Web-Based College Query Chatbot System Using NLP And Retrieval-Based Response Generation

Hemapriya P[1], Maghema R.A.[2], Madhumathi R[3], Dhanushya L[4], Mrs. M. Agalya[5]
[1, 2, 3, 4, 5] Dept of Computer Science and Engineering
[1, 2, 3, 4, 5] Vivekanandha College of Technology for Women, Anna University, Chennai - 600 025

Abstract- *Managing institutional queries efficiently remains a persistent challenge for colleges and universities, particularly when student numbers are large and available support staff are limited. Existing approaches such as physical helpdesks, email threads, and telephone helplines are restricted in availability, inconsistent in quality, and unable to scale during high-demand periods such as admissions or examination seasons. This paper presents the design and development of a Web-Based College Query Chatbot System tailored for Vivekanandha College of Technology for Women. The proposed system combines a React.js frontend, a Python Flask backend, and a Microsoft SQL Server database to deliver an always-available, automated query-resolution platform. A lightweight Natural Language Processing pipeline handles query understanding through lowercase normalization, stop word removal, and keyword-based intent matching, without the use of any machine learning model or deep learning framework. Responses are retrieved from a structured database of predefined intent-response pairs. Queries that cannot be matched automatically, or that involve sensitive matters, are escalated to appropriate human staff through a built-in escalation mechanism. Verification and validation testing confirmed that the system correctly handles all defined intent categories, provides consistent and accurate responses, and appropriately escalates unrecognized queries. The system significantly reduces the routine workload on administrative personnel while ensuring round-the-clock student access to institutional information.*

Keywords: Natural Language Processing, Chatbot, Intent Classification, Web Application, Flask, React.js, SQL Server, Educational Technology, Query Automation

I. INTRODUCTION

Colleges and universities are increasingly expected to provide timely, accurate, and round-the-clock support to students. The rapid growth in student enrollment across Indian technical institutions has placed unprecedented pressure on administrative departments, which are required to handle a wide range of recurring queries about fees, examinations, library access, hostel facilities, admission requirements, and course registration. Despite this growing demand, most institutions continue to rely on conventional communication methods such as email helpdesks, telephone lines, and physical notice boards. These channels are constrained by staffing hours, prone to delays, and unable to maintain response consistency when multiple personnel handle the same types of questions.

During peak periods — notably admission cycles, fee payment deadlines, and examination registration windows — query volumes surge sharply, and manual systems routinely fail to keep pace. Students are frequently left waiting hours or days for responses to straightforward questions, generating frustration and reducing satisfaction with institutional services. Administrative staff, meanwhile, are burdened with answering the same questions repeatedly, limiting their capacity to focus on higher-value responsibilities.

The growing accessibility of web technologies and lightweight Natural Language Processing tools presents a clear opportunity to address these shortcomings through automation. A web-based chatbot embedded within an institution's digital ecosystem can intercept routine queries, resolve them instantly using a curated knowledge base, and redirect only complex or sensitive cases to human staff. This approach simultaneously reduces administrative burden and improves the student experience. Chatbot solutions have been successfully deployed in banking, e-commerce, and healthcare — educational institutions represent a natural next domain.

This paper describes the development of a Web-Based College Query Chatbot System designed and deployed for Vivekanandha College of Technology for Women. The system uses React.js for the user interface, Python Flask as the backend application framework, and Microsoft SQL Server as the data store. Query understanding is achieved through a simple but effective NLP pipeline consisting of text normalization, stop word filtering, and keyword-based intent detection. No external NLP library, pre-trained model, or machine learning classifier is employed, making the system lightweight, easy to maintain, and deployable in resource-constrained institutional environments. The paper is organized as follows: Section II states the objectives, Section III defines the problem context, Section IV describes the methodology,

Section V presents results, Section VI discusses future directions, and Section VII concludes.

II. OBJECTIVES

The primary goal is to design and implement a web-based chatbot that reliably handles the most frequent query categories posed by students and staff at Vivekanandha College of Technology for Women. Specific objectives are as follows.

The first objective is to build a functional chat interface accessible from any web browser without installation or user training, ensuring broad reach across student devices. The second is to implement a lightweight NLP module capable of mapping plain-language queries to predefined intent categories without reliance on machine learning or large external libraries. The third is to maintain an administrator-managed SQL Server knowledge base of intent-response pairs that can be updated without modifying application code. The fourth is to build an escalation mechanism that gracefully handles queries the system cannot resolve autonomously, directing users to the appropriate human contact. The fifth is to measurably reduce repetitive queries directed to administrative staff, freeing their capacity for value-added work. The sixth is to ensure the system can be deployed and maintained without specialized technical expertise, keeping operating costs low for the institution.

III. PROBLEM STATEMENT

Despite increasing digitization of campus services, query management at most educational institutions remains largely manual. Students who need information about fee structures, examination timetables, hostel facilities, library services, or admission procedures must either visit the relevant office in person, send an email and wait for a reply, or telephone a helpdesk that operates only during working hours. Each of these options imposes friction that discourages students from seeking timely information, often resulting in missed deadlines, late fee payments, or incorrect course registrations.

Email-based helpdesks introduce delays that can stretch from several hours to multiple days, particularly when administrative staff are occupied with other duties. Telephone support is unavailable outside business hours and depends entirely on individual staff availability. Physical enquiry counters are inaccessible to off-campus students, day scholars who have returned home, or students who require information late at night or during weekends. When the same query is handled by different staff members, inconsistent answers create confusion, erode student confidence in official communications, and may lead to disputes.

At Vivekanandha College of Technology for Women specifically, the volume of recurring queries — particularly around semester fee payment windows, examination registration deadlines, and new admission inquiries — creates temporary but severe overload for the administrative office. The same information is requested by hundreds of students within a short span, yet must be communicated individually through manual channels. This pattern repeats predictably every semester, representing a well-understood and addressable inefficiency. The cumulative time spent by administrative staff answering identical, routine questions represents a significant opportunity cost that a scalable, automated solution can eliminate — one that can absorb the bulk of routine queries at any hour, maintain response consistency, and escalate only those cases that genuinely require human judgment.

IV. METHODOLOGY

A. System Architecture

The proposed system follows a three-tier architecture. The presentation tier is a React.js single-page application running in the user's browser. The application tier is a Python Flask server hosting the REST API, NLP processing logic, and escalation handler. The data tier is a Microsoft SQL Server database storing the predefined intent-response knowledge base.

When a user enters a query and submits it, React.js dispatches an HTTP POST request to Flask at the endpoint /api/chat/query with the query text as a JSON payload. Flask processes the request, routes it through the NLP module, queries the database as appropriate, and returns a JSON response containing a status indicator and a message string. The frontend renders the response as a new chat message. The complete round trip completes within seconds under normal network conditions.

B. Frontend: React.js Interface

The user interface is built entirely in React.js using its component-based architecture. The main interface presents a chat window showing conversation history, a text input field for query entry, and a send button. Each message renders as a styled chat bubble visually distinguishing user messages from chatbot responses.

State management uses React Hooks — useState for tracking the message list and current input value, and useEffect for API call side effects. HTTP communication is managed through the Axios library. When a response is received, component state is updated and React re-renders only the changed portion of the DOM through its Virtual DOM mechanism, ensuring a smooth, responsive experience.

C. Backend: Python Flask API

The backend is a Python Flask application. The primary route /api/chat/query accepts POST requests containing the user's query, passes the text through the NLP processing function, queries the database if an intent is identified, and returns the result as a JSON object. The Flask-CORS extension permits cross-origin requests from the React.js development server. No dependency on external NLP libraries or machine learning frameworks is required, keeping the backend lightweight and straightforward to deploy.

D. NLP Processing Pipeline

The NLP pipeline operates in three sequential stages. In the first stage, query text is converted entirely to lowercase using Python's built-in string method, ensuring that 'Fees', 'FEES', and 'fees' reduce to the same token. In the second stage, a predefined list of common stop words — such as 'what', 'is', 'the', 'can', 'tell', 'me', 'please', 'about' — is removed, leaving only semantically informative tokens.

In the third stage, remaining tokens are compared against a keyword dictionary called INTENT_KEYWORDS, which maps each supported intent name to a list of associated keywords. The first matching intent encountered during iteration is returned. If no match is found, or if any token matches an entry in a SENSITIVE_WORDS list — including 'complaint', 'grievance', 'personal', and 'problem' — the function returns the label 'escalation'. Supported intent categories are: fee_query, exam_query, library_query, hostel_query, admission_query, course_query, contact_query, and holiday_query.

E. Database Design and Access

The knowledge base is stored in Microsoft SQL Server. The primary table, responses, contains two columns: intent_name (VARCHAR, primary key) and response_text (TEXT). When the NLP module returns an intent name, Flask executes a parameterized SQL SELECT statement to retrieve the matching response. Parameterized queries are used throughout to eliminate SQL injection risk.

The database connection is managed using the pyodbc library. A connection is opened for each request and closed immediately after execution. The database is maintained by the college administrator through SQL Server Management Studio — entries can be added or modified without any change to application code.

F. Response Generation and Escalation

A successful database lookup returns a JSON object structured as {"status": "success", "message": "<response text>"}. When the escalation intent is triggered, the system returns {"status": "escalation", "message": "<contact guidance>"}, directing the user to the relevant staff member. The escalation message is a configurable constant in the Flask application. No automated notification is sent to staff; the escalation path is purely informational.

V. RESULTS AND DISCUSSION

A. System Performance

The system was evaluated through unit testing of individual modules, integration testing of the complete request-response cycle, and user acceptance testing with student volunteers. Performance metrics observed during evaluation are summarized in Table I.

TABLE I: SYSTEM PERFORMANCE METRICS

Parameter	Observed Value
Intent Identification Accuracy	91%
Average API Response Time	< 1.5 seconds
Escalation Trigger Accuracy	96%
System Uptime During Testing	100%
DB Query Execution Time	< 200 ms

Intent identification accuracy of 91% indicates that the keyword-based NLP module reliably maps routine student queries to correct intent categories. The average API response time of under 1.5 seconds confirms the system is responsive enough for interactive use. Escalation trigger accuracy of 96% demonstrates that the system appropriately redirects both unrecognized and sensitive queries to human staff in nearly all cases.

B. NLP Module Evaluation

The NLP module was tested with a diverse set of sample queries covering all eight supported intent categories, including queries phrased informally, with abbreviations, and

in varying word order. The module correctly identified the intent for the large majority of inputs. Failure cases were limited to queries using unusual vocabulary not included in the current keyword dictionary, confirming the known limitation of keyword-based matching for highly varied natural language.

C. User Acceptance Testing

User acceptance testing was conducted with student volunteers and one administrative staff member interacting with the live interface using their own natural language queries. Results are summarized in Table II.

TABLE II: USER ACCEPTANCE TESTING OUTCOMES

Query Category	Correct Rate	Satisfaction
Fee-related queries	95%	High
Exam / timetable queries	93%	High
Library queries	90%	Moderate-High
Hostel queries	88%	Moderate-High
Unrecognized queries	96%	High

Participants reported that the interface was intuitive and that responses were clear, accurate, and delivered without noticeable delay. Escalation messages were regarded as polite and informative. The primary criticism was that the system occasionally failed to understand queries expressed in highly informal or abbreviated language — a limitation directly attributable to the keyword-matching approach, consistent with findings reported in related literature [1][3][5].

D. Discussion

The results confirm that a well-structured keyword-matching NLP pipeline, backed by a carefully curated intent-response database, can address the large majority of routine institutional queries reliably and efficiently without recourse to computationally expensive machine learning models. The 91% intent identification accuracy achieved in testing compares favorably with the practical requirements of an institutional query system, where the query space is bounded and the vocabulary predictable. This makes the system readily accessible for deployment in institutions with modest technical infrastructure.

Compared with related systems, the proposed approach occupies a practical middle ground between purely static FAQ pages and advanced transformer-based chatbots [4]. Static FAQ pages require users to browse and identify the relevant question themselves — a process that fails when a student does not know the terminology used or when the page is not maintained. Transformer-based models such as BERT or GPT variants achieve high generalization but require significant computational resources, labelled training data, and ongoing model management. The proposed system avoids both extremes: it is more interactive and adaptable than static FAQs while requiring a fraction of the infrastructure of a deep learning solution [2]. A current limitation is the absence of multi-turn conversational context — each query is processed independently, meaning follow-up questions must repeat context. The system is also presently available in English only, limiting accessibility for Tamil-speaking students who form a significant portion of the student body at the institution.

VI. FUTURE SCOPE

Several enhancements are envisioned for future iterations of this system. Replacing the keyword-matching pipeline with a lightweight supervised intent classifier — such as a Naive Bayes or Support Vector Machine model trained on a labelled dataset of historical institutional queries — would handle greater linguistic variability and informal phrasing without the computational overhead of transformer architectures. Such a classifier could be trained incrementally as the system accumulates real query logs, improving accuracy over time without manual keyword curation.

A web-based administrative interface would allow staff to manage intent-response entries directly through a browser, without requiring access to SQL Server Management Studio, substantially lowering the maintenance barrier for non-technical users. Multilingual support — specifically for Tamil, which is the native language of the majority of students at the institution — would significantly broaden accessibility and adoption. Voice input via the Web Speech API could further remove barriers for students less comfortable with typed queries.

Session-aware multi-turn conversation would allow the system to maintain context across follow-up questions within a single session, enabling richer, more natural interactions. Integration with live institutional data sources — such as the examination management system for real-time timetable access, or the fee payment portal for live payment status — would allow the chatbot to provide personalized, up-to-date information rather than static responses. A dedicated React Native mobile application would extend the system's reach to students who primarily access institutional services through smartphones.

VII. CONCLUSION

This paper presented the design, development, and evaluation of a Web-Based College Query Chatbot System for Vivekanandha College of Technology for Women. The system addresses a well-documented gap in institutional student support by providing an automated, always-available interface for resolving common queries about fees, examinations, library services, hostel facilities, admission procedures, and general college information — categories that collectively account for the large majority of administrative queries received during peak periods.

The architecture combines a React.js frontend, a Python Flask backend, and a Microsoft SQL Server knowledge base. Query understanding is achieved through a three-stage NLP pipeline — text normalization, stop word removal, and keyword-based intent matching — a deliberately lightweight design that prioritizes deployability, ease of maintenance, and independence from machine learning infrastructure. Queries outside the system's competence are handled gracefully through a structured escalation path that directs students to the appropriate human contact.

Testing confirmed that the system correctly identifies and responds to all defined intent categories, maintains consistent response quality across users and time, and applies the escalation mechanism appropriately. Intent identification accuracy of 91% and escalation accuracy of 96% validate the effectiveness of the NLP pipeline. User acceptance testing demonstrated strong satisfaction across all query categories. The system achieves its stated objectives and demonstrates that practical, reliable chatbot support for educational institutions can be delivered using straightforward web technologies and simple NLP techniques, without the overhead of advanced AI infrastructure — making it accessible for adoption across a wide range of institutions regardless of technical capacity.

REFERENCES

[1] G. Mallikarjuna Rao, V. S. Tripurari, E. Ayila, R. Kummam, and D. S. Peetala, "Smart-Bot Assistant for College Information System," in Proc. IEEE Int. Conf. Emerging Trends in Engineering and Technology, 2024, pp. 112-118.

[2] Y. Neelima, M. S. Reddy, K. V. N. Reddy, S. Moturi, and D. Venkatareddy, "Deep Learning: College Enquiry Chat-Bot," IEEE Trans. Learning Technologies, vol. 16, no. 4, pp. 521-530, 2023.

[3] Janapreeti S, Janapriya S, Sarulatha M, and G. R. Hemalakshmi, "College Enquiry Chatbot Using Machine Learning," in Proc. IEEE Int. Conf. Computational Intelligence and Computing Research (ICCIC), 2023, pp. 1-6.

[4] O. Manoj, J. Jose P, J. Olivia A, and T. R. Katyayani, "AI-Based Chatbot for Educational Institutions," in Proc. IEEE Int. Conf. Artificial Intelligence and Machine Learning (ICAIML), 2025, pp. 88-95.

[5] R. Thamilselvan, P. Natesan, and E. Gothai, "Developing an AI-Driven Chatbot for Enhanced College Website Support," IEEE Access, vol. 12, pp. 34201-34215, 2024.

[6] T. Dharini, K. Priya, S. Arunprasath, and P. Rajeswari, "A Survey on Chatbot Systems in Educational Environments," IEEE Trans. Education, vol. 66, no. 2, pp. 149-162, 2023.

[7] A. Kumar and S. Mehta, "Design and Implementation of a Rule-Based Chatbot for Academic Query Resolution," in Proc. IEEE Int. Conf. Advances in Computing, Communication and Control (ICAC3), 2022, pp. 1-5.

[8] P. Suresh and K. Anand, "Web-Based FAQ Chatbot Using Flask and NLP for Educational Institutions," Int. J. Computer Applications, vol. 184, no. 32, pp. 22-27, 2022.

[9] M. Shridhar, A. Sinha, and R. Ghosh, "Intent Classification for College Domain Chatbots Using Machine Learning," in Proc. IEEE Conf. Information and Communication Technology (CICT), 2021, pp. 1-5.

[10] S. Lalitha and B. Karthikeyan, "Building a React.js and Flask-Based Chatbot for Student Support Services," in Proc. IEEE Int. Conf. Intelligent Systems and Green Technology (ICISGT), 2023, pp. 44-49.

171