iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.73970

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

RAG-Based AI Chatbot for Student and Institutional Assistance

Nisanth P, Arohan A R, Adhithyan P C, Muhammed Suhail, Shahzad Bin Muhammed, Linsa V U
Artificial Intelligence and Machine Learning Department Vidya Academy of Science and Technology, Thrissur, India

Abstract: ***In the contemporary higher education setting of heightened student involvement and the need for immediate information, institutions must counteract the challenge of deliv- ering effective and convenient customer service. The steady flow of questions about admissions, academics, and campus services overwhelms support personnel and requires responsive solutions to better support the student experience. Filling this void, the "RAG-Based AI Chatbot for College Customer Support" presents a solution, enabling students, parents, and staff to receive instant support and information. Building on Retrieval-Augmented Gen- eration (RAG) combined with Large Language Models (LLMs), the chatbot leverages a powerful architecture to tap into related information from reliable sources like college databases, web pages, and documents. Upon receiving user queries, the system efficiently identifies and ranks pertinent information, enabling the LLM to generate accurate and context-aware responses. This innovative solution offers 24/7 support, streamlines operational processes, and reduces the workload on support teams, fostering a more efficient and satisfying college experience.***

I. INTRODUCTION

The swift development of artificial intelligence (AI) is profoundly transforming interactions in educational institutions. There is an increasing need for effective, responsive systems to manage the various questions from students, employees, and potential applicants. AI-driven chatbots present a viable solution to answer typical questions and offer advice, promoting interaction with the college. Despite this, most of the current chatbots, though helpful, tend to base their responses on broad pre-trained knowledge and can fail to constantly deliver answers that are perfectly accurate and strictly derived from the institution's unique, current documents and policies. This leaves a void where responses can fall short of required context, accuracy, or reference to official sources.

This constraint emphasizes the necessity of more advanced methods that guarantee information reliability. Evidence has demonstrated the efficacy of chatbots in educational institutions, assisting with domains such as admissions and information distribution [1]. Nevertheless, guaranteeing factual correctness linked directly with institutional knowledge is still a challenge for traditional chatbot designs.

The College support Chatbot project seeks to close this gap through the application of a Retrieval-Augmented Generation (RAG) architecture in the context of college customer service. This makes use of top-of-the-line open-source packages and locally installed models, including Python as the central language, Streamlit as the user interface, Ollama for the execution of local language and embedding models, ChromaDB as the vector store, and Langchain, Sentence Transformers, Pandas, and PyMuPDF as document handling and pipeline coordination libraries.

The essence of the proposed chatbot's innovation is its RAG pipeline:

1) Institutional documents (PDFs, Excel spreadsheets) are processed, chunked, and turned into semantic embeddings via the nomic-embed-text model
2) These are stored in ChromaDB, building a searchable knowledge base purely based on college-specific data.
3) When a user query is input through the Streamlit inter- face, it's embedded, and corresponding text chunks are retrieved from ChromaDB.
4) When a user query is input through the Streamlit inter- face, it's embedded, and corresponding text chunks are retrieved from ChromaDB.
5) A CrossEncoder model (cross-encoder/ms-marco- MiniLM-L-6-v2) re-ranks these chunks to achieve optimal contextual relevance with respect to the query.
6) The locally executed Gemma 3:4B large language model finally produces a response, directly employing the re- trieved and re-ranked text as context to ensure that the answer remains grounded in the given documents.

This approach, building on the principles of RAG , aims to provide correct, context-specific answers by having answers map directly onto confirmed institutional data. Having local models through Ollama also presents potential benefits around data privacy and control. By improving search relevance via re-ranking and anchoring generation in retrieved facts, The proposed chatbot seeks to offer a more trustworthy and dependable AI support tool than chatbots based on generalized LLM knowledge. This is in line with the requirement for AI systems in education that are not only conversational but also factually reliable within their particular operational context.

II. RELATED WORK

The use of conversational AI, specifically chatbots, in schools has changed dramatically over the last few years. Initial implementations tended to involve rule-based systems or basic NLU methods to respond to FAQs or offer basic navigational support on university websites with the purpose of minimizing administrative workload . As helpful as they were, such systems tended not to have proper conversational capabilities and were unable to handle questions beyond their set parameters.

Other newer approaches have utilized NLU advancements along with cloud-based AI utilities to implement more advanced college enquiry chatbots. [2], implemented a chatbot with Microsoft Azure AI services such as LUIS for intent detection and QnAMaker for FAQ handling, with higher accuracy compared to rule-based and Rasa NLU implementations for typical college subjects [3]. Proposals such as Beigh Jahangir also seek to combine technologies such as OpenAI's GPT and vector databases to produce context-sensitive answers . Still, it is a pressing challenge to maintain the factual validity and appropriateness of information supplied by such systems, particularly in the context of particular institutional policies or constantly changing content.

The emergence of Large Language Models (LLMs) with considerable power has provided new avenues for extremely fluent conversational agents. However, general- purpose LLMs, when used directly, may have limitations like "hallucinations" (producing factually inaccurate information) and unavailability of access to specialized, current, or private institutional knowledge [1], [2]. Retrieval-Augmented Generation (RAG) has become a leading method to overcome these limitations . RAG frameworks ground LLM responses by first retrieving relevant information from a specific knowledge base (e.g., institutional documents) and then providing this retrieved context to the LLM along with the user's query during the generation phase.

RAG has seen increasing application in educational contexts. Xu Liu applied a RAG-based chatbot for university admission inquiries with the GLM-4 model to extract and process information from the official university website to establish a static knowledge base [1]. Their contribution is to point out the usefulness of RAG in minimizing hallucinations for certain institutional information. Soliman et al. created the "BiWi AI Tutor," a RAG system based on GPT-3.5 and LangChain for scalable educational assistance based on lecture documents [2]. Their contribution is most relevant as they specifically include a reranker (Cohere) to re-rank the retrieved context prior to input into the LLM in an attempt to enhance response accuracy. Our effort is an extension of these works but differs in that it centers on a dynamic knowledge base generated from user-uploaded documents (PDF/Excel) and applying locally hosted open-source models inside the RAG pipeline.

The key to improving RAG performance lies in guaranteeing the quality and relevance of the retrieved context presented to the LLM. Standard vector similarity search may return documents semantically similar but not necessarily directly relevant to the particular subtlety of the user's question. Re-ranking methods, usually based on more computationally costly Cross-Encoder models, respond by re-ranking the first set of returned documents specifically with respect to the user query to produce a more accurate context ranking [7]. The beneficial effect of re-ranking on RAG system performance, as shown by Soliman et al. [4] in an educational RAG application and talked about by Zhuang et al. [7] in conversation generation, encourages its introduction into our proposed Chatbot system for improvement of the factuality and specificity of the generated responses.

While RAG involves retrieving stored knowledge, fine- tuning is another way of making LLMs domain-specific by retraining the actual model on domain data, as studied by Kirtil et al. for the case of tourist chatbots [10]. RAG was selected for our project because it can be easily updated with new knowledge without having to retrain the model and has a more solid basis in given factual texts.

In addition, a significant amount of RAG development relies on large, cloud-hosted proprietary models. Our method investigates the possibility of using locally hosted open- source models (through Ollama) for both embedding and generation. This provides potential benefits in terms of data privacy, personalization, and operational expense, which fit into situations where use of external cloud APIs can be undesirable or impossible.

The proposed Chatbot in this work advances the field by combining dynamic document ingestion, local model execution, and context re-ranking in a RAG architecture designed for adaptable college customer support.

III. METHODOLOGY

The first step of this project focused on building the chatbot's knowledge base. Information was carefully collected from the official college website and supported with detailed forms provided by teachers, defining the seed dataset for the system.

Switching to the central Retrieval-Augmented Generation (RAG) architecture, the gathered text data was prepared. The nomic-embed-text embedding model, implemented via OllamaEmbeddingFunction in the ollama framework, was used to translate the text to vector representations. These embeddings were then stored and indexed in a ChromaDB vector database to support effective semantic retrieval. The retrieval process was based on ollama and was further optimized with a CrossEncoder model from the sentence- transformers library to boost the relevance ranking of retrieved documents.

For the generation side, the Gemma3:4B large language model, also handled by ollama, was used. This model generated coherent and contextually sound responses based on the information recovered from the knowledge base in the retrieval phase. User interaction was made via a web interface developed with the Streamlit framework, offering a convenient way for users to query the chatbot. Lastly, evaluation processes were implemented to validate the efficacy of the chatbot. Performance was mainly gauged manually, checking the generated answers against the source data for correctness, and tracking response generation time to measure system efficiency.

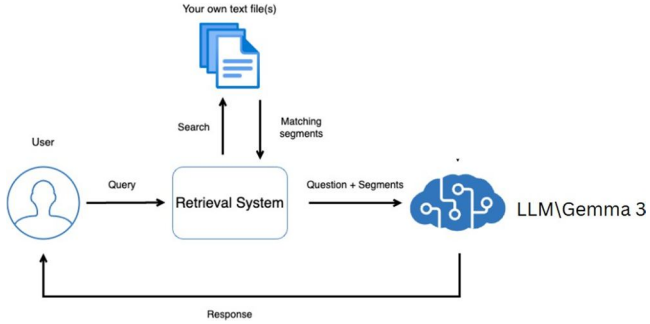


Fig. 1. System Architecture

IV. PHASES OF THE PROJECT

The development process for the RAG-based AI chatbot involved the following key stages:

1) Dependency Management and Environment Setup
2) Knowledge Base Creation and Pre-processing
3) Vector Em- bedding and Indexing
4) Implementation of the RAG pipeline
5) User Interface (UI) Development
6) Testing and Evaluation
7) User Feedback Collection and Analysis
8) Performance and Resource Optimization
9) Continuous Improvement and Knowledge Base Updates

A. Dependency Management and Environment Setup

The first step was to create a stable development platform by installing all the necessary Python packages, namely streamlit, ollama, chromadb, and sentence-transformers, within a con- tained virtual environment. Setup of the local Ollama service was also done to ensure the appropriate embedding (nomic- embed-text) and language (Gemma3:4B) models were down- loaded and available, reducing possible future compatibility problems.

B. Knowledge Base Creation and Pre-processing

Source data was collected from the assigned college web- site material and supplied forms. The raw textual data was subjected to necessary preprocessing, such as cleaning to eliminate artifacts and normalization for uniformity. Chunking was a crucial aspect of this phase, in which big documents were divided into smaller, semantically significant units to maximize the following retrieval process for relevance and specificity.

C. Vector Embedding and Indexing

The preprocessed text chunks were converted into numerical vector representations through the nomic-embed-text embed- ding model, accessed through the local Ollama service. These high-dimensional vectors, retaining the semantic content of the text, were then stored along with their respective text chunks and indexed in a persistent ChromaDB vector database, allowing for fast semantic similarity searches.

D. Implementation of the RAG pipeline

This involved writing the central Python code that orchestrates the Retrieve-Augment-Generate process:

- Retrieval: Implementing logic to take a user's query, embed it using the same nomic-embed-text model, and query ChromaDB to retrieve the n most similar text chunks based on vector similarity.
- Re-ranking: Considering the originally retrieved chunks and applying the CrossEncoder model to re-estimate their relevance particularly against the user query, returning a more fine-grained ordering and choosing the top-k (e.g., top 3) most contextually suitable chunks.
- Generation: Building a prompt for the Gemma3:4B LLM (through Ollama) with the original user query supplemented by the re-ranked, retrieved text chunks as context. The prompt is to tell the LLM to produce an answer in response to the given context.

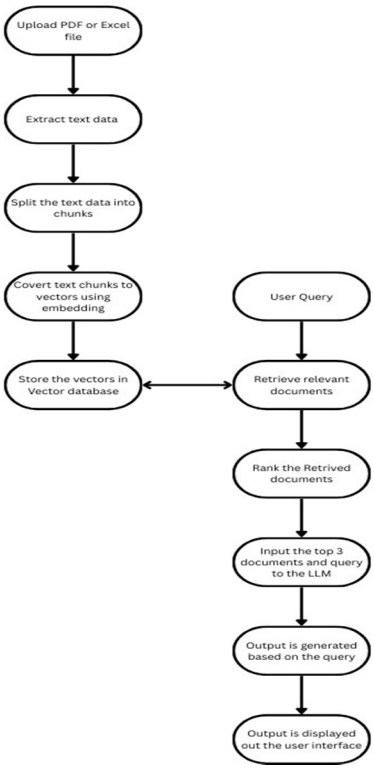


Fig. 2. Chatbot Dataflow Diagram

E. User Interface (UI) Development

A web user interface was created interactively with the Streamlit framework. The UI has necessary user interaction components like an area for inputting text queries, a submit button, and a response area for the chatbot's messages, in- cluding streamed output. Callback functionality was applied to map user actions on the UI to the running of the backend RAG pipeline.

F. Testing and Evaluation

Extensive testing was done to ensure the functionality and quality of the chatbot. This entailed designing test questions to encompass anticipated topics and manually checking for correctness and factuality of produced answers against the source knowledge base. Relevance of retrieved documents and response latency of the system were likewise systematically tested to guarantee reliability and performance.

G. User Feedback Collection and Analysis

There were attempts to collect feedback on the usability and performance of the chatbot, perhaps by informal demonstra- tions or face-to-face feedback sessions. This feedback analysis yielded useful information about user satisfaction and areas that may need possible tweaking, in order to make the system meet real-world user requirements.

H. Performance and Resource Optimization

On the basis of test results, in terms of latency mea- surements, the RAG pipeline stages were examined in order to determine and resolve bottlenecks in performance. This included code optimization, possibly optimizing database queries, and keeping an eye on the computational resource usage (CPU/RAM) of Streamlit and the Ollama service to optimize operation.

I. Continuous Improvement and Knowledge Base Updates

Understanding the incremental nature of development, pro- visions were made for frequent improvements based on evalua- tion results and feedback. Significantly, this involved establish- ing a process for continuous updating of the knowledge base using new or updated college information and then re-indexing the material to ensure that the chatbot remains accurate and relevant over a period of time.

V. RESULT

The proposed Chatbot system appears as a web application developed with Streamlit, offering a user-friendly interface to engage with a powerful Retrieval-Augmented Generation (RAG) backend executed locally. In contrast to pre-defined knowledge chatbots, Chatbot's functionality is solely based on the documents uploaded by the user, making responses contextually rooted within the given information. The use of the system on local models through Ollama (for generation and embeddings) and local storage through ChromaDB ensures that the performance aspects, i.e., response time and document processing, of the system directly depend upon the machine upon which it's executed.

When the Streamlit application is launched, the user initially starts the knowledge-building process. Through the sidebar interface, the user chooses and loads a pertinent document (PDF or Excel file) with the information to be queried. Clicking the "Process" button initiates the backend pipeline: the application fetches text, breaks it up into manageable sections with Langchain's text splitter, sends the text to the local Ollama service to have it generate embeddings using the nomic-embed-text model, and stores securely these embeddings along with the text chunks in the persistent ChromaDB vector store. The user gets a confirmation within the UI only after this ingestion process is over, indicating that the chatbot is ready to respond based on the content of that particular document.

After creating a knowledge base from one or several documents, the user communicates with the chatbot within the main application region. They enter their natural language question into the provided text field and press the "Ask" button. This step launches the central RAG pipeline: the system injects the query (via Ollama/Nomic), conducts a semantic search in ChromaDB to get potentially relevant text segments, prunes this set using the Sentence Transformers Cross-Encoder for greater relevance, and lastly, passes the original query and most relevant context to the gemma3 model through the local Ollama service. The generated answer, set up to be entirely reliant on the context provided, is then piped back to the user interface, offering an interactive and dynamic experience as opposed to waiting until the entire response has been generated. Should the context returned not be adequate enough to answer the query, the system is engineered to make clear that limitation.

In general, the proposed Chatbot offers a robust and targeted tool for users who require extracting certain information from specified documents without searching manually. The RAG architecture effectively anchors the LLM's output in the uploaded material, striving for factual correctness compared to the source material. The Streamlit interface provides easy-to-use usability for both document handling and querying, while the streaming output improves the user's sense of responsiveness. This makes the system an invaluable prototype for focused information retrieval in use cases such as searching specific college catalogs, policy reports, or custom knowledge repositories.

VI. CONCLUSION

This research endeavored to develop and test the feasibility of a Retrieval-Augmented Generation (RAG) grounded AI chatbot, that would deliver accurate, context-specific assistance using particular documents from a collegiate setting. Through combining local open-source models with vector storage, we sought to build an entity whose answers are strictly founded on validated institutional sources, thus reducing the tendency for hallucination usually seen within general-purpose large language models (LLMs).

The workflow involved processing varied college files (PDF, Excel), generating semantic embeddings with nomic-embed- text through Ollama, indexing these in a local instance of ChromaDB, and running a RAG pipeline within a Streamlit app. This pipeline had retrieval, CrossEncoder- based re-ranking, and answer generation by a locally served Gemma 3:4B model, only constrained by the retrieved context. The qualitative analysis, based on manual checking against source documents, showed the ability of the chatbot to provide responses factually consistent with the given knowledge base. The presence of the re-ranking step was seen to have the effect of improving contextual informativeness of information provided to the LLM. In addition, the project also demonstrated the effectiveness of using totally locally hosted models and databases (Ollama, ChromaDB) for building a domain-specific AI assistance tool, presenting possible advantages for data privacy and control. Some areas, nonetheless, need investigation and enhancement. One major limitation is the static nature of the knowledge base implemented. The accuracy of the chatbot depends directly on the update timeliness of the processed documents; it cannot track real-time updates unless the base vector store is updated. Future developments should involve working on developing an automated pipeline for tracking source document change and effectively updating the ChromaDB embeddings to keep them as relevant in the future.

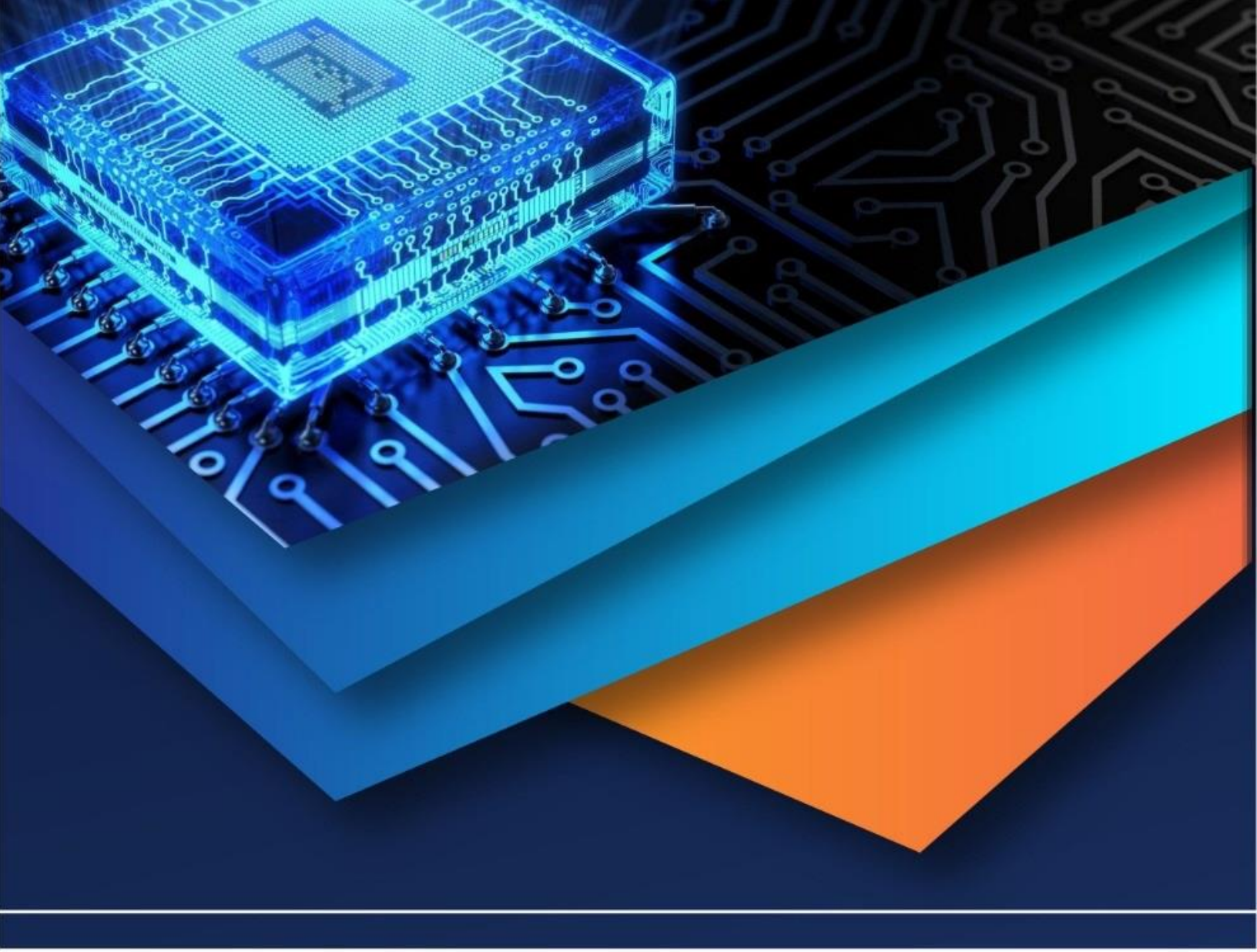
Another area for improvement is the assessment methodology. The existing assessment depended on manual verifications. The inclusion of automated evaluation systems, possibly modifying measures from tools such as RAGAS emphasizing faithfulness, answer relevance, and context accuracy, would offer more objective and scalable performance measures. In addition, formal user testing is required to collect feedback on usability and perceived effectiveness from the intended student and staff group.

Furthermore, although proving feasibility, the locally hosted models' performance (latency) and scalability on standard institutional hardware may be challenging under heavy loads. Subsequent versions may look into optimisation techniques like model quantisation, hardware acceleration, or other local model serving environments.

In summary, despite these limitations, this project illustrates the tremendous potential of using a RAG architecture with locally controlled, open-source modules to construct robust, institution-specific AI support systems. Through the devel- opment of the proposed chatbot framework, our work adds a practical solution based on available tools, illustrating a way towards constructing trustworthy AI assistants that reduce hallucination by adhering strictly to curated knowledge sources in the education domain.

REFERENCES

[1] Xu, Liwei and Liu, Jiarui, "A Chat Bot for Enrollment of Xi'an Jiaotong- Liverpool University Based on RAG"2024 8th International Workshop on Control Engineering and Advanced Algorithms (IWCEAA),IEEE, 2024.
[2] D. Patel, N. Shetty, P. Kapasi, and I. Kangriwala, "College enquiry chatbot using conversational AI", International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 11, no. 5, p. 2023, 2023.
[3] Mohammad Shahid Beigh, Shahida Jahangir, "AI-BASED CHATBOT FOR EDUCATIONAL INSTITUTES," in ResearchGate,June 2024.
[4] Kumar Shivam; Khan Saud; Manav Sharma; Saurav Vashishth; Sheetal Patil, "Chatbot for College Website,"IJCAT - International Journal of Computing and Technology, Volume 5, Issue 6, June 2018.
[5] C. V. Misischia, F. Poecze, and C. Strauss, "Chatbots in customer service: Their relevance and impact on service quality ," Procedia Computer Science, vol. 201, pp. 421–428, 2022.
[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Ku¨ttler, M. Lewis, W.-t. Yih, T. Rockta¨schel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," arXiv preprint arXiv:2005.11401, 2020x.
[7] H. Soliman, H. Kotte, M. Kravc˘´ık, N. Pengel, and N. Duong-Trung," Retrieval-augmented chatbots for scalable educational support in higher education", International Workshop on Generative AI for Learning Analytics, 2025.
[8] ˙Ismail Go¨kay Kırtıl, Beykan C¸ izel, ˙Ismail Uzut, and Serdar Uzun, "Bridging the Gap: Fine-Tuning Artificial Intelligence (AI) Chatbots for Tourism," Conference Paper, May 2024
[9] Marcondes, Francisco S and Gala, Adelino and Magalha˜es, Renata and Perez de Britto, Fernando and Dura˜es, Dalila and Novais, Paulo, "Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs," 2023.
[10] ˙Ismail Go¨kay Kırtıl1, Beykan C¸ izel2, ˙Ismail Uzut3, and Serdar Uzun1,"Bridging the Gap: Fine-Tuning Artificial Intelligence (AI) Chatbots for Tourism",The Conference on Managing Tourism Across ContinentsAt:˙Istanbul May 2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY