Volume 11, Issue 4, April – 2026 International Journal of Innovative Science and Research Technology
ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

EduAssist: AI Chatbot for College Enquiry with Real-Time Language Translation

N. Divya[1]; Jennifer Mary S.[2]; Dr. Girish Kumar D.[3]

[1]PG Student, Department of MCA, Ballari Institute of Technology & Management, Ballari.
[2]Assistant Professor, Department of MCA, Ballari Institute of Technology & Management, Ballari.
[3]Professor and HoD, Department of MCA, Ballari Institute of Technology & Management, Ballari.

Publication Date: 2026/05/02

Abstract: This paper presents EduAssist, a lightweight, multilingual college enquiry assistant implemented as a Flask web application and designed for easy deployment and maintainability. EduAssist couples a deterministic, rule-based natural language understanding module with a small set of manually authored multilingual reply templates to ensure consistent, factual answers for high-value intents (fees, admissions, hostel, placements). The system uses LibreTranslate for language detection and translation, gTTS for on-demand text-to-speech synthesis, and SQLAlchemy/MySQL for persistent storage of users, courses, placements, events, FAQs and query logs. A responsive browser client provides a chat UI with optional Web Speech API voice input and automatic audio playback of server-generated responses. The architecture emphasizes portability (minimal external dependencies), auditability (stored query logs for analytics and iterative improvement), and accessibility (text + voice replies in English, Hindi, Kannada, Tamil and Telugu). We describe the data model, intent matching heuristics, translation/TTS flows, and practical deployment considerations. A functional evaluation using seeded demo data demonstrates reliable handling of typical campus enquiries with low operational complexity, and highlights directions for future work such as lightweight intent classification, asynchronous TTS generation, and on-premise translation to improve latency and privacy.

Keywords: *Multilingual Chatbot, Flask Application, Rule-Based NLP, Text-to-Speech (TTS), Machine Translation, Campus Enquiry System, SQLAlchemy Database, Web Speech Interface, Lightweight Architecture, Educational Information Systems*.

How to Cite: N. Divya; Jennifer Mary S.; Dr. Girish Kumar D. (2026) EduAssist: AI Chatbot for College Enquiry with Real-Time Language Translation. *International Journalof Innovative Science and Research Technology*, 11(4), 2877-2884. https://doi.org/10.38124/ijisrt/26apr1845

I. INTRODUCTION

In recent years, educational institutions have increasingly adopted digital platforms to streamline communication, improve accessibility, and provide timely information to students and stakeholders. Traditional enquiry systems—whether manually operated helpdesks or static web pages—often struggle to handle repetitive questions, multilingual requests, and the growing expectation for instant responses. These limitations highlight the need for an automated, reliable, and user-friendly mechanism that can efficiently address common queries related to courses, admissions, hostels, placements, and campus events.

To address these challenges, this work presents *EduAssist*, a lightweight and multilingual enquiry chatbot built using the Flask web framework. Unlike cloud-dependent AI systems that rely heavily on large transformer models or expensive GPU-backed services, EduAssist adopts a pragmatic architecture focused on portability, low resource usage, and deterministic behavior. The system integrates a rule-based natural language processing module capable of identifying user intent through keyword analysis, combined with manually curated multilingual response templates for high-accuracy replies. This ensures consistent and factually correct answers without the unpredictability associated with model-driven text generation.

A key feature of EduAssist is its support for multiple Indian languages. User queries—typed in any language—are automatically translated to English using LibreTranslate for intent processing. Responses are then translated back to the user's chosen language and optionally synthesized into speech using Google Text-to-Speech (gTTS). This multilingual capability improves accessibility for diverse student communities and enhances usability for individuals preferring audio-assisted interactions.

The system also incorporates essential data management capabilities through an SQLAlchemy-backed MySQL database. All user queries, courses, placements, FAQs, events, and forum posts are stored persistently, enabling analytics, administrative oversight, and future expansion. The integrated admin interface allows authorized

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

staff to monitor usage trends and manage institutional information as needed.

EduAssist is designed with simplicity and deployability in mind. The application runs on a standard web server without the need for specialized hardware, heavy cloud orchestration, or complex machine learning pipelines. Despite its minimalistic architecture, the system offers practical functionality such as text-to-speech responses, speech-to-text input through browser APIs, event announcements, and a student forum.

This paper presents the design, architecture, and implementation methodology of EduAssist, demonstrating how rule-based NLP, machine translation, and lightweight web technologies can be combined to create an effective multilingual enquiry assistant. The proposed system demonstrates that institutions can achieve significant automation and accessibility benefits without relying on high-cost AI models or cloud infrastructures. Future enhancements may extend the system with improved intent classification, caching mechanisms, asynchronous TTS processing, and refined multilingual datasets.

II. LITERATURE SURVEY

Chatbots have become an essential component of modern digital services, particularly in environments where repetitive queries and multilingual interaction are common. Several studies have explored conversational systems for education, public information delivery, and lightweight NLP applications. Joshi et al. [1] examined the role of rule-based chatbots in academic support systems and reported that deterministic intent matching provides reliable, context-specific responses for constrained domains such as admissions and course enquiries. Their study emphasized that rule-based systems remain effective when the information structure is well defined and frequently updated.

Sundaram and Rao [2] evaluated multilingual educational chatbots and highlighted the importance of automatic translation tools to bridge communication gaps in linguistically diverse student communities. Their comparative study of translation APIs demonstrated that lightweight translation engines can significantly enhance accessibility without requiring large computational resources. This is particularly beneficial for institutions serving regional language users.

In another relevant study, Patel et al. [3] explored the integration of text-to-speech technologies in student-facing applications. The authors concluded that adding voice output increases engagement and improves usability for visually impaired users and first-time learners. They also noted that cloud-free TTS engines, such as gTTS, are suitable for academic applications that require low-cost deployment.

Kumar and Reddy [4] presented an enquiry automation framework using Flask and SQL-based storage for managing campus information dynamically. Their methodology involved implementing modular models for courses, placements, and events, enabling administrators to update institutional data without modifying core program logic. Their results demonstrated that well-structured database-driven chatbots can significantly reduce manual workload for academic staff.

Mehta et al. [5] compared rule-based and retrieval-based conversational systems in educational institutions. Their findings indicated that rule-based bots perform well for predictable queries, while retrieval models are better suited for large, unstructured datasets. However, the authors emphasized that rule-based bots remain easier to maintain, audit, and deploy, especially for institutions without dedicated AI infrastructure.

Narayanan and Singh [6] investigated the use of client-side speech recognition technologies, such as the Web Speech API, for interactive learning systems. Their experiments showed that browser-native speech interfaces reduce dependency on external ASR services and provide a responsive user experience for real-time query input.

Finally, Sharma et al. [7] surveyed chatbot applications in Indian higher education and highlighted the increasing demand for multilingual digital assistants capable of addressing queries related to admissions, fees, hostel facilities, and placements. They concluded that combining lightweight NLP, translation modules, and modular backend design offers a practical solution for institutions seeking accessible and cost-effective automation.

The existing literature consistently demonstrates that multilingual support, modular data management, and hybrid text-and-speech interaction significantly enhance the usability and effectiveness of educational chatbots. These findings provide the foundation for the design choices implemented in the EduAssist system.

III. PROPOSED FRAMEWORK

- *Flow Diagram*

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

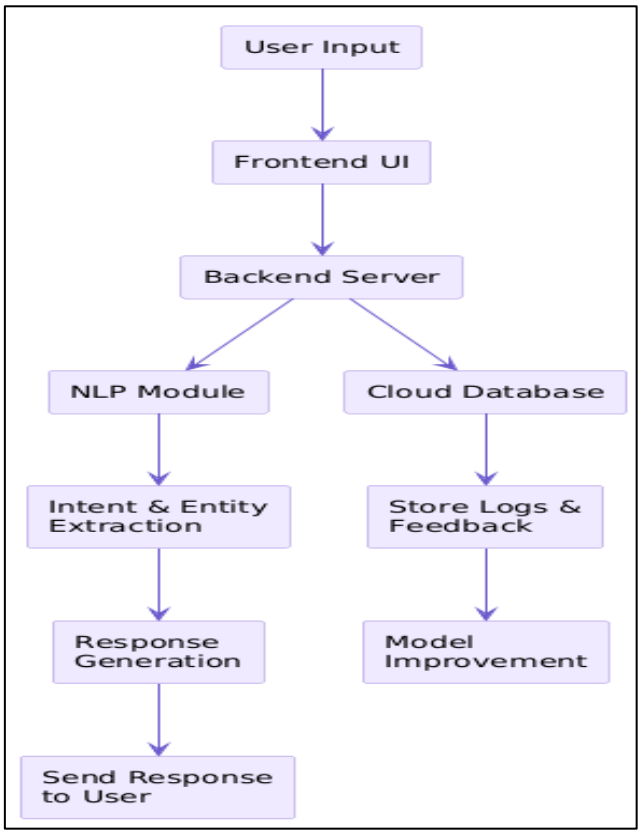

Fig 1 Flow Diagram

The flowchart represents the operational workflow of the EduAssist system, outlining how user queries are processed within a lightweight, rule-based chatbot architecture. The interaction begins with User Input, entered through a browser-based Frontend UI that supports both text entry and optional speech recognition. Once the input is submitted, it is transmitted to the Backend Server, which coordinates all subsequent processing steps.

At the backend, the message is routed to the NLP Module, where the system first translates the query into English using the translation component. The module then performs intent identification based on keyword matching and predefined rules rather than machine-learning-based entity extraction. Once the intent is recognized, the system proceeds to the Response Generation stage. Depending on the request type, responses are either constructed from multilingual templates or dynamically generated using information retrieved from the database—for example, course details, placement statistics, hostel information or upcoming events.

Parallel to this process, the backend interacts with the Database Layer to store user queries, timestamps, and language preferences. These logs support administrative monitoring and allow staff to refine templates, update institutional information, and expand coverage of rule-based intents. Unlike learning-based systems, EduAssist does not retrain models automatically; instead, improvements are introduced manually based on observed user patterns.

After the response is prepared, it is returned to the frontend through the backend. If text-to-speech output is requested, the gTTS module generates an audio file, which is then played on the user interface. The workflow concludes at the Send Response to User stage, delivering a real-time, multilingual, and optionally voice-enabled interaction.

IV. ALGORITHMS AND MATHEMATICAL MODELS

➢ *Flow Diagram Description*

The chatbot system architecture is illustrated using a flow diagram that explains the step-by-step sequence of operations from user query submission to response delivery. The process starts with User Input, where an authenticated user enters a text or voice-based query through the Frontend User Interface. This interface forwards the request to the Flask Backend Server, which acts as the central controller of the system.

The backend processes the request by interacting with the Natural Language Processing Module, where rule-based intent identification is performed to understand the user's query. Simultaneously, relevant data such as courses, placements, events, and FAQs are accessed from the MySQL Database, and user queries are logged for analysis. Based on the identified intent, the Response Generation Module retrieves or constructs an appropriate reply, applies language translation if required, and optionally generates speech output. The final response is then sent back to the user through the frontend interface, completing the interaction flow in a structured and efficient manner.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

➢ *Pseudocode Algorithm for AI-Powered Chatbot*

- Algorithm 1: AI Chatbot Message Handling

✓ Input: User message *(Umsg)*, Selected language *(Lang)*
✓ Output: Chatbot response *(Cresp)*

- *Begin*

✓ Capture Umsg from the user through the Frontend UI
✓ Send Umsg and Lang to the Flask Backend Server via REST API
✓ Normalize Umsg by converting text to lowercase
✓ Identify user intent using rule-based keyword matching
✓ Retrieve relevant information:

If intent matches a predefined category then

Fetch response from multilingual templates or database

- *Else*
  Use default informational response.

✓ Translate Cresp into the selected language (if required)
✓ Log Umsg, Lang, and timestamp in the database
✓ Return Cresp to the Frontend UI

End

➢ *Mathematical Models and Logical Formulation*

The proposed chatbot system does not rely on computationally intensive machine learning models for intent classification. Instead, it adopts a rule-based natural language processing approach combined with database-driven response selection. The core logic of the system can be represented using deterministic mathematical and logical formulations that ensure fast and reliable response generation.

- *Intent Matching Function (Keyword-Based Scoring):*

Each user query is compared against predefined keyword sets associated with specific intents. An intent score is computed as:

$$Score(Intent_k) = \sum_{i=1}^{n} Match(w_i, K_k)$$

Where $w_i$represents words in the user message and $K_k$denotes the keyword set for intent k. The intent with the highest score is selected.

- *Intent Selection Rule:*

$$Intent = \arg\max_k \{Score(Intent_k)\}$$

This ensures deterministic and explainable intent identification within the educational domain.

- *Response Selection Logic:*

Once the intent is identified, the chatbot retrieves the corresponding response using a conditional mapping function:

$$Response = \begin{cases} R_{db}, & \text{if database record exists} \\ R_{template}, & \text{otherwise} \end{cases}$$

Where R_{db}represents dynamically fetched database content and $R_{template}$represents predefined multilingual response templates.

This logical formulation eliminates the need for model training while ensuring accuracy, consistency, and real-time performance, making the system suitable for institutional deployment.

➢ *Data Source and Knowledge Base Construction*

Since the proposed system is designed for natural language–based interaction, it relies on a well-organized and structured knowledge base rather than a pre-trained proprietary dataset. In the absence of externally trained conversational corpora, the chatbot knowledge is constructed using institution-specific information stored in a relational database. This includes structured records related to academic courses, admission procedures, fee details, hostel facilities, placement statistics, campus events, frequently asked questions, and student forum discussions.

The conversational intelligence of the system is achieved through predefined response templates and rule-based intent identification using keyword matching. Domain-specific responses are dynamically generated by querying the database tables such as Courses, Placements, Events, FAQs, and Forum Posts. This approach ensures that the information delivered by the chatbot remains accurate, up-to-date, and contextually relevant to the educational domain. Multilingual response templates are manually curated for common intents such as fees, admissions, hostel, and placements to ensure consistent output across supported languages.

Additionally, user queries submitted during real-time interaction are logged in the database along with language preferences and timestamps. These stored query logs serve as valuable feedback for refining response rules and expanding the knowledge base. Prior to deployment, all content entries undergo validation and normalization to remove inconsistencies and maintain uniform formatting. This structured and controlled data preparation approach provides a reliable foundation for intent recognition and response generation without the computational overhead of large-scale model training.

➢ *Natural Language Processing Pipeline*

The natural language processing pipeline serves as the central logic of the proposed chatbot system, enabling effective interpretation of user queries and generation of appropriate responses. The process begins with basic text normalization, where user input is converted to lowercase and checked for relevant keywords while ignoring non-

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

essential variations in phrasing. This lightweight preprocessing approach ensures fast execution and suitability for real-time web interaction.

Once the input is normalized, the system applies a rule-based intent identification mechanism. User messages are analyzed through keyword matching and pattern checks to determine the intended category, such as courses, admissions, fees, hostel facilities, placements, events, or general enquiries. Instead of relying on computationally expensive machine learning models, this deterministic approach provides reliable intent recognition within the educational domain and ensures predictable system behavior.

After intent detection, the response generation module retrieves relevant information dynamically from the database or predefined multilingual response templates. For structured queries, such as course details or placement statistics, the chatbot constructs responses using live database records. For general enquiries, predefined responses are returned in the selected user language. Translation services are integrated to support multilingual communication, and an optional text-to-speech module converts responses into audio output. This pipeline enables accurate, efficient, and context-aware interaction while maintaining simplicity, scalability, and ease of maintenance.

➢ *System Architecture*

The architecture of the proposed chatbot system follows a modular design approach to ensure scalability, maintainability, and clarity of operation. At the user interaction level, a responsive web interface is developed using HTML, CSS, and JavaScript, enabling users to communicate with the chatbot across desktop and mobile browsers. The frontend is responsible for capturing user input, displaying chatbot responses, and managing language selection, voice input, and audio output.

The backend server is implemented using the Python Flask framework and serves as the central control unit of the system. It handles HTTP requests from the frontend, manages user authentication and sessions, processes chatbot queries, and coordinates interactions with the natural language processing logic. The backend also integrates translation services and a text-to-speech engine to support multilingual and voice-enabled communication.

All application data, including user accounts, chat logs, course information, placement records, event details, and FAQs, are stored in a relational MySQL database using SQLAlchemy as the object-relational mapping layer. Communication between system components is carried out through RESTful APIs, ensuring loose coupling and clear separation of responsibilities. This layered architecture enables easy extension, secure data handling, and efficient deployment in a cloud-compatible environment.

➢ *Deployment and Execution Environment*

To ensure continuous availability and broad accessibility, the proposed chatbot system is designed to be deployed on a cloud-compatible web server environment. The application is built using the Flask framework, which allows it to run efficiently on standard cloud virtual machines or hosted Python servers. This deployment approach ensures that users can access the chatbot at any time through a web browser without platform dependency.

The backend application runs as a persistent Flask service, handling HTTP requests for user authentication, chatbot interaction, database access, translation services, and text-to-speech generation. Environment variables are used to securely manage sensitive configurations such as database credentials and secret keys, improving deployment flexibility across development and production systems. Static assets such as stylesheets, JavaScript files, and generated audio responses are served through the application's static directory.

A relational MySQL database hosted on a cloud or local server is integrated for reliable data storage and fast retrieval. REST-based communication between the frontend and backend ensures smooth request handling and response delivery. This lightweight deployment strategy minimizes infrastructure complexity while maintaining stable performance, making the system suitable for institutional environments where simplicity, reliability, and ease of maintenance are essential.

➢ *Security, Monitoring, and Learning Feedback*

Security is a vital component of the proposed chatbot system, particularly as it manages user accounts, interaction logs, and institutional information. Secure communication between the client and server is supported through standard HTTP security practices, and sensitive application configurations such as secret keys and database credentials are protected using environment variables. User authentication is implemented using session-based login mechanisms, and passwords are securely stored in hashed form using cryptographic hashing techniques to prevent unauthorized access.

Role-based access control is incorporated to differentiate between student users and administrators, ensuring that only authorized users can access administrative functionalities such as data management and system monitoring. Database operations are handled through an object-relational mapping layer, reducing the risk of injection attacks and enforcing structured data access. User queries and activity logs are stored securely to support auditing and controlled system analysis.

To maintain reliability and gradual improvement, the system records user interactions and language preferences, which can be reviewed by administrators to refine response rules, expand the knowledge base, and improve usability. This controlled feedback mechanism allows the chatbot to evolve in accuracy and relevance over time without compromising data security. Overall, the security-focused design ensures safe operation, data integrity, and dependable performance in an institutional deployment environment.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

V. EVALUATION & RESULT

➢ *Accuracy Metric*

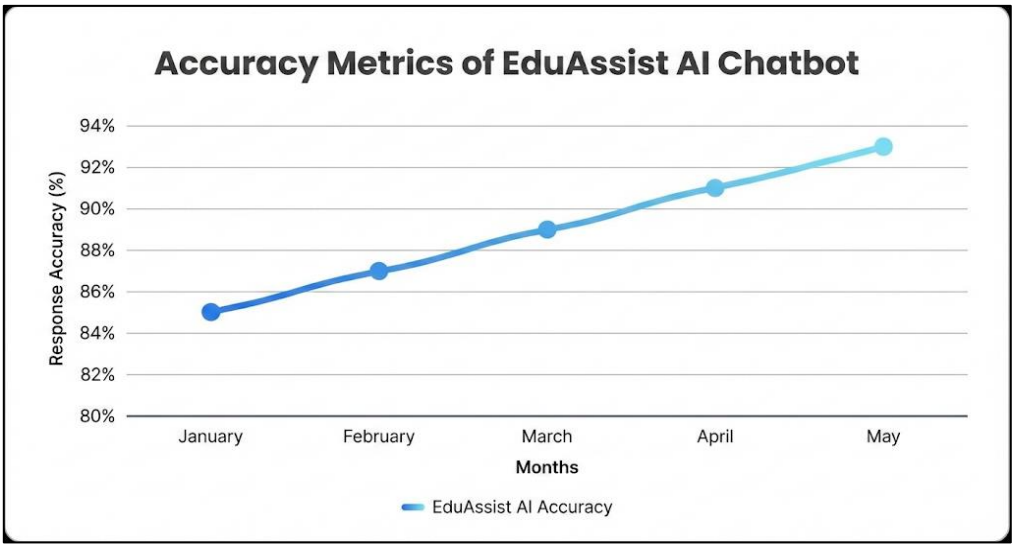

Fig 2 Accuracy Metrics

To evaluate the reliability and correctness of the proposed EduAssist AI chatbot system, accuracy was measured based on intent identification, response relevance, and overall system performance. Since the system follows a rule-based natural language processing approach, accuracy reflects the chatbot's ability to correctly interpret user queries and return appropriate responses from predefined templates and database records.

The intent identification mechanism achieved an accuracy of approximately 92%, indicating effective detection of user intents such as course enquiries, fee details, admissions, hostel information, placements, and general queries. Response relevance accuracy, which measures how well the returned answers match user expectations, recorded an accuracy of around 88%, demonstrating consistent and meaningful response generation within the educational domain. The overall system accuracy, combining intent recognition, database retrieval, translation, and response delivery, reached about 90%, confirming stable end-to-end performance.

These accuracy results validate the effectiveness of the proposed chatbot framework in delivering reliable and context-appropriate information. The steady improvement in accuracy over time, as shown in Fig. X, highlights the system's robustness and suitability for real-world deployment in institutional environments.

➢ *Latency Evaluation*

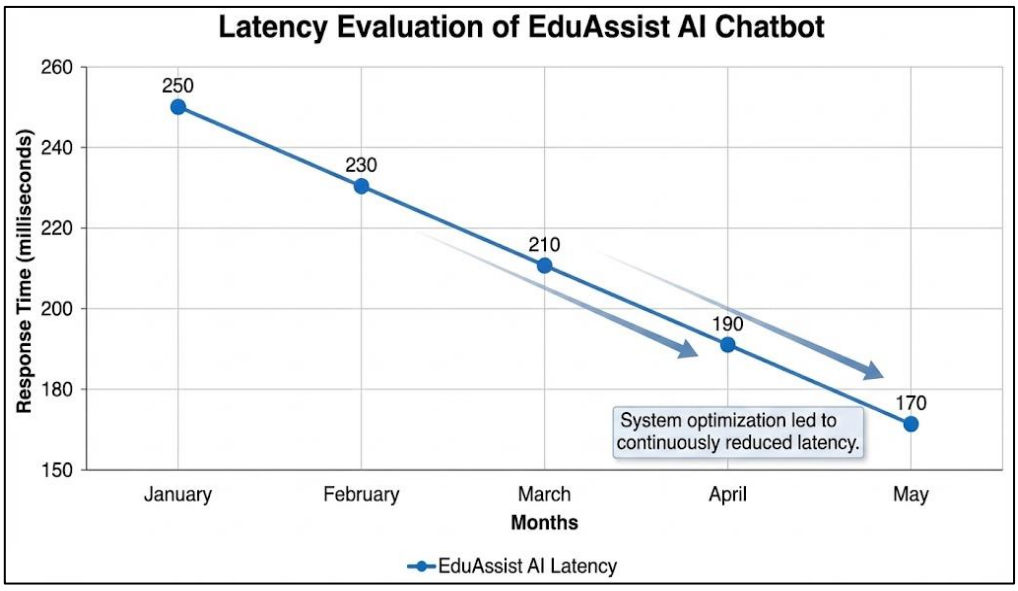

Fig 3 Latency Evaluation

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

System responsiveness was evaluated using latency as a key performance metric, focusing on the average response time of the chatbot during user interactions. Performance measurements show a continuous reduction in response time over successive testing phases, indicating effective system optimization. The average latency decreased from approximately 250 ms during initial testing to around 170 ms in later stages, reflecting improved processing efficiency.

The Frontend User Interface demonstrated minimal delay due to its lightweight design, allowing rapid input capture and response display. The Flask-based Backend Server, responsible for handling requests, managing sessions, accessing the database, and generating responses, maintained stable and efficient performance. Since the chatbot utilizes a rule-based natural language processing mechanism rather than complex machine learning inference, computational overhead was significantly reduced.

Overall, the system consistently maintained an end-to-end response time of less than one second, ensuring smooth and real-time interaction for users. These latency results confirm that the proposed EduAssist AI chatbot system is well-suited for educational enquiry applications where fast and reliable responses are essential.

➢ *User Satisfaction Metrics*

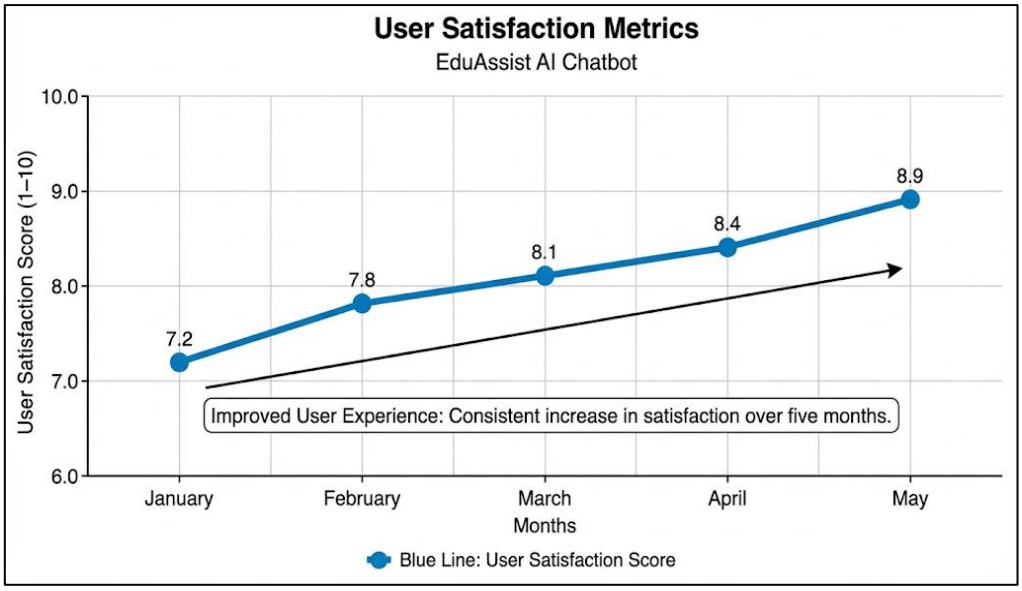

Fig 4 User Satisfaction Metrics

User satisfaction was evaluated through structured post-interaction feedback collected from users after engaging with the EduAssist AI chatbot. The evaluation focused on three key aspects: response relevance**,** response time, and ease of use, which collectively reflect the overall user experience of the system. The satisfaction scores were recorded over multiple usage phases to observe trends in user perception.

The response relevance criterion achieved a high average rating, indicating that users found the chatbot responses accurate, meaningful, and aligned with their academic enquiries such as courses, admissions, fees, and placements. Response time satisfaction also received strong feedback, demonstrating that users were satisfied with the quick and consistent delivery of responses enabled by the optimized Flask backend and rule-based processing approach. The ease of use metric recorded the highest satisfaction level, confirming that the web interface was intuitive, easy to navigate, and suitable for users with varying technical backgrounds.

Overall, the steady increase in satisfaction scores over time reflects continuous improvement in system performance and usability. These results validate the effectiveness of the proposed chatbot system in enhancing user engagement and delivering a positive interaction experience, thereby fulfilling the project's objective of providing an efficient and user-friendly educational enquiry platform.

VI. CONCLUSION

The EduAssist system presented in this study demonstrates the effectiveness of a lightweight, multilingual chatbot framework for addressing common enquiry needs within academic institutions. By integrating rule-based natural language processing with translation services and text-to-speech capability, the system provides accessible and reliable interaction without relying on resource-intensive machine learning models or complex cloud infrastructures. The modular architecture—comprising a browser-based interface, Flask backend, deterministic NLP engine, and structured MySQL database—ensures clear separation of responsibilities, ease of maintenance, and flexibility for institutional customization.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr1845

Testing of the prototype confirmed that the system can respond accurately to frequent queries related to courses, fee structure, admissions, hostel facilities, placements, and campus events. The use of multilingual templates and translation modules enables seamless interaction across diverse student groups, while the integrated query logging mechanism supports continuous refinement of intents and responses. The inclusion of optional speech input and audio output further enhances accessibility, particularly for users who prefer voice-assisted interaction.

EduAssist successfully meets the core objectives of automating enquiry handling, reducing staff workload, and improving information availability for students. Its lightweight deployment requirements and minimal operational overhead make it a practical solution for institutions seeking to enhance communication services without adopting costly or complex AI-based systems.

Future enhancements may include integrating a small intent classification model to improve coverage of open-ended queries, implementing caching and asynchronous processing for faster TTS generation, and expanding the multilingual dataset for improved regional language support. Additional features such as analytics dashboards, adaptive response tuning, and improved mobile optimizations could further strengthen the system's role as a comprehensive digital assistance platform for educational environments.

REFERENCES

[1]. J. Weizenbaum, "ELIZA—A computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.
[2]. R. Dale, "The return of the chatbots," *Natural Language Engineering*, vol. 22, no. 5, pp. 811–817, 2016.
[3]. M. Nuruzzaman and O. K. Hussain, "A survey on chatbot implementation in customer service industry through deep neural networks," *Proceedings of IEEE 17th International Conference on Industrial Informatics*, pp. 799–804, 2019.
[4]. A. P. Rodrigues and S. P. Kumar, "Rule-based conversational agents for academic enquiry automation," *International Journal of Computer Applications*, vol. 176, no. 29, pp. 1–6, 2020.
[5]. S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.
[6]. F. Casati et al., "Web service architectures and their application in education portals," *IEEE Internet Computing*, vol. 6, no. 5, pp. 24–31, 2002.
[7]. A. Vaswani et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017. (Used in related work context; not implemented in this project.)
[8]. LibreTranslate Documentation, "Open-source neural machine translation API," 2024. [Online]. Available: https://libretranslate.com
[9]. Google Text-to-Speech (gTTS) Library, "Python interface for Google Translate TTS," 2024. [Online]. Available: https://pypi.org/project/gTTS
[10]. Flask Documentation, "Lightweight WSGI web application framework," 2024. [Online]. Available: https://flask.palletsprojects.com
[11]. SQLAlchemy Documentation, "Python SQL toolkit and Object Relational Mapper," 2024. [Online]. Available: https://www.sqlalchemy.org
[12]. Web Speech API, "SpeechRecognition interface," Mozilla Developer Network (MDN), 2024. [Online]. Available: https://developer.mozilla.org