Educational and Humanities
ISSN Print: 3104-4352
ISSN Online: 3104-4360
CD ISSN: 3104-4379
https://doi.org/10.63313/EH.9123

Research on a Mongolian Intelligent Question-Answering System for University Information Services

He Su
Assistant lecturer, School of Artificial Intelligence, Hohhot Minzu College, Hohhot 010051, China
Email: huosuhe@gmail.com

How to cite this paper: Su, H. (2026). Research on a Mongolian intelligent question-answering system for university information services. Educational and Humanities, 5(1), 77–88. ISSN Print: 3104-4352, ISSN Online: 3104-4360.
https://doi.org/10.63313/EH.9123
Published: 2026-10-30

Copyright © 2026 by author(s) and Erytis Publishing Limited.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
CC BY 4.0

Abstract

With the emergence of the digital era, higher-education institutions increasingly require efficient information-service platforms capable of meeting the diverse needs of students and teachers. This paper presents the design, implementation, and testing of a Python-based intelligent question-answering system: a web chatbot for university information services with particular attention to Mongolian-language interaction. The system uses the Flask web framework and integrates MySQL to store and manage key campus information, including teaching buildings, faculty offices, and vacant classrooms. Elasticsearch is introduced to implement intent recognition and fuzzy information retrieval, allowing users to obtain answers even when their queries do not exactly match stored questions. The system further integrates the ChatGPT API to provide academic guidance and basic psychological support, and the Chimege API to deliver Mongolian speech-to-text and text-to-speech conversion, which is particularly valuable for international students who are learning Mongolian and may find written communication difficult. Because a commercial cloud deployment with managed Elasticsearch was estimated at approximately USD 120 per month, the chatbot was deployed through a cloud-access approach in which the developer's computer served as the host and a secure tunnel exposed the local service to the public Internet. Functional testing covered campus FAQ retrieval, ChatGPT interaction, and Chimege voice conversion; the system met approximately 80% of the expected requirements. The results demonstrate a practical and extensible solution for intelligent information services in a university environment, with planned extensions including literature-download translation, payment-related functions, and community-based improvement.

Keywords

Chatbot; Flask; Elasticsearch; ChatGPT API; Chimege API; University Information Services

1. Introduction

In today's rapidly developing digital environment, higher-education institutions face growing challenges in information management and service delivery. Students and teachers frequently need quick and accurate access to campus information such as building locations, faculty offices, and classroom availability, while demand for academic and psychological counseling services is also rising [1, 2]. Traditional information-service methods cannot always meet the complexity and immediacy of these needs, making an intelligent chat assistant for integrated information services particularly valuable [3].

Intelligent chatbots address information overload by allowing users to ask questions in natural language and receive direct answers instead of browsing multiple search-result pages. However, many existing solutions are not adapted to higher education: they lack understanding of students' learning and living contexts, cannot answer university-specific questions, and are often not user-friendly.

The objective of this research is therefore to design and implement a Python-based chat assistant for the higher-education environment. The study seeks to improve information-retrieval efficiency through rapid and accurate campus-related information; to strengthen academic and psychological support by integrating the ChatGPT API; to optimize user interaction through convenient text and voice channels; and to implement smart campus services using Elasticsearch and speech recognition.

The innovation of the work lies in two directions. First, the system provides question answering over a Mongolian-language FAQ knowledge base, combining ordinary retrieval with the ChatGPT and Chimege APIs to support academic and basic psychological guidance. Second, it addresses a specific and underserved user group: international students who are new to Mongolia and face language barriers in daily university life. Because some of these students can read Mongolian but not yet write it comfortably, Mongolian speech-to-text and text-to-speech interaction is a central feature rather than an auxiliary convenience.

2. Related Work and Key Technologies

2.1. Research on Intelligent Chatbots in Education

Research into conversational robots developed alongside artificial intelligence. In the 1950s, Alan M. Turing proposed the question "Can machines think?" as part of the Turing Test for assessing the development of artificial intelligence. Early systems were limited by computing power and knowledge bases, so designers often restricted conversations to specialized domains to reduce complexity [4]. With the rapid growth of Internet companies and the popularity of mobile terminals and applications such as Twitter and Weibo, major technology companies invested heavily in conversational technologies. Relatively mature cross-platform products such as Apple Siri, Microsoft Cortana, Facebook Messenger, and Google Assistant enabled users to obtain information and services more conveniently, using

large-scale data, semantic analysis, machine learning, and deep neural networks to improve conversational understanding [5]. Conversational systems have since been applied in technical question answering, negotiation, e-commerce, teaching, entertainment, and other human-computer interaction scenarios.

As demand for intelligent chatbots has increased, the use of artificial intelligence for natural-language processing has become an active research direction, and chatbot research has grown in importance for both artificial intelligence and human-computer interaction [6, 7, 8,]. In the education domain specifically, researchers have examined the application of chatbots in teaching and learning [9], the design and implementation of chatbots based on natural-language processing, and development trends of chatbots in the AI era [2]. Comparative studies of human–human and human–chatbot conversations have clarified what users expect from realistic dialogue with artificial intelligence [4], while work on changing user needs and motivations has shown that the perceived usefulness and social character of a chatbot strongly shape sustained use [10]. Effective strategies for implementing chatbot interfaces in higher education have also been proposed [3], and systematic reviews of conversational agents in adjacent fields such as healthcare indicate both the maturity of the underlying techniques and the limits of current educational specialization [11]. In recent years, chatbot technology has developed considerably: systems can not only provide basic information-retrieval services but also understand and answer more complex questions through natural-language processing and machine learning. In particular, the ChatGPT API provides powerful conversational-processing capabilities through advanced language models and can support effective interaction in different scenarios. Nevertheless, few existing solutions are tailored to university-specific information needs, and even fewer support a low-resource language such as Mongolian.

2.2. Challenges for Educational Question-Answering Systems

The literature identifies three persistent challenges for educational question-answering systems and educational robots more broadly. The first is educational insufficiency. Educationally relevant functions are at the core of intelligent educational robots, yet current products often focus on games or simple repetitive teaching functions, are relatively homogeneous, and remain limited in educational specialization; strategic learning design, learning-resource recommendation, and accurate assessment of cognitive ability are still insufficient [11]. The second is poor interactivity. Interaction is an important part of the learning process because it can correct or reinforce learners' cognitive processes in a timely manner and provide psychological and emotional support, but many current systems lack advanced natural-language processing and optimized learning-strategy design, so they cannot provide timely, accurate, and appropriate feedback. The third is a lack of understanding. Effective educational agents should

recognize learners' emotions, learning environment, and other conditions during the learning process, yet current systems rarely perceive learners and their learning processes in multiple dimensions and lack real-time understanding based on multidimensional data, resulting in insufficient intelligence. These challenges motivated the design choices reported in this paper, particularly the integration of a large language model for open-ended dialogue and the emphasis on accessible voice interaction.

2.3. Key Technologies

The technology stack was selected to balance development speed, flexibility, and retrieval quality. For the web framework, Django and Flask were compared. Django is a full-featured framework providing an administrator interface, database interface, object-relational mapping, authentication, messaging, and session management; it follows a "convention over configuration" approach, but its broad feature set creates a steeper learning curve. Flask, by contrast, is a microframework that provides core tools for web development without forcing a specific database abstraction layer or form-validation system [12, 13]. Flask was selected because its simplicity and flexibility suit small- and medium-sized web applications, rapid prototyping, and projects with specific architectural requirements, and because of its broad community support and well-documented libraries. For data storage and management, MySQL was adopted; its reliability, scalability, and speed make it suitable for structured data such as user queries, chat records, and university-related information. For retrieval, Elasticsearch, a search engine based on the Lucene library, was introduced to handle complex and large datasets, provide high-speed search, and, critically, support fuzzy searching for ambiguous queries [14]. The ChatGPT API supplies advanced natural-language processing capabilities built on large language models, enabling the system to understand user questions and generate detailed, contextually appropriate responses [15]. Finally, the Chimege API provides Mongolian speech-to-text and text-to-speech conversion, improving accessibility for users who prefer or require voice interaction, including users with visual limitations and international students learning Mongolian [16]. The integration handles audio data, converts speech input into text, and converts chatbot text responses into spoken audio, with attention to voice clarity, speech rate, and language intonation.

3. System Design

3.1. Conceptual Design and Functional Specifications

The conceptual-design stage established a framework defining the chatbot's core functions and user interactions. The primary target audience comprises university students, teachers, and administrative staff. Understanding these user groups is essential for adapting the chatbot to their specific needs, and surveys and interviews

were used to identify key needs such as campus information, academic assistance, and psychological support, so that the design responded to the actual requirements of the university community. On this basis, the chatbot's primary goals were defined as providing accurate campus information, facilitating academic inquiries, and offering basic psychological support; secondary goals include increasing user engagement through interactive functions such as personalized responses and a user-friendly interface. The essential functional specifications include an interactive question-and-answer system, database queries, and an integrated academic calendar, while advanced capabilities include language translation, speech recognition, and integration with university systems for personalized notifications. In terms of design considerations, the user interface was required to be intuitive and accessible across demographic groups, and the user experience was designed as a continuous flow with reduced response time and improved information accuracy. To validate the concept and the selected technology stack—Python, Flask, MySQL, Elasticsearch, and cloud-related services—a basic chatbot prototype was developed, and feedback collected from the target audience was used to iterate on the concept. The conceptual-design stage concluded with detailed documentation describing the intended functions, design principles, and technology choices, providing a clear path for subsequent design and development.

3.2. System Architecture

The system architecture (see Figure 1) organizes the chatbot into cooperating layers: a front-end conversational interface implemented in standard HTML, CSS, and JavaScript; a Flask back-end service that handles HTTP requests, routing, and web-page serving; a data layer combining MySQL storage with an Elasticsearch index; and an integration layer connecting the ChatGPT API for open-domain dialogue and psychological support and the Chimege API for Mongolian voice conversion. Figure 1 summarizes how user requests flow from the browser through the Flask server to the FAQ database, the search engine, or the external APIs, and how responses return to the user as text or speech.

The deployment structure (see Figure 2) reflects the project's cost constraints: rather than a commercial cloud server, the developer's computer hosts the application, with port 5001 exposed to the public Internet through a secure tunnel so that external testers can access the service. The use-case design defines the interactions available to students, teachers, and staff—querying campus information, invoking AI-assisted academic guidance through the "A.I" function, requesting basic psychological support through the "P.C" function, and using voice input and output—while the class design structures the system into components for the user interface, the Flask application, database access, search, and the external API clients, with clearly separated responsibilities for request handling, data persistence, retrieval, and speech processing. This modular organization keeps each

component replaceable: the FAQ knowledge base can grow independently of the dialogue logic, and the external APIs can be substituted as services evolve. The architecture thereby supports the scalability and adaptability discussed in later sections and allows the chatbot to develop together with changing user needs and technological advances.

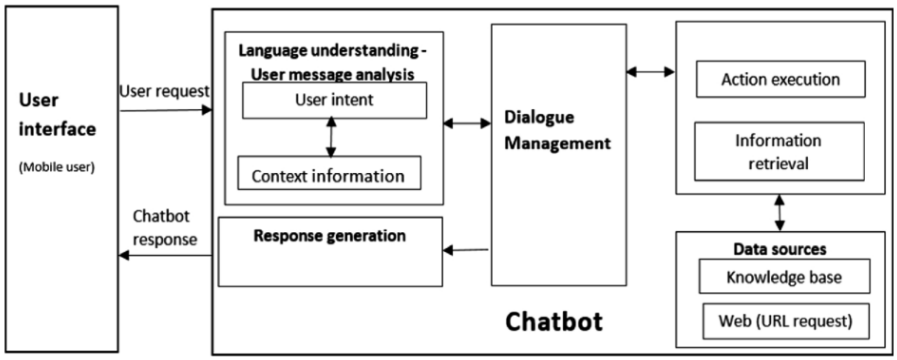


Figure 1. System architecture

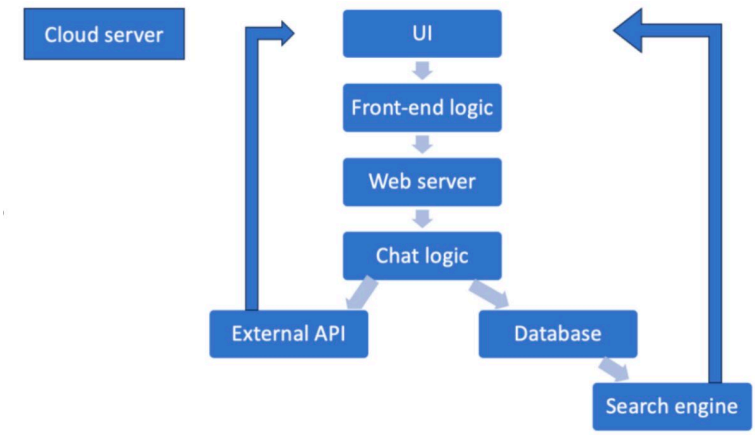


Figure 2. System deployment structure

4. System Implementation

4.1. Front-End Development

The chatbot user interface was designed around user experience, functionality, and aesthetics. Before the design began, the needs and preferences of the target users were analyzed, including how they might use the chatbot, what they expected from it, and what technical limitations might exist. The resulting interface is intended to be easy to understand and navigate for users with different levels of technical experience. Its conversational flow allows users to enter information and receive responses easily, following familiar interaction patterns such as Messenger and Facebook; visual cues such as message bubbles distinguish user and chatbot messages, while excessive decorative elements are avoided to keep information

clear. Because the Chimege API is used for speech-to-text and text-to-speech functions, the interface also provides dedicated controls for accessing the user's microphone, recording audio, sending the recorded audio file to the server, and returning the processed result to the user, together with text-to-speech and speech-to-text panels that make the voice functions discoverable. Finally, responsive-design principles were applied so that the interface works well across different devices and screen sizes, particularly the mobile devices on which students are most likely to consult the assistant, ensuring that campus information remains accessible wherever users are.

4.2. Back-End Development and API Integration

The Flask web application imports essential Python modules and libraries, including time, requests, Flask, pymysql, Elasticsearch, and OpenAI, and forms the basis of the web service, handling HTTP requests, routing, and the serving of web pages. On startup, Flask connects to MySQL and retrieves FAQ information; questions submitted from the front end are matched against questions stored in the database, and when a matching question is found, the corresponding answer is returned to the front end for display in the conversation. The application is served through a WSGI server so that the Flask process can handle requests reliably during the testing stage.

An "A.I" button is integrated into the chatbot interface to call the ChatGPT API. When the "A.I" button is selected, the system bypasses the MySQL matching process and uses the ChatGPT API directly, making the chatbot more flexible and allowing teachers and staff to use the function without relying on the FAQ database. The interface also provides a "P.C" button for psychological-counseling support. At this stage, the function calls the ChatGPT API and can provide students with basic psychological assistance, but it is explicitly experimental: the responses are generated by an AI model rather than a certified psychologist and should not be regarded as professional or authoritative psychological counseling. Future development should involve cooperation with professional psychological-counseling organizations to improve this function.

Because international students at the university are still learning Mongolian, and some can read Mongolian without yet being comfortable writing in Mongolian script, the Chimege API was integrated to provide voice-conversion functions that assist learners of Mongolian. The implementation encountered a practical problem: audio must be transferred between the client and the server for conversion. The chatbot therefore records audio in the browser, stores the recorded file locally, sends the audio file to the Chimege API server, and returns the processed result—recognized text or synthesized speech—to the user. The process requires attention to voice clarity, speech rate, and language intonation, and real-time voice recognition and synthesis remain challenging because speech styles and terminology can vary

considerably in university environments. Despite these difficulties, the pipeline makes the chatbot usable for students who cannot yet type comfortably in Mongolian script.

4.3. Database and Fuzzy Search

MySQL is used to create the school-level FAQ database table, which consists of an ID, a question, and an answer: the ID is used for numbering, the question field stores questions asked by students, and the answer field stores the corresponding responses. However, if an answer could be returned only when the user's question matched a stored question exactly, the chatbot would remain a rule-based robot, requiring users to phrase their questions in ways that correspond precisely to the stored entries. To overcome this limitation, the project adopts approximate matching and semantic search so that the system can identify the user's intended question and provide the appropriate answer. In simple terms, a user may only need to enter one or two relevant words, and the chatbot can identify the question being asked. Elasticsearch is used to implement this fuzzy matching: the content of the MySQL FAQ table is transferred into an Elasticsearch index through a dedicated import routine, after which Flask connects to Elasticsearch and performs fuzzy information retrieval over the Mongolian FAQ knowledge base (see Figure 3). When a query arrives, the search layer scores the stored questions against the user's input and returns the answer attached to the best-matching entry, so that partial or approximate wording still leads to a relevant response. This mechanism constitutes the system's intent-recognition layer and is central to making the chatbot usable for non-native speakers whose phrasing may deviate from the stored questions.

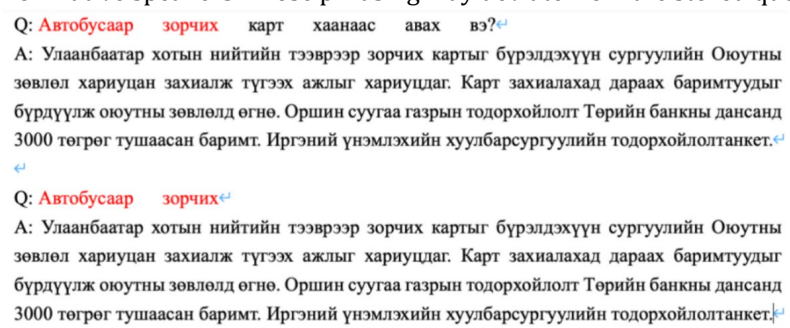


Figure 3. Fuzzy-match retrieval over the Mongolian FAQ knowledge base

4.4. Cloud Deployment

The original plan was to deploy the chatbot on an Alibaba Cloud server. However, the Alibaba Elasticsearch cloud-server option was estimated to cost approximately USD 120 per month, and no budget support was available at this stage. The project therefore used local-network access technology so that the developer's computer could serve as the server while exposing the local service to the Internet. Port 5001 was exposed to the public network to allow a small number of people to participate

in the testing stage and provide data and feedback. Cpolar, a secure cloud service for exposing local servers on a private network to the public network, was used to establish a secure tunnel so that public-network users could access the locally hosted service (see Figure 2); the Cpolar dashboard was used to manage and monitor the tunnel during the testing period. This arrangement allowed a small-scale public trial to proceed without incurring cloud hosting fees. Deployment on a cloud server can be considered after obtaining support from the university or government, and the architecture's reliance on standardized APIs and data-exchange formats is intended to make such a migration straightforward, since only the hosting environment—and not the application logic—would need to change.

5. System Testing and Results

Before end-to-end functional testing, the operating environment was verified step by step: the Elasticsearch service was started and confirmed to be running successfully; the MySQL database was opened and its FAQ table inspected; the Cpolar tunnel was activated and monitored through its dashboard; and the Flask application was launched by running the server script. These checks established that each infrastructure component operated correctly before user-facing functions were evaluated.

Functional testing then covered the system's principal interaction modes. First, FAQ retrieval was exercised by posing campus-related questions through the chat interface; the Elasticsearch-backed fuzzy matching returned appropriate answers from the Mongolian FAQ knowledge base even for loosely phrased queries. Second, ChatGPT-powered interaction was tested through the "A.I" function, which answered open-ended academic and general questions in multiple languages without recourse to the FAQ database. The psychological-counseling interaction was likewise exercised through the "P.C" function, confirming that the system could deliver basic supportive responses while remaining, as noted above, an experimental feature. Third, the Chimege voice-conversion pipeline was tested for both directions of speech interaction—text-to-speech output of chatbot responses and speech-to-text recognition of spoken Mongolian input—confirming that recorded audio could be transferred to the server, processed, and returned to the user (see Figure 4).

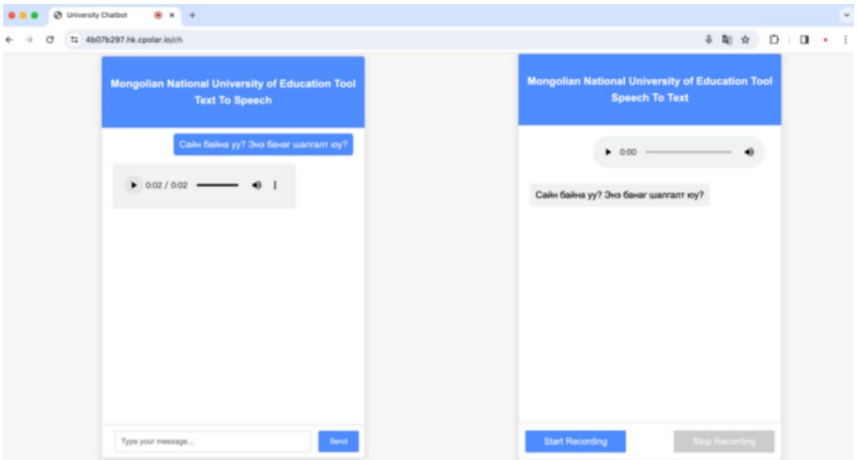


Figure 4. Mongolian speech interaction (text-to-speech and speech-to-text) via the Chimege API

Because the service was exposed to the public network through port 5001, a small number of external users could participate in the testing stage and provide data and feedback under realistic access conditions rather than only on a local machine. Their interaction confirmed that the conversational flow, the microphone controls, and the API integrations behaved consistently across devices.

At the time of the study, the chatbot was assessed as meeting approximately 80% of the expected requirements, and functional testing together with user feedback indicated good system performance and improved user satisfaction, demonstrating a practical solution for intelligent information services in a university environment. The remaining gap reflects both the limited coverage of the FAQ knowledge base and functions that were specified during conceptual design but not yet implemented, such as personalized notifications. Future work is planned to collect more answers to questions about the university and to add more functions, including technology-related resources, literature-download translation, payment-related functions, and other appropriate features. The project also proposes creating a community in which more people can participate in improving the chatbot, extending both its knowledge base and its functionality over time.

6. Challenges and Solutions

Seven main challenges arose during development, each with a corresponding solution. (1) Natural-language understanding: complex and varied phrasing of identical needs is difficult to interpret; integrating the ChatGPT API improves understanding accuracy, while Elasticsearch-based fuzzy matching absorbs variation in FAQ phrasing. (2) System integration: combining Flask, MySQL, and Elasticsearch is nontrivial; standardized APIs and data-exchange formats ensured smooth integration among components. (3) Scalability: maintaining performance and stability as users grow will require load balancing and a microservice architecture. (4) User experience: an interface that is easy to use for all users was

addressed through voice-interaction functions and a responsive UI. (5) User engagement: long-term engagement calls for personalized features such as user-behavior analysis and gamification elements. (6) Data security and privacy: protecting user information requires end-to-end encryption, strict access controls, and compliance with data-protection regulations, which is especially pertinent given the psychological-support function. (7) Cloud deployment: with no budget support and the Alibaba Elasticsearch server estimated at approximately USD 120 per month, the developer's computer was used as the host and port 5001 was exposed to the Internet through Cpolar, with full cloud deployment deferred until institutional or government support is obtained. Taken together, these solutions prioritized keeping the system available, secure, and useful while preserving a clear upgrade path for future development.

7. Conclusion

This project demonstrates substantial progress in applying AI and web technologies to education. Its major achievement is a chatbot that is functional, user-friendly, and accessible to a diverse university audience: attention to UI/UX design, accessibility, and ease of use has produced a tool that improves the everyday experience of students, teachers, and staff, while the speech-to-text and text-to-speech functions make it a more versatile resource for Mongolian-language learners. The scalability and adaptability of the system architecture allow the chatbot to evolve with changing user needs and technological advances. The project thus provides an example of how technology can be leveraged to address practical challenges in an educational environment and a reference for future efforts in this field.

Looking ahead, three trends are likely to shape educational question-answering systems. The first is educational specialization: future systems will require learning resources developed according to the characteristics of educational robots, higher-level methodological modeling, knowledge maps for professional courses, and learning strategies grounded in cognitive science. The second is interactive intelligence: educational robots can implement dialogue systems oriented toward teaching objectives, guide learners through specific learning tasks during interaction, and satisfy learners' emotional need for communication and expression. The third is contextualized understanding: future educational robots may incorporate richer embedded sensors and wearable devices, while technologies such as context awareness and affective computing could transform multidimensional data into information about learners' internal emotions, concentration, and psychological needs. These directions define a concrete agenda for the continued development of the chatbot reported here, from an information-service assistant toward a more comprehensive educational companion.

Acknowledgment

This work was supported in part by a grant from
Research Innovation Team on International Communication Capacity of Northern Frontier Culture (Grant No. RZ2500000453)

References

[1] Wang F. (2019) Design and implementation of a chatbot based on natural language processing[J]. Ruanjian Gongcheng (Software Engineering), 22(2): 112-118. (in Chinese)

[2] Liu T T. (2020) Development trends of chatbots in the AI era[J]. Jisuanji yu Wangluo (Computer & Network), 46(6): 62-65. (in Chinese)

[3] Knox V J, Wang Y. (2018) Effective strategies for implementing a chatbot interface in higher education[J]. International Journal of Educational Technology, 5(1): 15-21.

[4] Hill J, Ford W R, Farreras I G. (2015) Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations[J]. Computers in Human Behavior, 49: 245-250. DOI: 10.1016/j.chb.2015.02.026.

[5] McTear M, Callejas Z, Griol D. (2016) The Conversational Interface: Talking to Smart Devices[M]. Cham: Springer.

[6] Zhao X D. (2017) Design and application of intelligent chatbots[J]. Jisuanji Xitong Yingyong (Computer Systems & Applications), 26(9): 27-31. (in Chinese)

[7] Chen M, Guo F. (2019) A review of chatbot research based on deep learning[J]. Jisuanji Kexue (Computer Science), 46(6A): 11-15. (in Chinese)

[8] Yang J. (2018) Exploration of chatbot applications in customer service[J]. Shichang Zhoukan (Market Weekly), (34): 167-168. (in Chinese)

[9] Zhang W, Li Q. (2018) Research on the application of chatbots in education[J]. Xiandai Yuancheng Jiaoyu Yanjiu (Modern Distance Education Research), 30(4): 78-85. (in Chinese)

[10] Brandtzaeg P B, Følstad A. (2018) Chatbots: Changing user needs and motivations[J]. Interactions, 25(5): 38-43. DOI: 10.1145/3236669.

[11] Laranjo L, Dunn A G, Tong H L, et al. (2018) Conversational agents in healthcare: A systematic review[J]. Journal of the American Medical Informatics Association, 25(9): 1248-1258. DOI: 10.1093/jamia/ocy072.

[12] Grinberg M. (2018) Flask Web Development: Developing Web Applications with Python[M]. Sebastopol: O'Reilly Media.

[13] Pallets Projects. Flask documentation (Version 3.0.x)[EB/OL]. [2026-09-23]. https://flask.palletsprojects.com/

[14] Elastic. Elasticsearch[EB/OL]. [2026-09-23]. https://www.elastic.co/

[15] Radford A, Wu J, Child R, et al. (2019) Language models are unsupervised multitask learners[EB/OL]. [2026-09-23]. https://openai.com/blog/better-language-models/

[16] Chimege. Chimege API documentation[EB/OL]. [2026-09-23]. https://chimege.com/