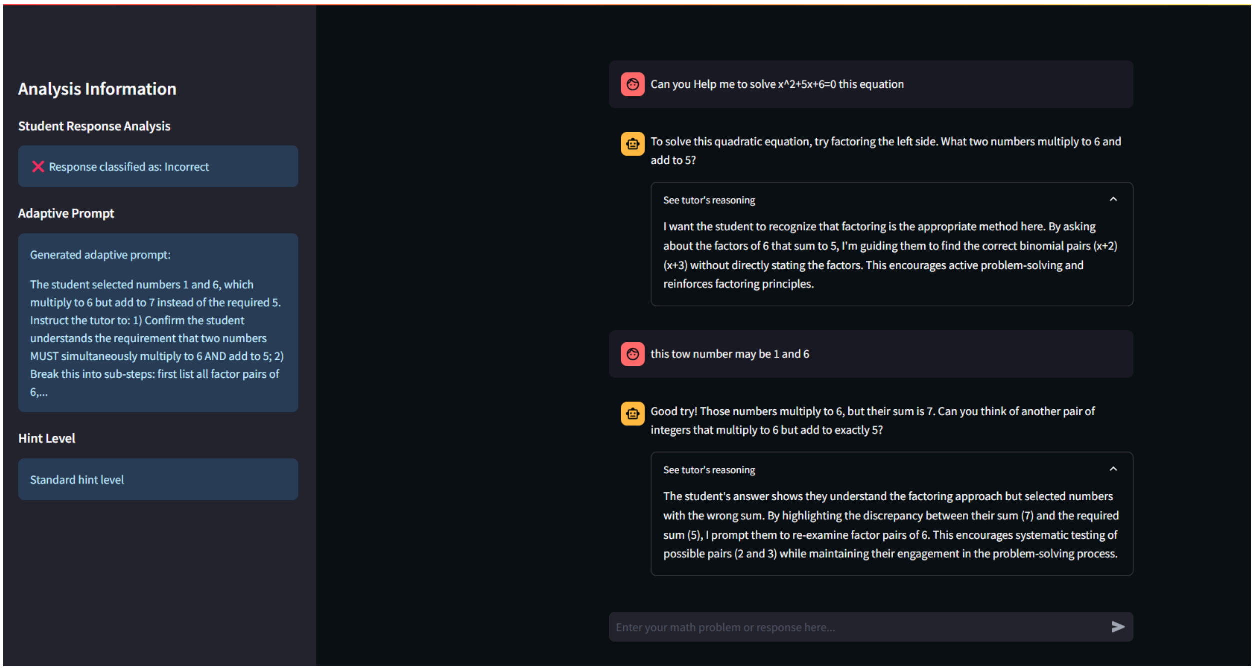

Figure 3: Adaptive Tutor handling an incorrect student response, showing analysis and adaptive prompt.

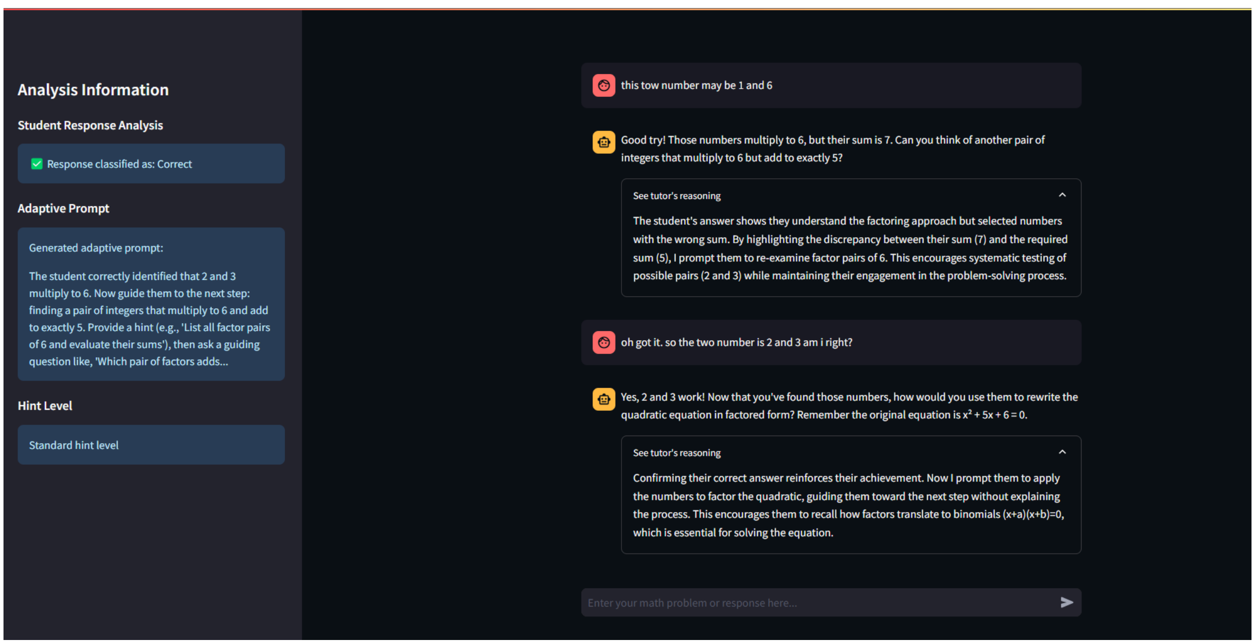

Figure 4: Adaptive Tutor handling a correct student response, showing analysis and adaptive prompt.

2.3 Bilingual Step-by-Step Problem Solver

The Bilingual Step-by-Step Problem Solver offers a simpler, direct approach to solving problems by breaking down the solution into explicit steps, supporting both English and Bengali.

Architecture:

This system is a more straightforward chat application leveraging language-specific static prompts and message history.

Implementation Details:

- **LLM:** A ChatGroq model (meta-llama/llama-4-scout-17b-16e-instruct) is used.
- **Bilingual Prompting:** The system uses two distinct, static system prompts: SYSTEM_PROMPT_ENGLISH and SYSTEM_PROMPT_BANGLA. These prompts instruct the AI to act as a math tutor, solve problems one step at a time, explain each step, and crucially, ask the student if they understand before proceeding. The response language is strictly enforced by the prompt.
- **Step-by-Step Logic:** The step-by-step nature is primarily driven by the instructions within the static system prompts, rather than dynamic analysis of student responses. The AI is told to explain *one* step and ask for confirmation.
- **LangChain Components:** A RunnableWithMessageHistory is used to maintain the conversation context.
- **Frontend:** Streamlit provides the chat interface and a sidebar language selection dropdown.

This system prioritizes accessibility through bilingual support and a clear, explicit step-by-step approach, making it suitable for students who benefit from seeing the solution process broken down.

Example Interaction (Bilingual Tutor - Bangla):

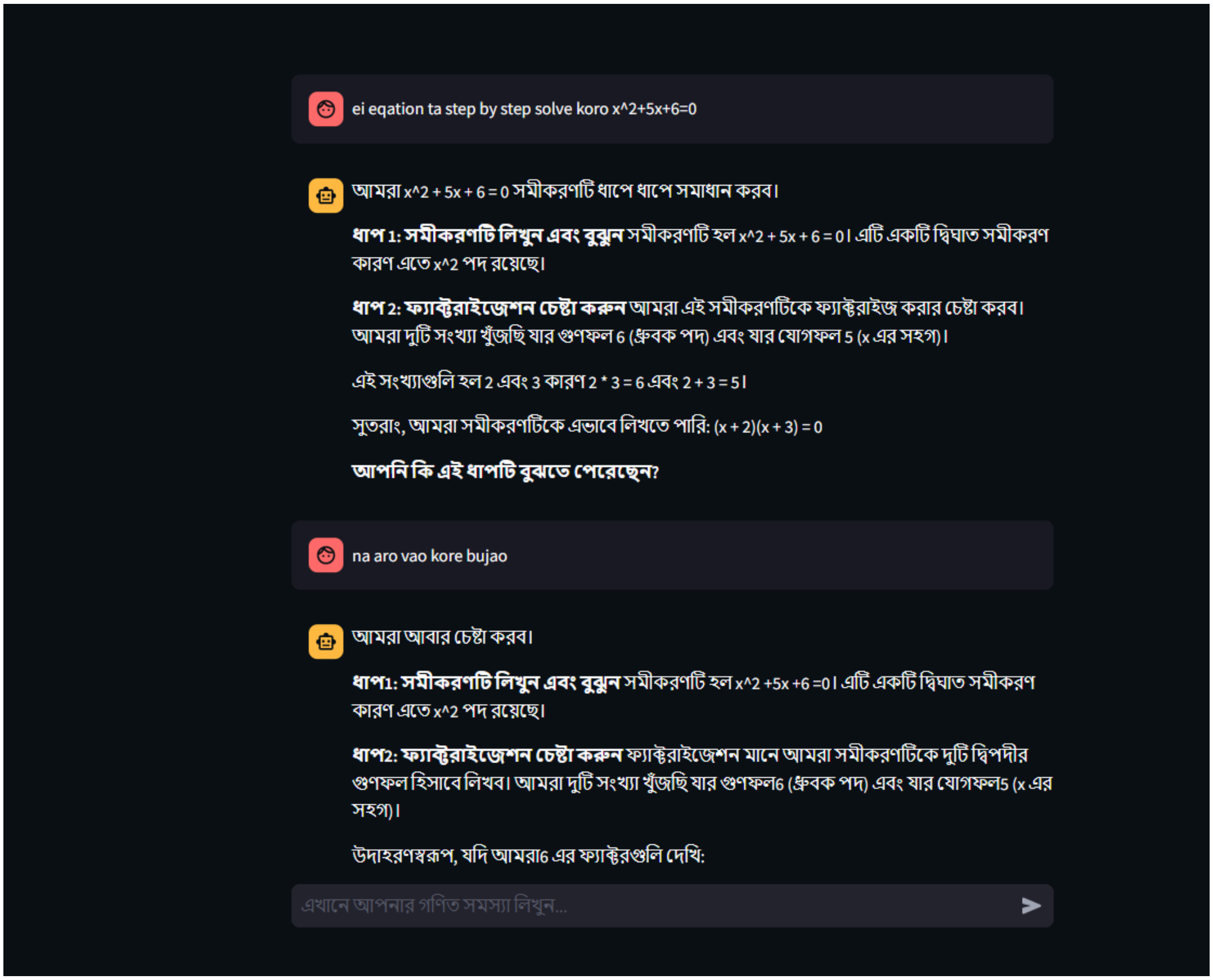

Figure 5: Bilingual Tutor (Bangla) starting to solve a quadratic equation and checking understanding.

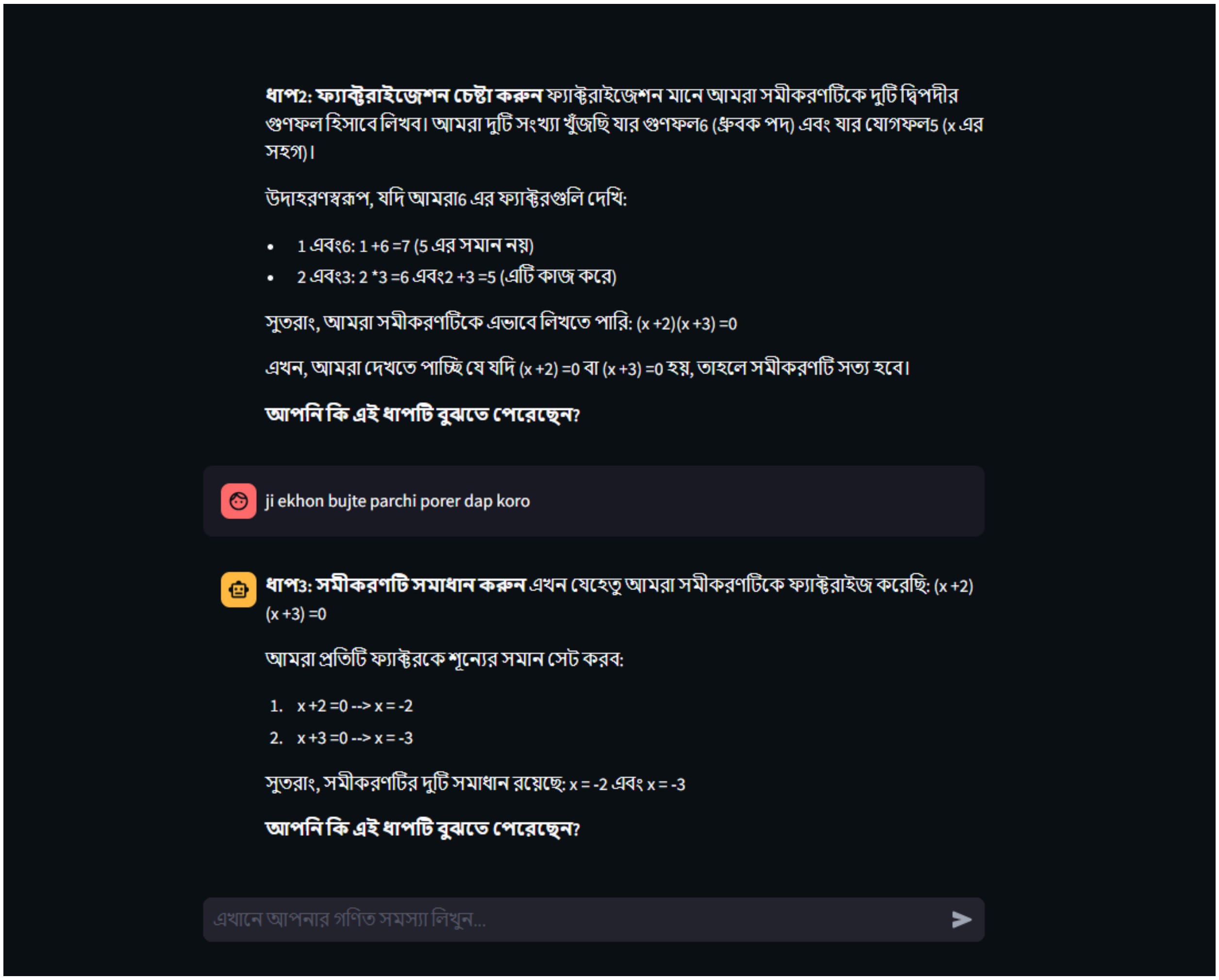

Figure 6: Bilingual Tutor (Bangla) proceeding to the next step after student confirmation.

3. Discussion and Future Work

The development of these three systems demonstrates the versatility of AI and LLMs in creating math tutoring tools with different pedagogical focuses.

- The **RAG-Based Teacher** is powerful for concept explanation and topic teaching, providing depth and relevance by grounding responses in a curated knowledge base. Its effectiveness is highly dependent on the quality and structure of the input documents. Future work could involve incorporating active learning techniques where the RAG system suggests related topics or prerequisites based on the conversation.
- The **Adaptive Tutor** offers a sophisticated approach to problem-solving by dynamically adjusting its hint strategy. The sentiment analysis and conditional memory components allow for a more personalized and challenging learning experience, pushing students towards independent thinking. The robust parsing is essential for reliable operation. Future improvements could include more nuanced sentiment categories, tracking progress across multiple problems, and integrating different problem-solving strategies based on student preference or history.

- The **Bilingual Tutor** provides a simple yet effective step-by-step method with crucial language support. While less adaptive than the Adaptive Tutor, its clarity and accessibility are significant advantages. Future work could involve incorporating some level of adaptability or hint-based feedback within the bilingual framework.

Combining elements from these systems could lead to a more comprehensive AI tutor. For instance, an adaptive problem solver could retrieve relevant concept explanations from a RAG system when a student struggles with a prerequisite skill. Similarly, the adaptive features could be integrated into the bilingual system for language-supported adaptive tutoring.

Further formal evaluation with student users is necessary to assess the learning effectiveness, engagement levels, and usability of each system in a real educational setting.

4. Conclusion

This project successfully developed and implemented three distinct AI math tutoring systems using modern LLM and RAG technologies. The RAG-Based Math Teacher offers a robust approach to teaching mathematical topics from structured documents, leveraging custom chunking and detailed prompt engineering to create a teacher persona. The Adaptive Problem-Solving Tutor provides personalized, hint-driven guidance by analyzing student responses and dynamically adjusting its strategy based on a memory of performance, effectively fostering independent problem-solving skills. The Bilingual Step-by-Step Tutor enhances accessibility by offering clear, incremental solutions in both English and Bangla. These systems represent valuable explorations into the capabilities of AI in educational technology, offering different models for enhancing math learning experiences. The insights gained provide a strong foundation for developing more integrated and sophisticated future AI tutors.

REFERENCES

[1] J. R. Anderson, C. F. Boyle, and G. Yost, "Intelligent tutoring systems," *Science*, vol. 228, no. 4698, pp. 456–462, Apr. 1985, doi: 10.1126/science.228.4698.456.

[2] T. B. Brown *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[3] Chroma, "Chroma Documentation," [Online]. Available: https://www.trychroma.com/. [Accessed insert date].

[4] Groq, "Groq Documentation," [Online]. Available: https://groq.com/. [Accessed insert date].

[5] Hugging Face, "Hugging Face Models," [Online]. Available: https://huggingface.co/models. [Accessed insert date].

[6] LangChain, "LangChain Documentation," [Online]. Available: https://www.langchain.com/. [Accessed insert date].

[7] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv preprint arXiv:2005.11401*, 2020.

[8] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] Pydantic, "Pydantic Documentation," [Online]. Available: https://pydantic-docs.helpmanual.io/. [Accessed insert date].

[10] V. J. Shute, "Focus on formative feedback," *Rev. Educ. Res.*, vol. 78, no. 1, pp. 153–189, Mar. 2008, doi: 10.3102/0034654307313795.

[11] Streamlit, "Streamlit Documentation," [Online]. Available: https://streamlit.io/. [Accessed insert date].

[12] K. VanLehn, "The challenge of cognitive tutor technology," *Int. J. Artif. Intell. Educ.*, vol. 16, no. 3, pp. 217–231, Jul. 2006.