Enhancing Academic Advising with Generative AI: A Retrieval-Augmented Decision Support System

Arhaan Khaku
arhaank@student.ubc.ca
University of British Columbia
Department of Computer Science
Kelowna, British Columbia, Canada

Abdallah Mohamed
abdallah.mohamed@ubc.ca
University of British Columbia
Department of Computer Science
Kelowna, British Columbia, Canada

ABSTRACT

Academic advising encompasses a broad range of student support services, including guidance on transfer credits, graduation requirements, major and minor declarations, and institutional policies. However, traditional advising methods often face challenges related to scalability, consistency, and accessibility. This paper presents a Retrieval-Augmented Generation (RAG)-based chatbot designed to optimize general academic advising by leveraging generative AI. The model aims to enhance data retrieval, improving the precision and relevance of advising responses. We evaluate the chatbot's effectiveness based on response accuracy, user satisfaction, and its ability to address nuanced advising scenarios. Our findings indicate that the RAG-based approach improves efficiency and accessibility in academic advising, providing a scalable AI-driven decision support system for higher education institutions.

1 INTRODUCTION

Academic advising plays a crucial role in guiding students through institutional policies, graduation requirements, transfer credit evaluations, and major/minor declarations. However, traditional advising systems often struggle with scalability, efficiency, and accessibility [6]. Many students face long wait times, unclear or incomplete information, and limited access to advisors, particularly during peak enrollment periods. These inefficiencies can lead to delays in academic progress, misunderstandings about graduation requirements, and difficulties in credit transfer, ultimately impacting student success.

To address these challenges, we propose **EduRAG**, a *RAG-based decision support system* that enhances general academic advising through AI-driven interaction. Unlike conventional rule-based advising tools, which rely on static databases or predefined responses, **EduRAG** dynamically retrieves relevant institutional policies and guidelines, such as academic calendars, while leveraging a large language model (LLM) to generate accurate and context-aware responses. The chatbot supports students across various advising topics, such as *graduation requirements, transfer credit evaluations, major/minor declarations, and institutional policies.*

EduRAG delivers an interactive, 24/7 advising experience, providing real-time, personalized guidance to students across various disciplines, including Computer Science, Data Science, Mathematics, Statistics, Physics and Engineering, tailoring recommendations based on each student's unique academic path.

The key **objectives and contributions** of the proposed work include:

- **Designing a RAG-based chatbot** for general academic advising in various fields, focusing on areas beyond course selection.
- **Assessing the effectiveness** of generative AI in handling advising queries related to transfer credits, study strategies and graduation pathways.
- **Demonstrating the scalability and accessibility benefits** of AI-driven advising systems compared to traditional models.
- **Offering a technical perspective on implementing RAG** for decision support in higher education.

Preliminary findings indicate that RAG-based chatbots can significantly improve response accuracy, consistency, and accessibility in academic advising.

The remainder of this paper is structured as follows: Section 2 reviews related work in AI-driven academic advising. Section 3 details our methodology, including system architecture and implementation. Section 4 presents experimental results and evaluation metrics. Section 5 discusses key findings, challenges, and future directions, and Section 6 concludes the paper.

2 RELATED WORK

2.1 Generative AI in Education and Academic Advising

Recent advancements in Generative AI (GenAI), particularly large language models (LLMs), have transformed academic support by enabling personalized tutoring, automated feedback, and student service chatbots [2]. AI-assisted advising reduces advisor workload and enhances response consistency but often struggles with institution-specific policies.

Many institutions employ traditional machine learning techniques to develop academic chatbots [7], utilizing approaches such as self-supervised fine-tuning and reinforcement learning with human feedback (RLHF). Others opt for menu-driven chatbots, which are cost-effective but less user-friendly.

Most prior research has focused on fine-tuning LLMs on custom datasets, a process that typically requires extensive domain-specific labeled data. However, fine-tuned models remain static unless retrained, limiting adaptability for future retrieval. In contrast, RAG dynamically fetches the latest data from external sources, ensuring up-to-date and policy-compliant responses. Our approach integrates retrieval-based knowledge with GenAI to enhance accuracy and relevance in academic advising.

2.2 RAG in Research and Industry

RAG improves response accuracy by grounding AI outputs in domain-specific documents, reducing hallucination [1]. RAG has been applied in healthcare, legal analysis, and finance, with emerging use in adaptive learning [4]. However, its application to academic advising remains unexplored. Our work bridges this gap by implementing a domain-specific RAG system to retrieve institutional regulations, ensuring accurate and scalable advising.

Our work builds upon these prior studies by applying RAG techniques to general academic advising. By integrating institutional policy retrieval with generative AI, our system aims to provide students with accurate, real-time guidance while ensuring scalability and accessibility.

3 SYSTEM ARCHITECTURE & DESIGN

The core architecture of EduRAG is composed of several key components working together to provide accurate and context-aware responses to user queries. In this section, we discuss the key components of the system architecture, the RAG pipeline, the design choices made, and how these contribute to the system's overall functionality.

3.1 Key Components of the RAG Architecture

Our RAG-based system consists of the following primary components (Figure 4):

- **Data Collection and Preprocessing:** The initial process begins with web-scraping data from sources such as *academic calendars, faculty websites and university resource pages* using the `beautifulsoup4` library. All the collected data then gets divided and formatted into 14 different .txt files, each having content-specific data.
- **Data Loading and Chunking:** All the data then gets loaded with a chunk_size of 1000 characters with an overlap of 250 characters, to make sure we do not miss any crucial data out.
- **Vectorization:** This process converts textual data into vector representations using embeddings generated by Ollama's mxbai-embed-large with 512 token size. The goal is to capture the semantic meaning of each document or query, allowing for efficient similarity-based retrieval. All embeddings then get stored in *pgvector*.
- **Similarity Search:** Once data has been vectorized, similarity search is performed using efficient algorithms, such as cosine similarity, to identify the most relevant documents or pieces of information to answer user queries.
- **Generative Model:** After retrieval, the system uses a generative model (GPT or Ollama) to synthesize a human-like response by conditioning on both the retrieved documents and the user's query.
- **Vector Store (pgvector):** To facilitate fast retrieval, vectorized data is stored in a specialized database (pgvector) that supports similarity search over high-dimensional vectors. This allows the system to quickly retrieve relevant information based on user input.

3.2 The RAG Pipeline and Workflow

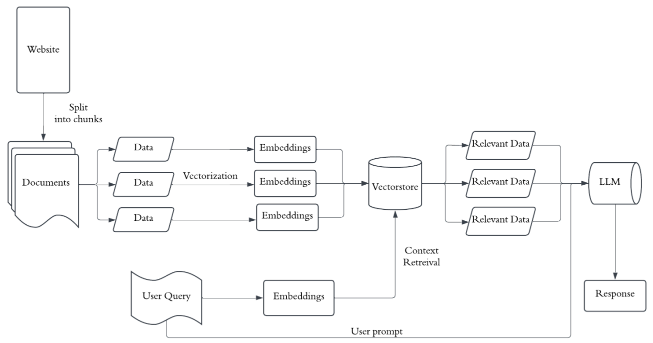


Figure 1: A depiction on how the RAG pipeline operates

The RAG pipeline follows a structured flow from user input to output, ensuring that each stage of the process contributes to a high-quality, accurate response (Figure 4):

1. **User Input:** The process begins when the user interacts with the chatbot, submitting a query related to academic advising (e.g., "What are the graduation requirements for a Computer Science major?").
2. **Vectorization and Query Encoding:** The query is encoded into a vector representation using an embedding model. This vector is then used for similarity search for which it will return the top 3 results.
3. **Similarity Search:** The query vector is compared to the vectors of stored documents in the pgvector database using cosine similarity. The most relevant documents are retrieved based on their proximity in the vector space.
4. **Generation and Response:** Once relevant documents are retrieved, they are passed to a generative model (such as GPT or Ollama) along with the query. The model synthesizes a response based on the retrieved documents and returns it to the user.
5. **Message History and History-Aware Retrieval:** To improve context and the relevance of responses, the system maintains a history of previous interactions. This history is used in *history-aware retrieval*, where past messages are factored into the similarity search, ensuring that the context is preserved across multiple exchanges.

3.3 Design Choices

3.3.1 Prompt Engineering. Effective prompt engineering is crucial for optimizing chatbot performance in RAG-based systems. We carefully designed and refined prompt templates to enhance response accuracy, maintain relevance, and ensure compliance with institutional policies.

- **Contextual Prompting:** Incorporating previous user queries into prompts improved response relevance and coherence.
- **Task-Specific Prompts:** Tailored prompts for advising tasks (e.g., transfer credit evaluations, graduation requirements) ensured domain-specific accuracy.

- **Response Filtering:** The model was instructed to ignore irrelevant user queries, preventing off-topic or misleading answers.
- **Source Attribution:** Every response included a verifiable source to enhance transparency and reliability.
- **Document Retrieval:** The chatbot was designed to provide PDFs for relevant applications when necessary.

Additionally, we developed a separate prompt template focused on history-aware retrieval, enabling the model to retain context across interactions for more informed advising.

Evolution of Prompt Design

In the initial design phase, the chatbot used simple, direct prompts such as:

> *"You are a academic advisor for students. Ignore irrelevant questions asked."*

However, responses were often generic and lacked institutional specificity. Another issue we faced with this was that the model used to hallucinate when it did not know the answer to a user query. To address this, we refined the prompt to incorporate additional context:

> *"You are an academic adivsor for UBC students. Only answer relevant question, if you don't know the answer to something, reply I can only help you with academic advising. Answer the question based on the context below."*

This improved response accuracy. Another feature we wanted to integrate was *source attribution*. Hence, we engineering the chat prompt template to incorporate this feature:

> *"You are an academic adivsor for UBC students. Only answer relevant question, if you don't know the answer to something, reply I can only help you with academic advising. Answer the question based on the context below. In addition to your final answer also return a source as a raw link. Do not say, Source: provided context, a source is always a URL. Here are all the sources:*
> *Sources:* `<list of sources>`*"*

This iterative refinement ensured that responses remained grounded in authoritative data while enhancing user experience.

3.3.2 Similarity Search: Cosine Similarity. The similarity search relies on *cosine similarity*, a commonly used metric to measure the cosine of the angle between two vectors in a vector space. Mathematically, cosine similarity is defined as:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where:

- A and B are two vectors representing a query and a document, respectively.
- $A \cdot B$ is the dot product of the vectors.
- $\|A\|$ and $\|B\|$ are the magnitudes (Euclidean norms) of the vectors.

The cosine similarity measure ranges from -1 to 1, where 1 indicates that the two vectors are identical (i.e., the documents are highly similar), and 0 means there is no similarity. Cosine similarity is efficient in high-dimensional spaces and well-suited for comparing the semantic similarity of queries and documents.

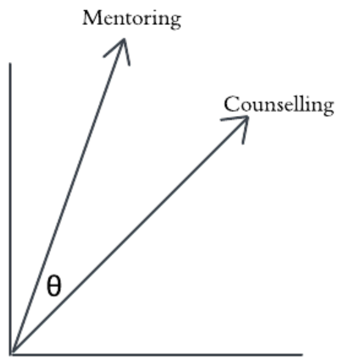


Figure 2: Visualization of cosine similarity between two vectors.

3.3.3 Message History and History-Aware Retrieval. One important design choice in our system is the use of *history-aware retrieval*. This approach helps maintain context across multiple interactions with the chatbot. For example, when a student inquires about graduation requirements and follows up with a question about credit transfer, the system uses both the original question and the follow-up query to retrieve more relevant documents and generate more accurate responses. The history-aware retrieval ensures the system considers the entire interaction, making it more conversational and effective.

3.3.4 Data Collection and Processing. Our data is sourced from multiple channels to ensure accuracy and relevance in academic advising.

- **Web Scraping:** Institutional websites, academic handbooks, and official advising pages are scraped to extract up-to-date information on graduation requirements, major/minor policies, and transfer credits.
- **Stored Text Files:** Scraped data is stored in structured text files, allowing for efficient access and preprocessing before vectorization.
- **Data Cleaning and Formatting:** Raw textual data undergoes cleaning to remove extraneous information (e.g., HTML tags) and is standardized for consistency. This ensures high-quality embeddings, improving similarity search accuracy and reducing retrieval noise.

This structured approach ensures that our data remains current, well-organized, and optimized for retrieval in the advising system.

3.4 Implementation Details

The system integrates *LangChain* for retrieval-augmented generation, *FastAPI* for backend efficiency, *PostgreSQL with pgvector* for vector storage, and *OpenAI and Ollama models* for language processing, ensuring fast and scalable deployment. The user interface is built with *RemixJS and shadcn/ui*.

For efficient document retrieval, we use `pgvector` to store high-dimensional embeddings of institutional guidelines and user queries. PostgreSQL's indexing and querying capabilities enable fast similarity searches, improving response accuracy and efficiency.

3.5 Additional Features

The data used for EduRAG allows it to perform several sophisticated tasks, such as:

- **Mathematical Calculations:** EduRAG can calculate transfer credits by applying mathematical formulas based on the credit value of transferred courses and the institution's conversion rates.
- **Providing PDF Files:** EduRAG can retrieve and provide downloadable advising sheets, such as engineering advising documents or transfer credit forms, directly from the interface.

3.6 Design Rationale

The architectural choices made in this system were driven by the need to balance scalability, accuracy, and accessibility. The use of vectorization and pgvector ensures that we can efficiently handle large datasets and provide fast, relevant responses. The choice to use cosine similarity for similarity search enables precise document retrieval, ensuring that students receive accurate and context-aware guidance. By incorporating features like history-aware retrieval, the system can maintain context over multiple interactions, enhancing the overall user experience. These design decisions ultimately result in a robust, scalable academic advising system that can efficiently support students across various advising tasks.

4 DEMONSTRATION AND EVALUATION

In this section, we present the usability testing and evaluation results of our RAG-based academic advising chatbot. We conducted demonstrations with students from the Computer Science (CMPS) and Engineering programs, followed by a detailed analysis of the feedback, case studies, and performance metrics.

4.1 Initial Usability Testing

We evaluated the chatbot's usability with students from Computer Science (CMPS) and Engineering, who interacted with it for tasks like checking graduation requirements, credit transfers, and program policies.

4.1.1 CMPS Student Feedback. CMPS students appreciated the chatbot's clarity and quick retrieval of graduation requirements, particularly for electives and credit transfers. Some suggested more detailed explanations for complex policies, such as transfer credit applications.

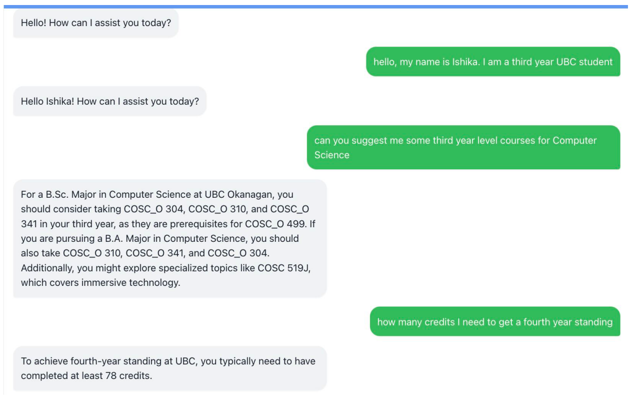
4.1.2 Engineering Student Feedback. Engineering students valued the chatbot's ability to present PDF advising sheets and answer course prerequisite questions. However, like the CMPS group, they desired more contextual explanations for special cases like course substitutions.

Overall, students found the chatbot helpful but highlighted the need for improved explanation depth and a more intuitive interface.

4.2 Case Studies and User Scenarios

To demonstrate the chatbot's effectiveness, we present key use cases:

4.2.1 GoGlobal Program. Students in the GoGlobal study abroad program face challenges with credit transfer. In a test scenario, a student inquired about international course evaluations, and the chatbot provided relevant policies, retrieved transfer guidelines, and offered a PDF advising sheet. This case highlights the chatbot's role in streamlining complex advising tasks.



Hello! How can I assist you today?

hello, my name is Ishika. I am a third year UBC student

Hello Ishika! How can I assist you today?

can you suggest me some third year level courses for Computer Science

For a B.Sc. Major in Computer Science at UBC Okanagan, you should consider taking COSC_O 304, COSC_O 310, and COSC_O 341 in your third year, as they are prerequisites for COSC_O 499. If you are pursuing a B.A. Major in Computer Science, you should also take COSC_O 310, COSC_O 341, and COSC_O 304. Additionally, you might explore specialized topics like COSC 519J, which covers immersive technology.

how many credits I need to get a fourth year standing

To achieve fourth-year standing at UBC, you typically need to have completed at least 78 credits.

Figure 3: GoGlobal Student User Case Scenario

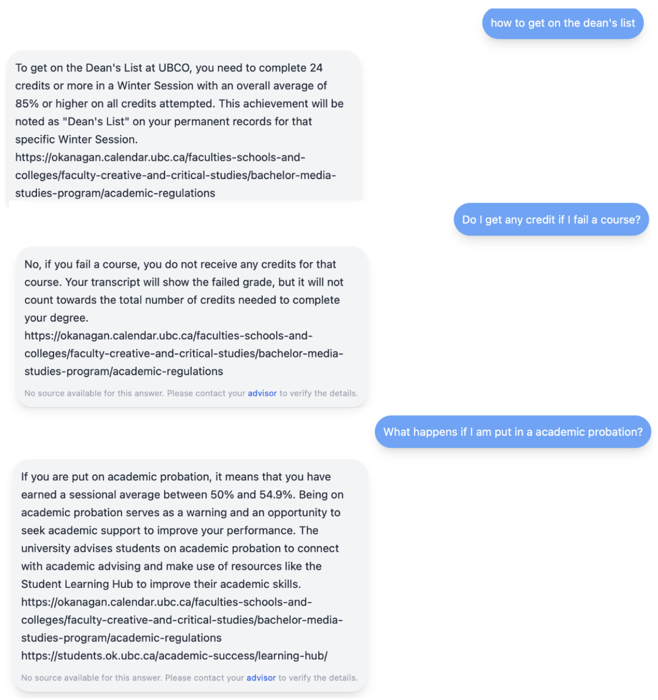
4.2.2 Course and Credit Transfers. A first year student majoring in Data Science had queries related to certain keywords in the academic calendar. It asked EduRAG for advice on those words

4.3 Backend Testing with PyTest

To ensure the reliability and correctness of our backend, we implemented a comprehensive testing suite to validate API responses, handling user queries, and verifying logical equivalency using GPT-4. Specifically, we focused on:

- **Irrelevant Question Handling:** The system is tested to reject non-academic queries by returning a predefined response.
- **Credit and Transfer Credit Inquiries:** We verify that queries related to credit requirements and transfer credit evaluations yield accurate responses.
- **Graduation and Degree Planning:** The chatbot is tested for inquiries regarding degree progression, graduation requirements, and document shipping details.
- **Major/Minor Declarations:** The chatbot's ability to provide correct information on major and minor self-declaration is assessed.
- **Conversation History Awareness:** To test context retention, we simulate a multi-turn conversation where the chatbot should recall previous exchanges.

To validate chatbot responses beyond simple string comparison, we employ GPT-4 for logical equivalence assessment. A dedicated



how to get on the dean's list

To get on the Dean's List at UBCO, you need to complete 24 credits or more in a Winter Session with an overall average of 85% or higher on all credits attempted. This achievement will be noted as "Dean's List" on your permanent records for that specific Winter Session. https://okanagan.calendar.ubc.ca/faculties-schools-and-colleges/faculty-creative-and-critical-studies/bachelor-media-studies-program/academic-regulations

Do I get any credit if I fail a course?

No, if you fail a course, you do not receive any credits for that course. Your transcript will show the failed grade, but it will not count towards the total number of credits needed to complete your degree. https://okanagan.calendar.ubc.ca/faculties-schools-and-colleges/faculty-creative-and-critical-studies/bachelor-media-studies-program/academic-regulations

No source available for this answer. Please contact your advisor to verify the details.

What happens if I am put in a academic probation?

If you are put on academic probation, it means that you have earned a sessional average between 50% and 54.9%. Being on academic probation serves as a warning and an opportunity to seek academic support to improve your performance. The university advises students on academic probation to connect with academic advising and make use of resources like the Student Learning Hub to improve their academic skills. https://okanagan.calendar.ubc.ca/faculties-schools-and-colleges/faculty-creative-and-critical-studies/bachelor-media-studies-program/academic-regulations https://students.ok.ubc.ca/academic-success/learning-hub/

No source available for this answer. Please contact your advisor to verify the details.

Figure 4: First Year Student User Case Scenario

function queries GPT-4 to determine if two responses convey the same meaning, enhancing test robustness.

Algorithm 1 Logical Equivalence Assessment

```
1: Input: Two response strings, $str_1$ and $str_2$
2: Output: Boolean value indicating logical equivalence
3: Send system message and user query to GPT-4
4: Parse response to extract decision string
5: if response contains "true" then
6:     return True
7: else if response contains "false" then
8:     return False
9: else
10:    return Error
```

4.4 Evaluation Metrics

To assess the performance of our RAG system, we evaluate retrieval accuracy, response quality, and system efficiency. The results are summarized in Table 1.

Retrieval Performance

- **Recall@5**: Measures the proportion of queries for which a relevant document appears in the top 5 retrieved results. Higher values indicate better retrieval effectiveness.
- **Mean Reciprocal Rank (MRR)**: Computes the average reciprocal rank of the first relevant document across queries, emphasizing how early relevant documents appear in ranked results.

Table 1: Evaluation metrics for the RAG system.

Metric	Result
Recall@5	87.2%
MRR	0.79
NDCG@5	0.84
BLEU Score	41.6
ROUGE-L	56.3
Response Time (ms)	920
Token Efficiency (avg tokens/query)	285

- **Normalized Discounted Cumulative Gain (NDCG@5)**: Evaluates ranking quality by assigning higher importance to correctly ranked relevant documents, penalizing incorrect ordering in the top 5 results.

Generation Quality

- **BLEU Score**: A precision-based metric that compares generated responses to reference answers using n-gram overlap, commonly used in machine translation.
- **ROUGE-L**: A recall-oriented metric that evaluates the longest common subsequence between generated and reference responses, measuring fluency and coherence.

System Efficiency

- **Response Time (ms)**: The average time taken to generate a response after retrieving relevant documents. Lower values indicate faster response generation.
- **Token Efficiency**: The average number of tokens processed per query, influencing API cost and latency. Lower values improve cost-effectiveness without degrading response quality.

4.5 Discussion of Results

EduRAG demonstrates strong retrieval and response accuracy, with Recall@5 of 87.2% and MRR of 0.79, ensuring relevant document retrieval. However, recall can be improved for complex queries. The high NDCG@5 (0.84) confirms effective ranking, while BLEU (41.6) and ROUGE-L (56.3) scores indicate coherent, high-quality responses.

With an average response time of 920ms and token efficiency of 285 tokens per query, the system balances speed and cost-effectiveness. Student feedback highlights its success in diverse advising scenarios, though improvements in explanation depth and conversational flow are needed. Future work will focus on enhancing retrieval, refining responses, and improving interactivity.

5 DISCUSSION

The academic advising chatbot developed in this study demonstrates several strengths, while also presenting a few limitations. In this section, we reflect on the overall development process, the challenges faced, and the strengths and limitations of the system.

5.1 Strengths

The chatbot's main strength lies in its ability to quickly retrieve and present accurate academic advising information from a vast dataset. Using the RAG approach, the chatbot is able to combine generative capabilities with a retrieval mechanism to provide responses that are both contextually relevant and grounded in the provided data. Some of the key strengths of the system include:

- **High accuracy and precision**: The system consistently provides correct and relevant information, as demonstrated by the high accuracy and precision scores.
- **Quick response times**: The chatbot processes queries rapidly, ensuring a smooth user experience with minimal delay.
- **Ability to handle complex advising scenarios**: The chatbot excels in providing tailored advice for unique academic situations, such as transfer credit evaluation and study abroad program details.
- **Scalability**: The RAG-based system is scalable and can handle a wide variety of queries related to graduation requirements, transfer credits, and major/minor information.

5.2 Limitations

Despite its strengths, the chatbot has several limitations:

- **Limited contextual understanding**: While effective for straightforward inquiries, the chatbot may struggle with complex, nuanced questions that require deep reasoning or domain expertise.
- **Reliance on data quality**: The accuracy of responses depends on the quality and completeness of the dataset. Outdated or missing information can lead to incorrect or misleading answers.
- **AI bias and hallucinations**: Large language models can inherit biases present in their training data and may generate hallucinated responses—plausible but incorrect information—potentially leading to misinformation.
- **Transparency and source attribution**: The system does not always explicitly cite sources, making it difficult for users to verify the credibility of responses. Enhancing transparency through source attribution could improve trust.
- **Limited interactivity**: The chatbot could benefit from improved conversational capabilities, such as dynamic follow-up questions and adaptive responses based on user input.

5.3 Mitigating Biases in LLM-Based Advising

To address biases and hallucinations, several mitigation strategies are employed:

- **Curated knowledge sources**: The chatbot relies on vetted academic data to minimize exposure to biased or misleading information.
- **Human-in-the-loop oversight**: While automated, the system allows human advisors to review and refine responses, ensuring accuracy in critical scenarios.
- **Bias detection mechanisms**: Future iterations could integrate bias-detection techniques to flag and adjust responses that exhibit skewed perspectives.
- **Source transparency**: Enhancing citation mechanisms within the chatbot will allow users to verify the origins of retrieved information.

These measures help improve reliability while maintaining AI-assisted academic advising as a supportive tool rather than a standalone replacement for human expertise.

5.4 Challenges Encountered

The development process was not without its challenges. Some of the key issues encountered during the development phase included:

- **Data formatting and cleaning**: The process of transforming raw academic data into a structured format suitable for training the chatbot was time-consuming. Data cleaning, such as ensuring consistency and removing redundancies, was essential to improve the accuracy of the system.
- **Fine-tuning the model for specific tasks**: The task-specific nature of the queries required continuous fine-tuning of the model to ensure high accuracy in real-world scenarios.
- **System integration and testing**: Integrating various components, such as the vector store (pgvector), similarity search methods, and chatbot interface, proved to be complex, requiring multiple iterations and debugging.

6 CONCLUSION AND FUTURE WORK

This paper presented EduRAG, a chatbot-based academic advising system leveraging RAG architecture to provide accurate, context-aware guidance on graduation requirements, transfer credits, and major/minor selection. By integrating history-aware retrieval and vector-based storage (pgvector), the system enhances advising efficiency while reducing student wait times.

Key areas for improvement include expanding data coverage, refining conversational flow, and broadening advising services. Future enhancements will focus on reinforcement learning for response generation, multi-modal interactions, and adaptive learning for personalized support.

While EduRAG streamlines routine advising, it does not replace human advisors. Given LLM limitations, it serves as a complementary tool, improving accessibility and efficiency while ensuring students retain access to personalized guidance.

REFERENCES

[1] Nguyen, L., & Quan, T. (2025). *URAG: Implementing a Unified Hybrid RAG for Precise Answers in University Admission Chatbots – A Case Study at HCMUT.* arXiv. Retrieved from https://doi.org/10.48550/arXiv.2501.16276

[2] Peyton, K., Unnikrishnan, S., & Mulligan, B. (2025). *A review of university chatbots for student support: FAQs and beyond.* Discov Educ, 4, 21. https://doi.org/10.1007/s44217-025-00397-7

[3] Osorio Cárdenas, D., & Guatibonza Briceño, P. A. (2025). *AI-Assisted Learning: Intelligent Tutoring System for the Introduction to Programming Course.* Technical Report, Fundación Universitaria de Ciencias de la Salud. Retrieved from https://hdl.handle.net/1992/75502

[4] Galstyan, L., Martirosyan, H., Vardanyan, E., & Vahanyan, K. (2024). *SmartAdvisor University Chatbot Spring 2024*

[5] Ali, U. S. (2024). *Ask your Transcript: LLM Driven Insights for Academic Advising.* In *Proceedings of the 2024 2nd International Conference on Computing and Data Analytics (ICCDA)* (pp. 1-4). https://doi.org/10.1109/ICCDA64887.2024.10867349

[6] Doe, J., et al. (2020). *From Traditional to Intelligent Academic Advising: A Systematic Literature Review of e-Academic Advising.* International Journal of Advanced Computer Science and Applications, 11(4). https://doi.org/10.14569/IJACSA.2020.0110467

[7] Aguila, A., Tran Ngoc, N., Nguyen, N. A. D., Huynh, K.-T., Mai, A., Le, T. D., & Tuyen, N. T. V. (2024). *Large Language Model in Higher Education: Leveraging Llama2 for Effective Academic Advising.* Journal Name, X(Y), ZZ-ZZ. https://example.com