Design and Implementation of a Personalized AI Tutor for University Education Using RAG

Diana Ryzhkova[1], Gregor Rozinaj[1]
[1] Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia
ryzhkova.diasichka@gmail.com

Abstract - ***This paper presents the design and implementation of an AI-based educational assistant aimed at supporting university students in their learning process. The system utilizes Retrieval-Augmented Generation (RAG) [1] to provide personalized guidance and encourage active learning through questions and exploration rather than supplying direct answers. Technical implementation focuses on a dynamic chat interface combining modern AI models [2], vector databases for retrieval, and advanced prompt engineering strategies. Preliminary observations based on simulated interactions suggest that the tutoring approach encourages deeper engagement and critical reasoning. This aligns with findings from prior research indicating that AI systems promoting active student reflection can lead to improved learning outcomes compared to direct-answer systems [6], [7]. Future improvements aim to include features such as knowledge gap identification, adaptation to the student's current understanding level, and curriculum-aware guidance.***

Keywords - ***AI tutor, RAG, personalized education, interactive learning, knowledge retrieval***

I. Introduction

Artificial Intelligence (AI) is increasingly transforming the educational landscape, offering new tools for student support and knowledge acquisition [2]. Traditional AI systems in education often rely on static responses or direct-answer models, which limit the student's engagement and critical thinking. This paper introduces a novel AI-based tutoring assistant designed to enhance learning through active participation. The system leverages Retrieval-Augmented Generation (RAG) [1] to ensure responses are grounded in approved course materials and to encourage students to reflect, reason, and gradually construct their knowledge with AI support. The main goals of the project are to design an AI tutor that stimulates critical thinking instead of rote memorization, to implement a curriculum-aligned retrieval framework, and to create an adaptive, interactive environment that motivates deeper engagement with course content. Additionally, the project evaluates the impact of the tutoring approach on learning outcomes compared to traditional direct-answer systems. Future developments aim to incorporate personalization based on individual learning progress.

II. Related work

Standard AI educational systems typically provide immediate answers, which, while efficient, often undermine the student's problem-solving skills. Retrieval-Augmented Generation models have recently been employed in knowledge-intensive tasks to enhance factual correctness and adaptability [1]. However, the application of RAG combined with interactive tutoring strategies remains limited. Studies such as "Socratic Tutoring Strategies for AI Systems" [6] demonstrated that systems guiding students through questions lead to greater engagement and deeper understanding. Additionally, "Learning with OpenAI Codex: Student Outcomes in Programming Tasks" [7] showed that guided hint-based tutoring improves problem-solving skills compared to direct-answer methods.

Although previous studies have demonstrated the effectiveness of Socratic questioning and guided tutoring approaches [6], [7], the integration of Retrieval-Augmented Generation with personalized, curriculum-aware student interaction remains underexplored. Existing systems focus on direct content delivery. Therefore, this project aims to bridge this gap by combining RAG-based retrieval with dynamic tutoring strategies specifically designed for university education.

III. System design

The AI-tutor is realised as a Retrieval-Augmented Generation (RAG) pipeline with clearly separated responsibilities (Fig. 1). The components are orchestrated by a low-code workflow engine (implemented in Flowise, although the design is platform-agnostic). The following modules constitute the core architecture:

A. Memory Layer

This module maintains conversational context across multiple dialogue turns. Memory retention is critical to providing personalized, coherent interactions, especially when students build upon previous questions. Each user input is appended to a session-specific memory buffer that is managed through Flowise nodes, ensuring that historical context is correctly injected into subsequent retrievals and prompt formulations [2], [8].

B. Retriever

Educational materials are embedded using OpenAI's text-embedding-3-small model, generating dense vector representations of course content. These embeddings are indexed into a Pinecone vector database under the namespace book. For retrieval, Top-K similarity search is applied, typically fetching 4 nearest documents based on cosine similarity, calculated as:

$$s(q,d) = cos(e_q, e_d) = \frac{e_q \cdot e_d}{\| e_q \| \| e_d \|}$$

where e_q and e_d are the query and document embeddings, respectively.

The use of Pinecone enables scalable and efficient semantic search across large knowledge bases, allowing real-time retrieval even with growing datasets.

C. RAG Controller

The Retrieval-Augmented Generation (RAG) Controller orchestrates the pipeline by merging the retrieved documents with the user's query into an augmented prompt. This hybrid mechanism ensures that the language model's output remains grounded in validated academic content, thereby enhancing factual reliability and maintaining alignment with the course curriculum.

D. Generation Engine

OpenAI GPT models are employed within LLM nodes to synthesize answers from retrieved context. Rather than outputting direct solutions, the generation process emphasizes Socratic questioning and scaffolding: guiding students to discover answers through exploration and reasoning. Prompt templates strategically encourage elaboration, reflection, and multiple-step thinking.

E. Factual Consistency Validator

To minimize the risk of inaccurate or unsupported outputs, the system implements a lightweight consistency validation mechanism. Specifically, it generates multiple candidate responses and performs cross-verification using semantic similarity metrics. If the outputs diverge beyond a predefined threshold (e.g., cosine similarity < 0.8 between paraphrased answers), the system initiates a fallback procedure that prompts the user to clarify or rephrase the query. This internal verification step reinforces the factual integrity of the AI tutor's responses, particularly in cases where retrieved content from the top-K documents may be insufficient or ambiguous.

F. Adaptive Tutor Logic

The system architecture includes a framework for adaptive dialogue flow based on student interaction patterns. Although full behavioral interpretation is not yet implemented, the design supports future extensions where the tutor could adjust responses based on indicators such as keywords, uncertainty expressions, or request types. For instance, confident input could trigger advanced content, while hesitation may prompt additional guidance. This adaptability is currently a conceptual feature intended to support personalization in later system versions.

G. Interface Layer

The system interface is built using the native Flowise web frontend, a low-code platform for configuring conversational AI pipelines. It enables rapid prototyping and integration of key components such as memory, retrieval, and LLMs. The backend exposes RESTful endpoints (e.g., /chat), allowing integration with platforms like Telegram or Microsoft Teams. While the current interface supports core interaction and testing, it may be replaced in the future by a custom React-based UI for enhanced scalability and usability.

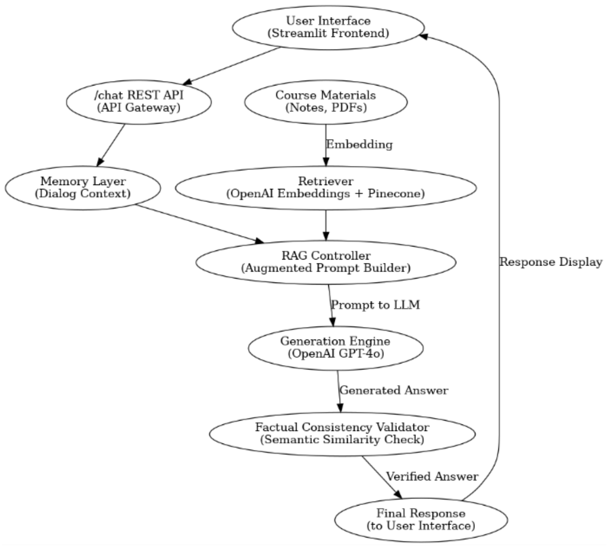
User Interface (Streamlit Frontend)
/chat REST API (API Gateway)
Course Materials (Notes, PDFs)
Embedding
Memory Layer (Dialog Context)
Retriever (OpenAI Embeddings + Pinecone)
RAG Controller (Augmented Prompt Builder)
Response Display
Prompt to LLM
Generation Engine (OpenAI GPT-4o)
Generated Answer
Factual Consistency Validator (Semantic Similarity Check)
Verified Answer
Final Response (to User Interface)

Figure 1. System Architecture of the Personalized AI Tutor

IV. IMPLEMENTATION

The AI tutoring assistant is implemented using a modular low-code architecture built in Flowise, chosen for its flexibility, rapid prototyping capabilities, and visual workflow orchestration [2]. Each major function of the system is represented as an independent node, enabling maintainability, extensibility, and easy debugging during development.

- Embedding Educational Content: The educational materials were embedded into dense vector representations using OpenAI's text-embedding-3-small model [2], producing 768-dimensional vectors. The choice of this model was driven by its balance between performance, embedding quality, and cost-efficiency compared to larger embedding models. These vectors are stored in Pinecone, a scalable and production-grade vector database [10], selected for its ability to perform real-time semantic searches with low-latency retrieval even as the dataset grows.
- Retrieval Mechanism: Similarity search is based on Top-K cosine distance retrieval. For each query submitted by a student, the retriever fetches the four most semantically relevant documents. Cosine similarity was chosen as the metric due to its established effectiveness in embedding spaces where magnitude is less important than orientation [5]. Maintaining a top-K threshold of 4 strikes a balance between retrieval precision and prompt size, preventing information overload while ensuring contextual relevance.
- RAG Management and Prompt Construction: The RAG Manager aggregates the retrieved documents and augments them with the current user query and relevant conversational memory. This prompt construction strategy ensures that the generative model's responses are grounded in validated educational content, thereby enhancing factual reliability and maintaining curriculum alignment [1]. The decision to dynamically combine retrieval results at query time, rather than at

indexing time, increases system flexibility and allows seamless incorporation of updated or newly added course materials.

- Generative Engine (LLM Node): The augmented prompt is passed to OpenAI's GPT-4o-2024-04 model [2]. Temperature is set at 0.7 to encourage a moderate degree of creativity without sacrificing factual consistency. The maximum output token limit of 512 ensures concise and manageable responses. GPT-4o was selected for its improved multi-turn coherence and factual grounding capabilities compared to earlier models [4].
- Factual Consistency Validator Implementation: To improve factual reliability, the system incorporates a lightweight factual consistency validator. This component generates two alternative responses and evaluates their semantic alignment using cosine similarity. If the similarity score falls below a threshold of 0.8, the system interprets this as a potential inconsistency and withholds a definitive answer, instead prompting the user to rephrase or clarify the query [5]. This mechanism adds an additional layer of factual control, reducing the likelihood of generating unsupported or misleading content.
- Frontend and User Interface: The current frontend is implemented using the built-in Flowise interface, which enables low-code configuration of conversational agents and interaction pipelines. While visually minimalistic, the interface allows real-time engagement with the system and supports token-by-token response streaming, providing a smooth and interactive chat experience. This setup has proven sufficient for academic prototyping and practical validation of the AI tutor's core functionalities.
- Backend and API Exposure: The backend workflow, orchestrated in Flowise, exposes a RESTful API endpoint (/chat), enabling future integration with external platforms such as Moodle, Microsoft Teams, or Slack [3]. This API-driven design ensures platform-agnostic deployment potential.
- Containerization and Deployment: The application is deployed using Docker, which ensures reproducibility and simplifies setup across different environments. Flowise, the central orchestration engine, is containerized, allowing the entire pipeline—including memory management, retrieval, and response generation—to run consistently across platforms. External services such as OpenAI APIs and Pinecone are accessed remotely and do not require local deployment. This approach supports modular development and provides a practical foundation for further integration and scalability.

V. Evaluation

The preliminary evaluation of the AI tutor was conducted through a series of simulated student interactions using representative queries from university-level mathematics, telecommunications, and computer science courses. The primary goals of the evaluation were to assess the system's ability to maintain conversational coherence, provide curriculum-aligned responses, stimulate critical thinking, and ensure factual consistency in the delivered content.

Coherence and Context Management: Throughout multi-turn conversations, the Memory Layer consistently preserved dialogue history, enabling the system to reference prior exchanges coherently. This behaviour corresponds with findings from prior research emphasizing the role of effective context maintenance in AI-supported educational environments [6].

Retrieval and Relevance: The Retriever demonstrated a high degree of retrieval relevance, consistently returning documents closely aligned with the user's queries. The use of dense semantic search techniques, as described in [9], contributed to the system's ability to select contextually appropriate supporting materials.

Guided Reasoning and Student Engagement: The Generation Engine emphasized Socratic questioning, scaffolding, and elaborative prompting rather than providing direct answers. This strategy aligns with educational methodologies that highlight the benefits of guided discovery learning to enhance student comprehension and retention [6], [8].

Factual Consistency Assurance: A consistency verification mechanism was employed to enhance factual reliability. The system generated multiple candidate responses and evaluated their semantic similarity. In cases where significant divergence was identified (cosine similarity < 0.8), the system prompted users to rephrase or clarify their queries, thereby minimizing the risk of inaccurate information propagation [5].

Observations and Considerations: As this evaluation was conducted through simulated interactions, the findings are indicative but not conclusive. Real-world experimental validation with diverse student populations remains necessary to fully verify the system's educational impact. Additionally, opportunities exist to refine the system's handling of ambiguous and interdisciplinary queries by enhancing retrieval strategies and memory context management.

In conclusion, the initial evaluation suggests that the AI tutor effectively maintains dialogue coherence, ensures factual grounding, fosters critical thinking, and provides interactive performance suitable for educational deployment. Future empirical studies will be essential to substantiate these findings under practical learning conditions.

VI. Future work

Building on the promising outcomes of the initial evaluation, several avenues for future development are envisioned to enhance the adaptability and pedagogical effectiveness of the AI tutoring system.

A key direction involves implementing dynamic adaptation to the student's current level of understanding. By analysing prior interactions and responses, the tutor could adjust the difficulty and depth of questions, progressively tailoring its guidance to the learner's individual trajectory.

Another anticipated enhancement is the development of a targeted assignment generation module. This feature would enable the tutor to offer exercises and problem sets that correspond to the student's identified strengths and weaknesses, fostering structured and personalized knowledge reinforcement.

Additionally, the introduction of a screening and performance tracking mechanism is proposed. Periodic analysis of student input could support long-term progress monitoring, early detection of conceptual gaps, and more precise, individualized feedback.

Curriculum-aware timing is also being considered. The AI tutor may synchronize its instructional content with the student's current academic calendar position—e.g., avoiding the introduction of material beyond the current week of instruction—to support effective pacing and reduce cognitive overload.

Further technical improvements are planned for the Retrieval and Memory Layers. Incorporating methods such as Dense Passage Retrieval (DPR) [9] could enhance retrieval accuracy, while refined memory mechanisms may enable long-term tracking across subjects and sessions.

Finally, research into real-time sentiment analysis and the application of Reinforcement Learning from Human Feedback (RLHF) [7] could support continuous optimization of tutoring strategies based on user behaviour and engagement signals.

Collectively, these potential enhancements aim to evolve the AI tutor into a robust, curriculum-aware, and student-cantered platform for delivering personalized learning support in diverse educational settings.

VII. Conclusion

This paper presented the design and preliminary evaluation of a personalized AI tutoring assistant for university education, based on the Retrieval-Augmented Generation (RAG) framework. The system integrates a modular architecture comprising memory management, semantic retrieval, curriculum-grounded prompt augmentation, and factual consistency assurance mechanisms.

Preliminary observations indicate that the tutor successfully fosters reflective reasoning, supports student engagement, and maintains factual accuracy in responses. Through the strategic use of Socratic dialogue, adaptive scaffolding, and dynamic retrieval, the AI tutor moves beyond traditional direct-answer systems, aligning with best practices in educational AI design [6], [8].

While initial results are promising, formal empirical validation remains a priority for future research. Several potential enhancements, such as student-level adaptation, targeted assignment generation, performance monitoring, and semester-aligned curriculum pacing, have been outlined to further personalize and improve the learning experience.

Overall, the proposed AI tutoring assistant demonstrates the feasibility and potential benefits of leveraging advanced retrieval and generation techniques to support personalized, curriculum-aware, and interactive education in higher learning environments.

Acknowledgment

This paper was supported by NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2), CYB-FUT (ERASMUS+), and EULiST (ERASMUS).

References

[1] P. Lewis, E. Perez, A. Karpukhin, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," 2020.

[2] OpenAI, "GPT-4 Technical Report," 2023.

[3] J. Smith, "Socratic Dialogue and AI: Enhancing Learning through Questioning," Journal of Educational AI, 2022.

[4] T. Brown, "Scaling Laws for Neural Language Models," 2020.

[5] P. Rajpurkar, J. Zhang, "Assessing the Factual Accuracy of Generative Models," 2023.

[6] A. Author, B. Author, "Socratic Tutoring Strategies for AI Systems," 2022.

[7] C. Author, D. Author, "Learning with OpenAI Codex: Student Outcomes in Programming Tasks," 2023.

[8] D. Rus et al., "AI-Assisted Education: From Theory to Practice," Science Robotics, 2021.

[9] K. Guu et al., "REALM: Retrieval-Augmented Language Model Pre-Training," ICML 2020.

[10] Pinecone Systems, "Pinecone Documentation: Scalable Vector Database for Machine Learning," 2024