Personalized E-Learning Course Recommendations: A Chatbot Approach Using LangChain

Mavin Sao
Graduate School of Data Science,
Chonnam National University,
Gwangju, South Korea
kr225043@jnu.ac.kr

Hoi-Jeong Lim *
Graduate School of Data Science,
Public Data Analytics Center,
Chonnam National University,
Gwangju, South Korea
hjlim@jnu.ac.kr

Abstract— **This study presented a personalized e-learning method utilizing an intelligent chatbot for course recommendations. By integrating Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG), we developed a recommendation system that aligned with user learning preferences and goals. The system leveraged a pretrained embeddings model and vector databases to access a dataset of 16,223 courses from 40 different subjects, sourced from platforms like edX and Coursera. We evaluated the chatbot's responses by analyzing query relevance and context consistency score matrices. The evaluation yielded an average score of 0.77 in both relevance and consistency metrics. We also conducted response evaluation through human surveys. The survey focused on response relevance, comprehensibility, readability, accuracy, and helpfulness of explanations, achieving high satisfaction ratings (4.10-4.42 out of 5.0), which provided a more comprehensive approach to evaluating our RAG responses. Finally, the chatbot was implemented as a Streamlit web application, enabling user interaction and feedback collection for future improvements.**

Keywords—component; LLM; chatbot; e-learning; retrival-augmentation-generation

I. INTRODUCTION

The educational landscape has evolved significantly, with e-learning platforms providing widespread access to information [1]. However, students face many challenges in navigating through numerous courses to find options that match their requirements. The traditional approach requires users to search independently across multiple e-learning platforms—a process that is both overwhelming and time-consuming. While recent advances in Large Language Models (LLMs) can assist in course discovery, they may generate unreliable responses containing incorrect information or invalid links [2].

To address these challenges, we propose an intelligent e-learning course recommendation chatbot application that employs retrieval augmentation generation techniques. Our implementation incorporates careful prompt engineering to enhance context relevance and query consistency. This paper details the design, implementation, and evaluation of a chatbot intended to transform how learners access online educational resources. Our methodology leverages recent advances in LLMs, incorporating the LangChain framework - an open-source development framework that simplifies the creation of LLM-powered applications by providing reusable components and tools for common operations. LangChain enables developers to build complex chains of operations that can handle tasks like document processing, question-answering, and summarization with minimal effort.

The system uses Retrieval-Augmented Generation (RAG) technique, and FAISS datastore to process user queries and deliver tailored course recommendations. It integrates web scraping for course information collection, data processing for embedding generation, and prompt engineering for personalized suggestions, creating a comprehensive solution that addresses learner needs beyond basic keyword matching.

The main contributions of this study are as follows:

- **Novel RAG Implementation for Course Recommendations**: This study presents the first application of Retrieval-Augmented Generation with LLMs for personalized course recommendations, utilizing a comprehensive dataset of 16,223 courses across 40 subject areas.
- **Optimized Technical Architecture**: The system implements FAISS vector storage integrated with the 'all-MiniLM-L6-v2' embedding model, enabling efficient and precise semantic matching between user queries and course content.
- **Empirical Validation**: The system demonstrated robust performance metrics in both query relevance and context consistency. Student evaluation yielded high satisfaction ratings (4.10-4.42 out of 5.0) across multiple dimensions, including comprehensibility, accuracy, and helpfulness.
- **Practical Application**: The implementation features a user-friendly chatbot interface developed using Streamlit, demonstrating the system's viability in educational settings.

* Corresponding Author: Hoi-Jeong Lim

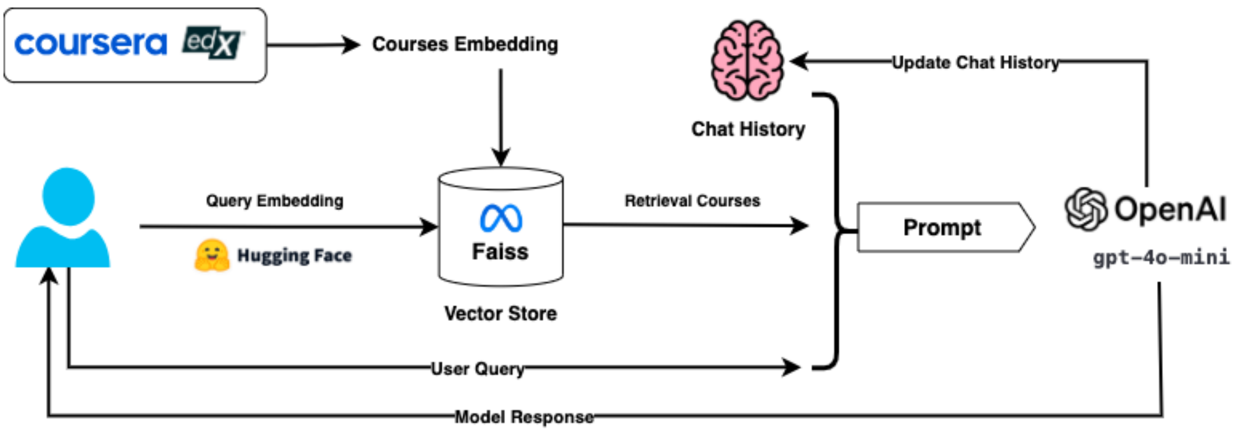


FIGURE 1. OVERALL PIPELINE OF THE COURSE RECOMMENDATION CHATBOT

```
Input:  Topics (list of course topics to scrape)
Output: CSV file containing course information

function ScrapeCourses(topics)
    webDriver ← InitializeWebDriver()
    allCourses ← empty list

    for each topic in topics do
        totalPages ← GetTotalPages(webDriver, topic)

        for pageNumber from 1 to totalPages do
            courseURLs ← GetCourseURLs(webDriver, topic, pageNumber)

            for each url in courseURLs do
                courseInfo ← ExtractCourseInfo(webDriver, url)
                allCourses.append(courseInfo)
            end for
        end for
    end for

    SaveToCSV(allCourses, "coursera_courses.csv")
    webDriver.close()
end function
```

FIGURE 2. WEB SCRAPING ALGORITHM FOR COURSE DATA EXTRACTION

II. RELATED WORK

In recent years, Large Language Models (LLMs) have grown exponentially in capability and strength [3]. Organizations increasingly integrate these models into their operations to enhance decision-making and quality outcomes. To utilize LLMs with private datasets, a technique called Retrieval Augmented Generation (RAG) is often employed [4]. RAG is a process that combines data retrieval from a specific dataset as context with the generative capabilities of LLMs. This approach enables organizations to leverage the power of LLMs while incorporating their own private dataset.

The integration of RAG with LLMs offers several benefits. First, it allows organizations to take advantage of the vast knowledge and language understanding capabilities of pre-trained LLMs. Second, by incorporating private datasets, the generated output can be tailored to the specific domain and requirements of the organization. This customization enhances the accuracy and usefulness of the model's responses. Furthermore, RAG helps to address concerns related to data privacy and intellectual property [5]. By keeping the private dataset separate from the pre-trained LLM, organizations can maintain control over their sensitive information while still benefiting from the advanced natural language processing capabilities of the model.

In [6] study, the authors review LangChain, a novel language processing model for enhanced PDF document interaction. They examine its capabilities, architecture, and applications, evaluating its potential to improve document analysis and its impact across various domains. In another study [7], the authors propose a Retrieval-Augmented Generation (RAG) approach to enhance large language models' ability to answer challenging science-based questions. They incorporate the Wikipedia6.5M dataset and combine vector similarity retrieval with the Platypus2-70B LLM to overcome data scarcity and improve STEM subject comprehension in limited computational settings.

RAG has been successfully applied in various domains such as legal document analysis [6] and scientific question-answering [7]. In this study, we present the first implementation of RAG specifically for e-learning course recommendations. We leveraged multiple advanced technologies: Hugging Face's 'all-MiniLM-L6-v2' embedding model for semantic text representation, FAISS vector database for efficient similarity search [8], and OpenAI's GPT-4o-mini for response generation. We enhanced these technologies with optimized prompt engineering techniques to deliver high-precision course recommendations that accurately matched user queries. This combination enabled our system to effectively understand user requirements and retrieve relevant courses from our database.

III. METHODOLOGY

Below is our methodology for building our recommendation system. The overall pipeline process can be found in Fig. 1.

A. Data Collecting

To get our data, Selenium WebDriver was employed to automate the process of collecting course data from edX and Coursera platforms. Custom scraping scripts were developed to navigate through course listings and extract relevant information. The scraping process was designed with

```
As an AI course recommendation expert, provide
personalized, high-quality suggestions based
on the user's interests, goals, and
background.

Chat History: {chat_history}
User Query: {question}
Relevant Courses: {context}

Response Guidelines:
1. Tone: Warm, professional, and approachable.
2. Analysis: Consider user's query, history,
and educational needs.
3. Recommendations: For each course, include:
   - Title and institution
   - Brief overview
   - Skills to be gained
   - Key topics
   - Level, duration, language
   - Ratings (if available)
   - Course URL (if available)
4. Personalization: Explain how courses align
with user's interests and needs.
Prioritize accuracy, relevance, and user-
centricity to help users make informed
educational decisions.

Recommendation:
```

FIGURE 3. LARGE LANGUAGE MODEL PROMPT TEMPLATE

appropriate delays to avoid overloading the servers. The scraped data includes course titles, sub-information, subjects, ratings, difficulty levels, institutions, course descriptions, learning objectives, syllabi, course URLs, and skills outcomes. We collected 16,223 courses from 40 different subjects which is an comprehensive amount and diverse subjects. The extraction algorithm can be found in Fig. 2.

B. Text Embedding Process

Text embedding is a conversion technique that transforms text into a dense vector embedding. This special number represents the pattern that allows the computer to understand it. We rely on a unified embedding process for both course content and user queries, utilizing the pre-trained Sentence Transformer model "all-MiniLM-L6-v2" [9]. The all-MiniLM-L6-v2 model is a popular, lightweight tool that transforms text into a list of 384 numbers while preserving its meaning. For courses, we begin by cleaning the scraped data by removing duplicates to ensure consistency. We then combine relevant features such as title, subject, level, language, institution, learning outcomes, syllabus, and descriptions into a single text representation for each course.

User queries undergo a similar transformation, with each query converted into a dense vector embedding using the same model. By employing identical embedding techniques for both courses and queries, we ensure they occupy the same vector space, enabling direct comparisons and facilitating accurate, semantically relevant recommendations.

C. FAISS: Facebook AI Similarity Search

All embedded courses were stored in a vector store called FAISS (Facebook AI Similarity Search). The Vector Store is the specialized database which used to store dense vectors. It is designed to store high-dimensional vectors which are representations of data, especially text. To enable efficient similarity searches and swift retrieval, we leverage FAISS (Facebook AI Similarity Search) to create a vector store for our course embeddings. FAISS, designed for efficient similarity search and clustering of dense vectors, excels at processing large datasets.

In our implementation, we store course embeddings along with metadata such as title, description, and URL in an FAISS index. This approach facilitates rapid similarity searches and quick retrieval of course information, capable of handling datasets of any size, including those exceeding available RAM. We save the created index locally, allowing for fast loading and querying in subsequent sessions without the need for recomputation.

D. Prompt Template

Our recommendation process begins with a carefully crafted prompt template. In Fig. 3, The template is designed to combine retrieved course information, a structured recommendation format, and chat history. Its purpose is to guide the language model in analyzing user queries, selecting relevant courses, and providing detailed recommendations while maintaining a conversational tone. By structuring the input in this way, we ensure that the language model has all the necessary context to generate personalized and informative course recommendations.

E. Conversation Memory

To enhance personalization over time, we implement a ConversationSummaryBufferMemory. This component maintains context across multiple interactions, allowing the system to remember and refer back to previous conversations. By preserving key information from past exchanges, the system can provide more contextually relevant recommendations, taking into account the user's evolving interests and previous inquiries. This feature significantly improves the continuity and coherence of the conversation, making the interaction feel more natural and personalized.

F. Large Language Model Processing

The core of our recommendation system is powered by GPT-4o-mini model. The process begins when a user's query is embedded via Sentence Transformer, allowing FAISS to retrieve relevant courses based on embedding similarity. These retrieved courses, along with the user's query and chat history, are formatted into a prompt template. The GPT-4o-mini model then processes this integrated information to generate natural, context-aware course recommendations that align with the user's interests.

G. Streamlit Chatbot Interface

For our chatbot interface, we leveraged Streamlit, an open-source Python library that simplifies the creation of web applications. Streamlit provides a user-friendly framework for rapidly developing interactive and visually appealing interfaces without extensive web development experience. To build our chatbot, we utilized key Streamlit components such as chat_input() for user message input, chat_message() for displaying both user and bot messages, and session_state variable for managing conversation history. By following the streamlit tutorial in building the LLM chatbot application [10], we obtain a chatbot UI that is clean and easy to use. By combining Streamlit's powerful features with our RAG systems we were able to create a responsive, user-friendly chatbot interface that effectively showcases our course recommendation system as shown in Fig. 4.

IV. CHATBOT EVALUDATION AND RESULTS

Fig. 5 showcases the chatbot's response to a user's query about courses in machine learning for finance. The chatbot successfully recommended two relevant courses in this field. For each course, the chatbot provided comprehensive details including:

- The Institution
- A brief course overview
- Skills students will acquire
- Key topics covered
- Course difficulty level
- Language of instruction
- A direct URL to the course page

This structured presentation allows users to quickly compare the courses and access more information with a single click on the provided links. The chatbot's response was evaluated through both automated metrics and human assessment.

A. Automated Metrics

To evaluate our chatbot system's performance, we analyzed the relevance between user queries and generated responses using cosine similarity measurements. We created a diverse dataset of 71 queries covering various subject areas (examples shown in Table I).

Each query was processed through our RAG system to generate recommendation responses. To quantify the semantic alignment between queries and responses, we calculated the cosine similarity between each query-response pair, allowing us

TABLE I. EXAMPLES OF GENERATED QUERIES

Query	Category
I'm looking to enhance my leadership and organizational skills. Seeking courses in Business Management that cover strategic planning, team management, and operational efficiency.	**Business Management**
I want to learn programming from scratch. Looking for beginner courses in Python, Java, or C++.	**Programming**
I need to strengthen my understanding of data structures for better coding practices. Seeking courses that cover arrays, linked lists, trees, and graphs.	**Data Structures**
As an intermediate programmer, I'm looking to dive deep into machine learning for financial applications. Seeking advanced courses in Machine Learning and AI.	**Machine Learning**
I need to strengthen my statistical analysis skills for research. Seeking courses that cover probability, statistical inference, and regression analysis.	**Statistics**
I want to explore the laws of nature and physical phenomena. Seeking courses in Physics that cover mechanics, electromagnetism, and quantum physics.	**Physics**

FIGURE 4. USER INTERFACE OF THE COURSE RECOMMENDATION CHATBOT

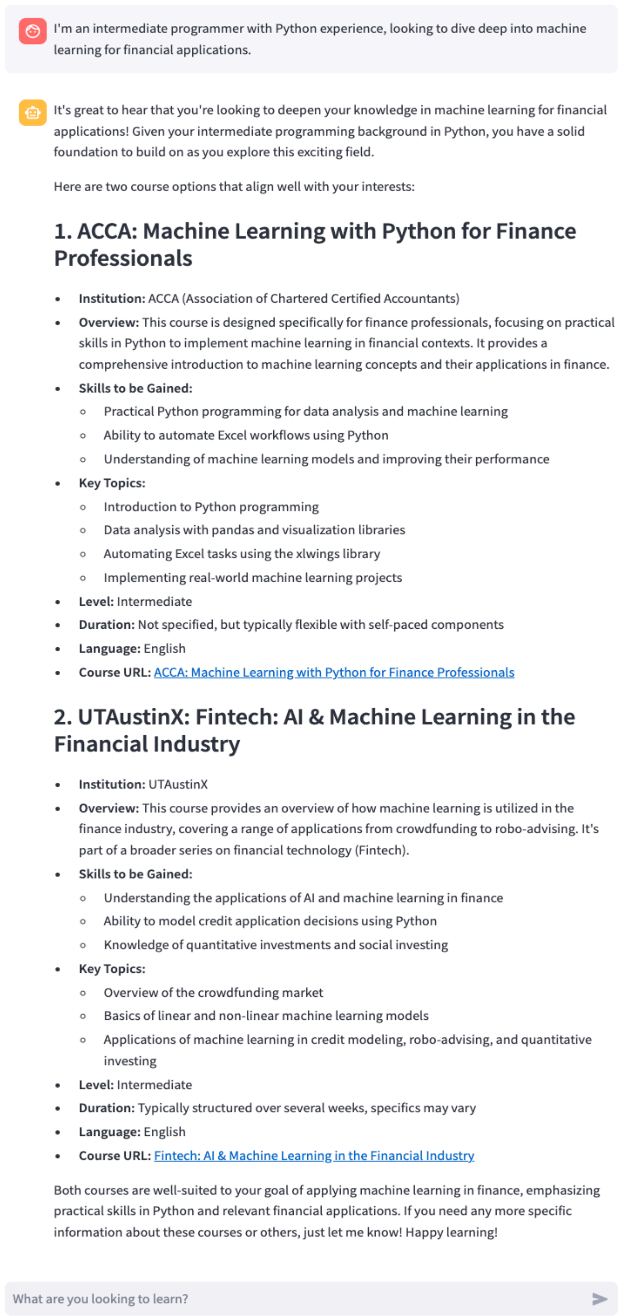


FIGURE 5. SAMPLE CHATBOT RESPONSES AND RECOMMENDATIONS

to assess how well the chatbot's responses matched the users' original questions. The similarity score for each query-response pair was computed using the following (1).

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \quad (1)$$

Where A and B are the numerical vector representations of Query and Response. For context consistency, cosine similarity was also calculated between the retrieved context from the vector store and the chatbot's response. This metric evaluates how well the response aligns with the context used to generate the answer.

TABLE II. AUTOMATED EVALUATION METRICS

Evaluation Metric	**Avg. Score**
Query Relevance Score	**0.77**
Consistency Score	**0.77**

Table II showed an average score of **0.77** for both relevance and consistency, demonstrating the chatbot's strong ability to provide relevant and contextually accurate responses.

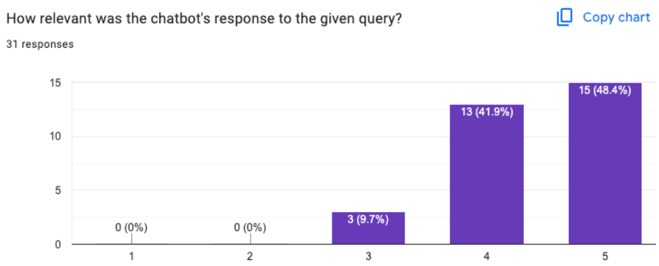
B. Human Assessment

TABLE III. HUMAN EVALUATION RESULTS

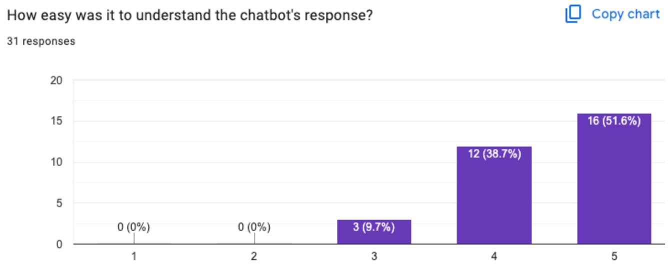
Evaluation Metric	**Avg. Score**
Relevance of the chatbot’s response to the given query	**4.39**
Ease of understanding the chatbot's response	**4.42**
Enhancement of readability through response format	**4.26**
Accuracy of the course recommendations provided	**4.10**
Helpfulness of the explanation given for course recommendations	**4.19**

We conducted a human evaluation of the chatbot's responses through a survey that involved 30 students from our graduate school. The survey focused on five criteria: relevance, comprehensibility, readability, accuracy, and helpfulness. Participants rated each criterion on a scale from 1 to 5, and the results are shown in Fig. 6 and summarized in Table III.

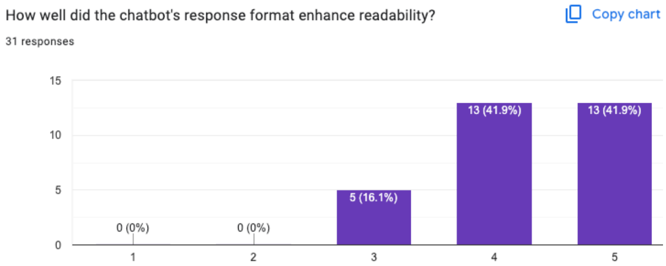
These scores indicate that the chatbot performed well across most criteria. The ease of understanding (comprehensibility) received the highest average score of 4.42, followed by relevance of responses at 4.39. The readability enhancement through response format scored 4.26, while the helpfulness of explanations and accuracy of recommendations scored 4.19 and 4.10 respectively. These results suggest that while the chatbot excels in delivering comprehensible and relevant responses, there might be room for improvement in the accuracy of its course recommendations. Nevertheless, all criteria received scores above 4.0, indicating strong overall performance across all evaluated aspects.



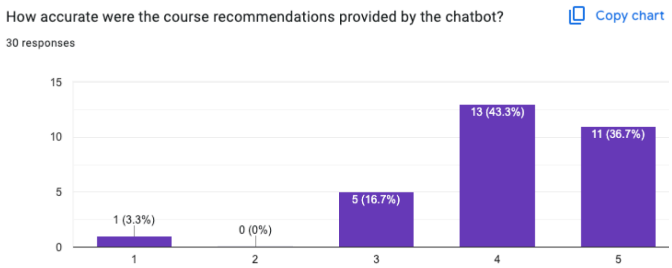
A. Relevance of Response to Query



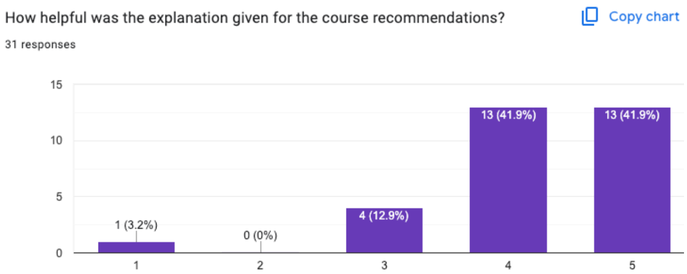
B. Ease of Understanding Response



C. Response Format's Enhancement of Readability



D. Accuracy of Course Recommendations



E. Helpfulness of Course Recommendation Explanation

FIGURE 6. STUDENT SURVEY RESPONSE ANALYSIS

V. Conclusion

This study leveraged the OpenAI GPT-4o-mini model to develop a course recommendation chatbot. Built with LangChain, the system integrates components like FAISS vector store, retrievers, prompt templates, Conversation Buffer Memory, and LLMChain. By employing Retrieval-Augmented Generation (RAG) and embedding vectors sourced from edX and Coursera course data, the chatbot delivers personalized course recommendations tailored to individual user needs.

For evaluation, we conducted both human and automated assessments. The human evaluation metrics revealed that the chatbot consistently performed well across five key criteria: response relevance, comprehensibility, readability, accuracy, and helpfulness of explanations, achieving high satisfaction ratings (4.10-4.42 out of 5.0). For the automated evaluation, we focused on measuring context relevance and consistency. The analysis examined the relationships between queries, retrieved contexts, and chatbot responses, achieving an average similarity score of 0.77.

In future work, we aim to explore alternative vector stores like ChromeBD and Pinecone, enable real-time dataset updates to incorporate newly added courses and develop a mobile app with advanced features. This study underscores the potential of integrating advanced language models with structured data retrieval to create intelligent educational resource recommendation systems.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MIST; RS-2023-00242528, 50%). This work was partly supported by the Innovative Human Resource Development for the Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MIST) (IITP-2024-RS-2022-00156287, 50%).

References

[1] A. Haleem, M. Javaid, M. A. Qadri, and R. Suman, "Understanding the role of digital technologies in education: A review," *Sustainable Operations and Computers*, vol. 3, pp. 275-285, 2022.

[2] S. M. Williamson and V. Prybutok, "The era of artificial intelligence deception: Unraveling the complexities of false realities and emerging threats of misinformation," *Information*, vol. 15, no. 6, p. 299, 2024.

[3] Legal Foundations, "Legal considerations with retrieval augmented generation (RAG)," unpublished.

[4] M. Klokhoj, "Demystifying RAG: Making AI useful in legal practices," unpublished.

[5] Cybersecurity Intelligence, "Revolutionizing legal research & document analysis with RAG technology," unpublished.

[6] A. Guha et al., "LegalBench-RAG: A benchmark for retrieval-augmented generation in the legal domain," unpublished.

[7] "Legal and ethical issues in human language technologies 2024," in *Proc. LREC Workshop on Legal and Ethical Issues in Human Language Technologies*, 2024.

[8] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The Faiss library," unpublished.

[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in press.

[10] *Build a basic LLM chat app - Streamlit Docs*. (n.d.). https://docs.streamlit.io/develop/tutorials/llms/build-conversational-apps