EFFICIENT CHATBOT FOR UNIVERSITY ADMISSION CONSULTATION USING LARGE LANGUAGE MODELS

CHATBOT HIỆU QUẢ CHO TƯ VẤN TUYỂN SINH ĐẠI HỌC SỬ DỤNG MÔ HÌNH NGÔN NGỮ LỚN

Truc Thi Kim Nguyen[1], Van Nam Hoang[2], Pham Ngoc Tinh[2], Nguyen Nang Hung Van[1]*

[1]The University of Danang - University of Science and Technology, Vietnam
[2]Dong A University, VietNam

*Corresponding author: nguyenvan@dut.udn.vn

(Received: June 17, 2025; Revised: September 10, 2025; Accepted: September 17, 2025)

DOI: 10.31130/ud-jst.2025.23(9A).329E

Abstract - This paper presents an Artificial Intelligence–driven chatbot for university admission consultation using Large Language Models (LLMs). The system integrates semantic retrieval with Retrieval-Augmented Generation (RAG) and employs a hybrid strategy that combines vector similarity and keyword matching to provide accurate and context-aware answers. The chatbot was trained on admission FAQs, official documents, and consultation records from Dong A University, ensuring relevance to real user needs. Implementation leverages efficient prompt construction and memory management to support interactive and personalized responses. Experimental results show improved retrieval precision and practical benefits in reducing staff workload and offering consistent support to prospective students. Current limitations include the use of a single-university dataset and a technical evaluation focused on retrieval metrics. Future work will expand to multi-institution data, user studies, and multilingual or voice-enabled interaction to enhance generalizability and real-world impact.

Key words - Chatbot; Large Language Model; AI in Education; Natural Language Processing; Admission Counseling; Question-Answering.

Tóm tắt - Bài báo tập trung vào việc xây dựng chatbot tư vấn tuyển sinh thông minh dựa trên Mô hình Ngôn ngữ Lớn, nhằm nâng cao khả năng cung cấp thông tin và ngữ cảnh cho thí sinh. Hệ thống được xây dựng theo hướng tạo sinh tăng cường kết hợp giữa truy xuất ngữ nghĩa và tìm kiếm theo độ tương đồng véc-tơ. Nghiên cứu đã khai thác dữ liệu tư vấn tuyển sinh thực tế từ Trường Đại học Đông Á, bao gồm bộ câu hỏi thường gặp, thông tin tuyển sinh và lịch sử tư vấn. Nhờ vậy, chatbot thể hiện khả năng ứng dụng thực tiễn, giúp cải thiện độ chính xác trong truy xuất thông tin và giảm đáng kể khối lượng công việc của người tư vấn. Hạn chế chính của nghiên cứu là dữ liệu chỉ giới hạn ở một trường đại học và tiêu chí đánh giá chỉ dừng ở các chỉ số kỹ thuật. Tương lai, nghiên cứu sẽ mở rộng sang nhiều cơ sở giáo dục và bổ sung đánh giá từ người dùng, để tăng tính tổng quát cũng như giá trị ứng dụng của hệ thống.

Từ khóa – Mô hình ngôn ngữ lớn; AI trong giáo dục; Xử lý ngôn ngữ tự nhiên; Tư vấn tuyển sinh; hỏi - đáp.

1. Introduction

The emergence of Large Language Models (LLMs) has significantly enhanced Artificial Intelligence (AI) chatbots' conversational capabilities, enabling them to engage in more human-like interactions. Advancements in data availability and computational techniques have further improved the performance of LLM-based chatbots, leading to their widespread adoption across various industries. These systems are now capable of comprehending and generating human language with unprecedented contextual relevance and accuracy, while managing extensive streams of information Bharathi Mohan et al. [1]. Consequently, LLM-powered chatbots have become essential tools in sectors such as education Abedi et al. [2], economy Steve et al. [3], research Macdonald et al. [4], and healthcare Sallam [5].

Despite their immense potential, the increasing use of LLM-based chatbots presents several challenges that demand thorough investigation. The rapid pace of development in this field has led to a substantial body of research, which can be overwhelming for scholars and practitioners.

Recent advancements in Natural Language Processing (NLP) and deep learning have significantly improved chatbot efficiency. Transformer-based models such as BERT and GPT facilitate superior language understanding and response generation. Retrieval-Augmented Generation (RAG) further refines chatbot performance by incorporating external knowledge sources. Previous studies have highlighted the effectiveness of fine-tuned language models in domain-specific applications, particularly in educational chatbots.

Vietnamese, as a tonal language with distinct syntactic and semantic features, requires specialized approaches for effective NLP. PhoBERT, a transformer-based model tailored for Vietnamese, has demonstrated exceptional performance in tasks such as part-of-speech tagging, dependency parsing, named-entity recognition, and natural language inference Nguyen and Nguyen [6], Duc et al. [7].

The increasing complexity of university admissions underscores the need for advanced technologies to enhance both efficiency and user experience Yigci [8]. Traditional methods often struggle to deliver timely and accurate information to the growing number of prospective students

and their families. AI, particularly LLMs, offers a promising solution to this challenge. By leveraging AI, institutions can develop chatbots capable of providing precise, context-aware, and immediate responses, thereby streamlining the admissions process Element451 [9].

Building upon the flowchart and foundational framework introduced in our previous work [10], this study advances the development of LLM-based admission chatbots with several key extensions. While [10] focused primarily on LLM selection and large-scale dataset behavior using data from The University of Danang - University of Science and Technology (UD-DUT), our current work applies the approach to a localized dataset from Dong A University, emphasizing embedding diversity, deployment-oriented engineering, and domain-specific adaptation.

Specifically, we integrate new-generation LLMs - Grok, Gemini, and LLaMA 3 - and conduct a comprehensive comparison of embedding models (MiniLM, gte-multilingual, bge-m3, nomic) to evaluate their effectiveness in Vietnamese-language admission counseling. In addition, tailored prompt engineering and memory optimization techniques are introduced to improve retrieval precision, scalability, and user experience. These enhancements not only extend the original concept but also provide a practical framework for deploying AI-driven chatbots in real-world admission processes.

2. Mathematical Problem Formulation

The chatbot system integrates semantic retrieval and natural language generation to optimize university admission consultations. Below, we formalize the problem.

2.1. Problem Setup

Let:

$\mathcal{Q}$ be the set of all possible user queries, where each $q \in \mathcal{Q}$ is a natural language string (e.g., "what are the admission deadlines?").

$\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ be the corpus of N admission-related documents (e.g., FAQs, policies), where each d_i is a text string.

f_{embed}: $T \rightarrow \mathbb{R}^k$ be the embedding function mapping text strings in $T = \mathcal{D} \cup \mathcal{Q}$ to k-dimensional vectors, where k (e.g., 384 or 768) is the dimensionality of the chosen embedding model (e.g., nomic-embed-text-v1.5)

$\mathcal{V} = \{v1, v2, \ldots, v_N\}$ be the vector representations of documents, where $v_i = f\text{embed}(d_i)$.

$v_q = f_{\text{embed}}(q)$ be the vector representation of query $q \in \mathcal{Q}$.

2.2. Objective

The objective is twofold: (1) retrieve the top-K most relevant documents from $\mathcal{D}$ for a given query q, and (2) generate an accurate response based on those documents.

- Retrieval: Define a similarity function $S : \mathbb{R}^k \times \mathbb{R}^k \rightarrow [-1, 1]$ (e.g., cosine similarity) to measure relevance between v_q and v_i. The retrieved subset $\mathcal{R}_q \subseteq \mathcal{D}$ contains the K documents with the highest individual similarity scores:

$$\mathcal{R}_q = \{d_i \in \mathcal{D} \mid v_i \text{ is the top } K \text{ by } S(v_q, v_i)\} \quad (1)$$

Equation (1) retrieves the top-K documents that are most relevant to the user query based on similarity scores. This ensures that the chatbot focuses on the most informative sources.

- Prompt Construction: Define $P : \mathcal{Q} \times 2^{\mathcal{D}} \rightarrow S(v_q, v_i)$ S as a function mapping a query and retrieved documents to a prompt string $s \in S$, where S is the set of valid input strings for the LLM. For:

$$P(q, \mathcal{R}_q) = \text{"Query:"} + q + \text{"Context"} + (\mathcal{R}_q) \quad (2)$$

Equation (2) formalizes the process of building a prompt by combining the user query with the retrieved documents. This step prepares the context for the LLM.

- Response Generation: The LLM $F_{LLM} : S \rightarrow \mathcal{A}$ generates a response $\tilde{a}_q = F_{LLM}\left(P(q, \mathcal{R}_q)\right)$, where $\mathcal{A}$ is the set of possible answers.

- Optimization: Maximize the expected accuracy of the response relative to a ground truth $a_q^* \in \mathcal{A}$:

$$\max_{\mathcal{R}_q, P} \mathbb{E}_{q \sim Q}[\text{Accuracy}(\tilde{a}_q, a_q^*)] \quad (3)$$

where $\text{Accuracy}(\tilde{a}_q, a_q^*)$ is a metric (e.g., BLEU score, exact match, or MRR@k), and the expectation is over a distribution of queries.

Equation (3) defines the optimization goal, which is to maximize the accuracy of generated responses when compared to ground truth answers.

2.3. Constraints

The system must satisfy the following constraints:

- Token Limit: The prompt's token count must satisfy:

$$token\_count\left(P(q, \mathcal{R}_q)\right) \leq T_{max} \quad (4)$$

where T_{max} (e.g., 8192) is the LLM's maximum context window, and token_count is defined by the LLM's tokenizer.

Equation (4) specifies the token limit constraint to ensure the prompt length does not exceed the maximum input size of the LLM.

- Computational Constraints:

+ Embedding time: $\text{time}(f_{\text{embed}}(x)) \leq T$ seconds per input $x \in T$

+ GPU memory usage: $\text{memory}(f_{\text{embed}}, F_{LLM}) \leq \mu$ GB.

2.4. Hybrid Retrieval Scoring

To enhance retrieval, we use a hybrid scoring approach:

- Define normalized similarity functions:

$S_{vector}(v_q, v_i) \in [-1, 1]$: Cosine similarity between embeddings.

$S_{keyword}(q, d_i) \in [0, 1]$: Normalized keyword overlap (e.g., Jaccard similarity on tokenized text).

- Hybrid score:

$$S_{hybrid}(q, d_i) = \alpha \cdot S_{vector}(v_q, v_i) + (1 - \alpha) \cdot S_{keyword}(q, d_i) \quad (5)$$

where $\alpha \in [0, 1]$ is optimized via cross-validation on a labeled dataset to maximize retrieval precision (e.g., P@3).

Equation (5) introduces a hybrid score that combines cosine similarity (semantic meaning) and keyword overlap. This balances contextual relevance with exact keyword matching.

- Retrieval is updated as:

$$\mathcal{R}_q = \{d_i \in \mathcal{D} \mid v_i \text{ is among the top K by } S_{hybrid}(q, d_i)\} \quad (6)$$

Equation (6) updates the retrieval process using the hybrid score, leading to improved precision and recall in admission queries.

Figure 1 illustrates the retrieval and prompt construction process described in Equations (1) – (6). The user query is first embedded and compared against stored document vectors. A hybrid scoring function combines semantic similarity with keyword overlap to identify the top-K relevant documents. These documents are then merged with the query to form the final prompt, which is passed to the LLM for response generation.

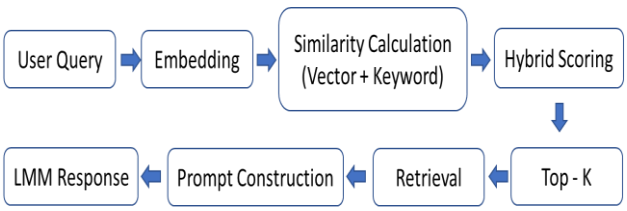

Figure 1. *Illustration of retrieval and prompt construction process*

3. Proposed Chatbot System

The proposed system aims to develop an AI-powered chatbot leveraging Large Language Models (LLMs) to optimize the university admissions consultation process. The architecture is designed to provide users with prompt, accurate, and user-friendly information regarding admissions. The system's workflow is divided into two primary stages, shown in Figure 2.

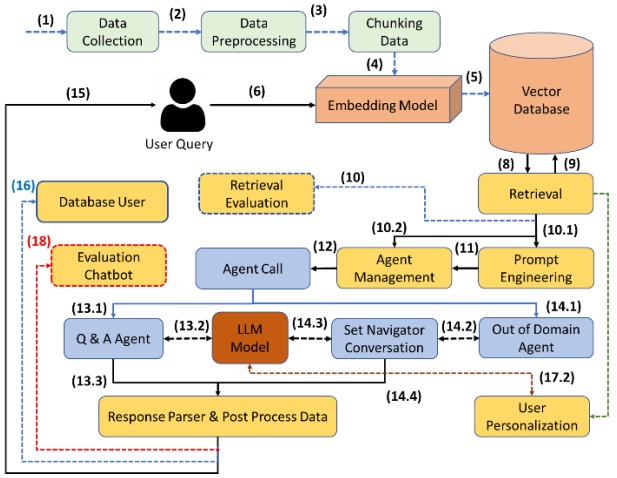

Figure 2. *System workflow of the proposed chatbot [10]*

Stage 1: System Preparation and Data Processing

1. Data Collection and Scenario Creation: Gather admissions-related data from Dong A University's official website and generate potential question scenarios based on real-life admissions situations, such as those from the university's 2024 Facebook admissions page.

2. Data Preprocessing: Clean and standardize the collected data, labeling it according to specific categories or topics. This step includes expanding common abbreviations to enhance the chatbot's response accuracy.

3. Data Segmentation: Divide lengthy data samples into shorter segments to align with the processing

4. Data Embedding: Transform the preprocessed data into vector representations using embedding models, ensuring semantically similar data points are positioned closely within the vector space

5. Vector Storage and Retrieval Evaluation:

- *Vector Storage*: Store the embedded vectors in a vector database to optimize subsequent information retrieval processes.

- *Retrieval Evaluation:* Assess the retrieval performance using various algorithms to ensure efficient and accurate query responses.

Stage 2: Chatbot Deployment and User Interaction

1. Vector Storage and Retrieval Evaluation: When a user inputs a query, the system embeds the input into a vector format compatible with the stored data representations

2. Information Retrieval: Utilize a Retrieval-Augmented Generation (RAG) system to compare the user's query vector against the database, identifying the most relevant information.

3. Prompt Generation and Agent Management:

- *Prompt Generation:* Create optimized prompts to guide the LLM in understanding the user's context and intent.

- *Agent Management:* Implement an agent management system to route user requests to the appropriate processing modules, including the LLM.

4. Agent Execution: Direct the processed query to the suitable agent for response generation.

5. Response Generation:

- *For In-Scope Queries:* The LLM generates responses based on retrieved information and conversation history, which are then displayed to the user.

- *For Out-of-Scope Queries:* The system formulates appropriate responses, guiding users to contact admissions staff for further assistance.

User Feedback and Data Logging: Collect user interactions and feedback to refine the chatbot's performance and maintain context continuity in conversations.

User Information Extraction and Personalization: Automatically extract user preferences and interests using the LLM to tailor responses and suggest relevant topics, enhancing user engagement and satisfaction.

This structured approach integrates modern AI techniques with a systematic workflow, ensuring the chatbot operates efficiently, meets user needs effectively, and remains scalable for future expansions or integrations.

4. Implementation Details

The chatbot is built using Python and TensorFlow, supplemented by Hugging Face Transformers and FastAPI for backend services. FAISS optimizes nearest-neighbor searches in the embedding layer. Apache Airflow manages data ingestion and model updates.

4.1. System Architecture

The chatbot is integrated into a web-based platform utilizing the Chainlit framework. The core components consist of:

- *User Interface:* An interactive chat-based interface guiding users through structured interactions.

- *Backend Processing*: Query interpretation, response formulation, and knowledge retrieval.

- *Database Management:* Efficient storage and retrieval of vector embeddings via Milvus.

- *API Integration:* Real-time updates and information synchronization.

4.2. Embedding Model Selection

Selecting an effective embedding model is essential for optimizing retrieval quality, latency, and resource efficiency in chatbot systems. Comparing multiple candidates enables a holistic understanding of their trade-offs and ensures that the chosen solution aligns with the specific requirements of real-time, domain-specific applications.

In our study, we evaluated four embedding models for semantic retrieval: all-MiniLM-L6-v2, gte-multilingual-base, bge-m3, and nomic-embed-text-v1.5. Their performance was assessed using Top-K accuracy, embedding time, memory usage, and other efficiency metrics (Table 1).

Table 1. *Comparison and Performance Evaluation of Proposed Embedding Models*

Model Name	Sentence-transformer/ all-MiniLM-L6-v2	Alibaba-NLP gte-multilingual-base	BAAI/ bge-m3	nomic-ai/ nomic-embed-text-v1.5
Average Embedding Time per Question (s)	0.005	0.010	0.021	0.012
Average Question Length	14.681	166.484	9.734	14.681
Average Embedding Time per Sentence (s)	0.013	0.056	0.024	0.014
Average Sentence Length	389.149	1,744.195	373.787	398.149
GPU Memory Usage (GB)	0.109	1.466	2.725	0.656
Number of Parameters	22,713,216	305,368,320	567,754,76	136,731,64
Context Length	256	8,192	8,192	8,192
Embedding Dimension	384	768	1,024	1,024
Top 10 Accuracy	0.468	0.872	0.851	0.574
Top 5 Accuracy	0.351	0.840	0.798	0.436
Top 3 Accuracy	0.234	0.755	0.787	0.351
Top 1 Accuracy	0.096	0.500	0.521	0.213

Among them, *nomic-embed-text-v1.5* was selected as the primary embedding model. It provides high retrieval precision, balanced embedding dimensions, efficient processing time, and moderate resource requirements. These characteristics make it well-suited for real-time chatbot applications with limited hardware resources.

In Table 1, we compare the performance of various free LLMs in a RAG context. The Top-K Accuracy metric indicates the percentage of correct answers appearing within the Top-K results (e.g., top 1, 3, 5, or 10). Higher values signify better model performance in ranking relevant results for a given query.

After a comprehensive analysis, we selected the nomic-ai/nomic-embed-text-v1.5 model as our primary embedding model for the following reasons:

High Top-K Accuracy: This model demonstrates superior retrieval precision, ensuring that relevant information is effectively surfaced during query processing.

Optimal Embedding Dimensions: With a balanced vector size, it captures essential semantic features without imposing excessive computational demands.

Efficient Embedding Time: The model offers rapid embedding capabilities, which are crucial for applications requiring swift data processing, such as real-time chatbots and question-answering systems.

Extended Context Handling: It supports a context length of up to approximately 8,000 tokens, making it adept at managing lengthy and complex textual inputs.

Resource Efficiency: The model maintains a favorable balance between performance and GPU memory consumption, facilitating deployment on systems with limited hardware resources.

Low Latency and Batch Processing: Its quick encoding capabilities enhance the processing of large datasets, optimizing scenarios that demand prompt responses and efficient batch operations.

Cost-Effective Trial: The availability of a free trial with a generous token allowance makes it an accessible choice for small to medium-scale projects, allowing for thorough testing and validation.

These attributes collectively make nomic-ai/nomic-embed-text-v1.5 a compelling choice for embedding tasks within our RAG framework, aligning with our objectives of accuracy, efficiency, and scalability.

4.3. Performing Retrieval Methods

After embedding the data and storing it in a vector database, we employed various retrieval models to assess the effectiveness of the embeddings, data quality, and retrieval accuracy prior to integration with LLM. An effective retrieval method ensures high-quality answers. In this project, we utilized three retrieval methods: vector retrieval, keyword retrieval, and hybrid retrieval.

Vector Retrieval: This method uses vector embeddings to represent information objects (such as documents, questions, or answers). It's a common technique in modern information retrieval systems, especially when combined with deep learning models. Vector retrieval captures the semantic meaning of queries and documents, allowing for more nuanced matching beyond exact keyword overlaps.

Keyword Retrieval: A more traditional approach where user queries are matched with keywords in the database. This method often employs string search techniques, such as exact keyword matching or variations within the text. While straightforward, it may miss semantically relevant information that doesn't share the same keywords.

Hybrid Retrieval: This approach combines both vector and keyword retrieval methods to leverage the strengths of each, thereby improving the quality and efficiency of search results. By integrating semantic understanding from vector retrieval with the precision of keyword matching, hybrid retrieval aims to provide more comprehensive and accurate results.

To evaluate the effectiveness of our retrieval methods, we employed three key metrics: Precision at 3 (P@3), Recall at 3 (Recall@3), and Mean Reciprocal Rank at 3 (MRR@3). P@3 measures the proportion of relevant items among the top three retrieved results, indicating the system's ability to present pertinent information promptly. Recall@3 assesses the system's capability to retrieve all relevant documents within the top three results, reflecting its comprehensiveness. MRR@3 evaluates how quickly the system returns the first correct answer within the top three results, focusing on the rank position of the initial relevant result.

Table 2. *Performance Comparison of Retrieval Methods*

Model	P@3	Recall@3	MRR@3
Keyword	0.197	0.590	0.477
Hybrid	0.213	0.640	0.440

In our performance comparison in Table 2, the hybrid retrieval method outperformed both vector and keyword retrieval approaches across all evaluated metrics. This suggests that combining semantic embeddings with traditional keyword matching provides a more comprehensive retrieval strategy, effectively capturing relevant information and ranking it appropriately. Consequently, the hybrid approach was selected for integration into our system to optimize retrieval performance.

4.4. Implementation of the Chatbot Agent System

In the university admissions chatbot system, the agent is pivotal in facilitating user interactions, processing information, querying data, and delivering responses. It communicates with the LLM by preparing prompts-comprising questions, messages, or requests-and sending them to the LLM to receive appropriate replies. This section outlines the agent's components and the implementation strategy for each.

The prompt is crafted based on the design illustrated in Figure 2 of the proposed methodology. It guides the LLM in responding to user inquiries, with the quality of the chatbot's replies heavily dependent on the constructed prompt. Once search results from the database are retrieved and ranked according to the user's query, they are utilized to create a prompt for the model. This prompt combines the user's question with information from the search results, providing a clear and accurate context for the LLM. The process involves: reminding the LLM of its identity and role (e.g., informing it that it is UDAChat with the task of providing users with essential information), supplying a contextual summary, presenting the user's question, identifying relevant retrieval and rerank results, and offering necessary instructions and notes to ensure effective responses.

4.5. Implementation of Chatbot Memory Management

In developing our university admissions chatbot, we integrated MemGPT, an advanced memory management system designed to enhance long-term context handling and response accuracy. MemGPT Packer et al. [11] enables the chatbot to summarize and store crucial information from previous interactions, allowing it to maintain context within token limits and recall important details from past conversations. This capability is particularly beneficial for providing personalized and effective assistance in complex scenarios like admissions counseling.

The primary advantage of MemGPT lies in its combination of accuracy, contextual continuity, and user adaptability. By managing different memory tiers, MemGPT allows the chatbot to effectively provide extended context within the limited context window of large language models. This approach ensures that the chatbot delivers precise and contextually relevant responses, enhancing both performance and user satisfaction.

4.6. Large Language Models Utilized

The rapid development of Large Language Models (LLMs) has significantly improved their accuracy, contextual understanding, and multilingual capabilities. Modern LLMs extend beyond text generation to support semantic search, reasoning, and dialogue systems, making them essential for chatbot applications. This study employs three representative models - LLaMA 3, Gemini 1.5, and Grok - each offering distinct advantages.

LLaMA 3 (Meta) is a high-performance model with up to 128k token context, trained on over 15 trillion tokens. It delivers strong comprehension, reduced hallucination, and seamless integration with chatbot frameworks.

Gemini 1.5 (Google) emphasizes contextual reasoning and multilingual support, achieving high efficiency in dialogue generation and translation tasks.

Grok (xAI) is optimized for concise, real-time content generation and lightweight conversational tasks, complementing the capabilities of larger models.

The selection of multiple LLMs for chatbot development aims to maximize the free resources these models offer, such as computational resources and limited query allowances. Additionally, distributing user requests across these models can reduce processing time, thereby enhancing response efficiency and decreasing the chatbot's latency.

5. User Interface Development with Chainlit Framework

In this project, we employed the Chainlit framework to build an intuitive and responsive user interface for the chatbot. As an open-source Python toolkit for rapid conversational AI development Contributors [12], Chainlit

enabled efficient interface creation while allowing us to focus on system design, deployment, and optimization - ultimately ensuring a seamless user experience for admission inquiries at Dong A University.

Figures 3 showcase the deployed chatbot interface, highlighting its interactive elements and user-centric design.

Figure 3. *Chatbot Interface Example 1*

These illustrations demonstrate the practical application of Chainlit in creating an effective chatbot interface tailored for university admissions inquiries.

In this project, we utilized the Chainlit framework to develop the user interface (UI) for our admissions chatbot. Chainlit is an open-source Python package that enables rapid construction of ChatGPT-like applications, offering a range of UI components, customization options, and seamless integration capabilities. This approach allowed us to focus more on designing, implementing, and optimizing the chatbot system efficiently. Below are illustrative images showcasing the deployment of the Dong A University admissions chatbot using LLMs.

6. Conclusion

This research highlights the potential of AI-powered chatbots in university admission consultations. The proposed chatbot improves accessibility, streamlines information delivery, and enhances user engagement. Future iterations will prioritize multilingual support and voice-based functionalities to further optimize the user experience.

To further improve our system, we plan to implement several key enhancements. Firstly, we aim to expand support for multiple languages, including Vietnamese and English, to cater to a broader user base. Secondly, integrating speech recognition and synthesis will facilitate voice-based interactions, enhancing accessibility for users who prefer or require auditory engagement. Additionally, we intend to refine response personalization by analyzing query histories, thereby providing more tailored assistance. Lastly, implementing advanced graph-based data structures will enhance our knowledge retrieval capabilities, leading to more accurate and contextually relevant responses.

Future work will incorporate data from multiple institutions, unstructured sources such as emails and social media, and conduct real-world evaluations with students.

Acknowledgments: This work was supported by The University of Danang - University of Science and Technology, code number of Project: T2024-02-37.

REFERENCES

[1] G. B. Mohan, “An analysis of large language models: their impact and potential applications”, *Knowledge and Information Systems*, vol. 66, no. 9, pp. 1–24, 2024.

[2] M. Abedi, I. Alshybani, M. R. B. Shahadat, and M. S. Murillo, *“Beyond Traditional Teaching: The Potential of Large Language Models and Chatbots in Graduate Engineering Education*”, Qeios, pp. 1–45, 2023.

[3] M. Steve, S. Brightwood, and O. Godwin, “Implementing AI Chatbots for Real-Time Supply Chain Monitoring and Risk Management”. *www.researchgate.net,* 2024, [Online]. Available: https://www.researchgate.net/publication/383395365 [Accessed: June 25, 2025]

[4] C. Macdonald, D. Adeloye, A. Sheikh, and I. Rudan, “Can ChatGPT draft a research article? An example of population-level vaccine effectiveness analysis”, *Journal of Global Health*, vol. 13, 2023.

[5] M. Sallam, “ChatGPT utility in healthcare education, research, and practice: systematic review on the promising perspectives and valid concerns”, *Healthcare*, vol. 11, pp. 887, 2023.

[6] N. Q. Dat and N. T. Anh, “PhoBERT: Pre-trained language models for Vietnamese”, *23rd International Conference Artificial Intelligence and Soft Computing,* no. 00744, 2020.

[7] N. Q. Duc, L. H. Son, N. D. Nhan, N. D. N. Minh, L. T. Huong, and D. V. Sang, “Towards Comprehensive Vietnamese Retrieval-Augmented Generation and Large Language Models”, *arXiv preprint*, no. 01616, pp. 1-6, 2024.

[8] D. Yigci, M. Eryilmaz, A. K. Yetisen, S. Tasoglu, and A. Ozcan, “Large Language Model-Based Chatbots in Higher Education”, *Advanced Intelligent Systems published by Wiley-VCH GmbH,* 2024 pp. 1-16, https://doi.org/10.1002/aisy.202400429.

[9] B. Hurter, “The Role of AI Chatbots for Higher Education Success in 2024”, *Element451.com*, Nov 24, 2024. [Online]. Available: https://element451.com/blog/chatbots-in-higher-ed-what-you-should-know. [Accessed: June 25, 2025].

[10] N. N. H. Van, P. H. Do, V. N. Hoang, T. T. K. Nguyen, and M. T. Pham, "AI-Powered University Admission Counseling: A Use Case of Large Language Models in Student Guidance", *in IEEE Transactions on Learning Technologies,* vol. 18, pp. 856-868, 2025, doi: 10.1109/TLT.2025.3604096.

[11] C. Packer *et al.,* “MemGPT: Towards LLMs as Operating Systems”, *arXiv preprint*, no. 08560, pp. 1-13, 2023.

[12] Contributors, “Chainlit Documentation: Overview”, *Chainlit.io*. 2025. [Online]. Available: https://docs.chainlit.io/get-started/overview. [Accessed: June 25, 2025].