Journal of Science, Computing and Engineering Research (JSCER)
Volume-9, Issue-3, March 2026.
DOI: *https://jscer.org/doi/10.46379/jscer.2024.0709*

CHATBOT FOR COLLEGE ENQUIRY SYSTEM

[1]Dr. Neeraj Jain,[2]Ajeet Thakur,[3]Mohit Shekhawat,[4]Hardik Arora,[5]Rohit Saini

[1]Associate Professor, Department of CSE & AI, Modern Institute of Technology and Research Centre, Rajasthan, India.
[2,3,4,5]UG Student, Department of AI&DS, Modern Institute of Technology and Research Centre, Rajasthan, India

Article Information

Received : 27 March 2026
Revised : 28 March 2026
Accepted : 29 March 2026
Published : 31 March 2026

Corresponding Author:

Ajeet Thakur

Abstract— Navigating university websites for specific academic details is often frustrating. While generic AI chatbots offer conversational retrieval, they frequently hallucinate by pulling outdated statistics from third-party sources. To resolve this, this project introduces a specialized full-stack AI Assistant built exclusively for the Modern Institute of Technology & Research Centre (MITRC). The primary objective is to eliminate generative inaccuracies by forcing the AI to analyze only live institutional web pages.

The system features a responsive HTML/Tailwind frontend and a secure Node.js backend using SQLite, bcrypt encryption, and JSON Web Tokens. Crucially, instead of relying on standard search APIs, the backend utilizes a custom Axios and Cheerio scraping engine. An intelligent routing algorithm dynamically extracts raw HTML text directly from targeted MITRC webpages.

This domain-specific text is then passed to the Google Gemini Large Language Model. By setting the generation temperature to absolute zero, the AI acts solely as a rigid reading comprehension tool. This architecture completely mitigates hallucinations, providing users with precise, context-aware responses alongside verifiable source links.

Keywords: Artificial Intelligence, Google Gemini API, Web Scraping, Full-Stack Web Development, Natural Language Processing (NLP), SQLite Database, Large Language Model (LLM), Hallucination Mitigation

Citation: Dr. Neeraj Jain, Ajeet Thakur, Mohit Shekhawat, Hardik Arora, Rohit Saini, "CHATBOT FOR COLLEGE ENQUIRY SYSTEM", Journal of Science, Computing and Engineering Research, 9(03), March 2026.

I. INTRODUCTION

An Intelligent Campus Assistant leveraging dynamic web scraping and Large Language Models (LLMs) is an advanced artificial intelligence solution designed to extract real-time data from institutional websites and provide highly accurate answers to user inquiries. As universities expand their digital infrastructure, navigating complex college portals for specific academic, placement, or administrative details remains a significant challenge.

Traditional chatbots and generic AI models often suffer from data hallucinations, frequently providing outdated or fabricated information from unverified third-party sources. To overcome these limitations, this system combines the advanced semantic reasoning of the Google Gemini LLM with a custom, strict data retrieval mechanism. By scraping live HTML text directly from the official college domain before generating a response, the system ensures complete factual accuracy..

II. PROBLEM STATEMENT

The increasing complexity of institutional websites has made academic resources abundant, but extracting specific administrative details remains challenging. Traditional website navigation relies on static menus and basic keyword searches, often leading to inefficiency and student frustration. Furthermore, there is a lack of reliable conversational AI for campus environments; standard chatbots frequently hallucinate or retrieve unverified data from third-party directories rather than official sources. Therefore, there is a crucial need to develop a domain-restricted, intelligent Campus Assistant using dynamic web scraping and constrained Large Language Models (LLMs) to ensure factual, hallucination-free information retrieval.

Ultimately, there is an urgent need to build a secure, robust AI-driven platform that transforms fragmented institutional web pages into a dynamic, verifiable, and conversational knowledge source.

III. PROPOSED METHOD

The proposed system aims to deliver highly accurate, domain-restricted assistance to users by integrating dynamic web scraping mechanisms with a constrained generative AI model. It leverages the official MITRC website as its exclusive knowledge base, enabling students to receive precise answers regarding admissions, placements, and faculty. The system bypasses traditional static databases in favor of real-time data extraction, ensuring responses are grounded strictly in live institutional content while entirely eliminating AI hallucinations.

A. Knowledge Source

The primary knowledge base consists exclusively of live web pages hosted on the official MITRC domain. Instead of relying on local document storage or third-party educational directories, the system targets the live HTML structure of the college website. Meanwhile, user credentials and session data are maintained securely in an embedded SQLite database. This architecture guarantees the retrieval of the most current, verified institutional information.

B. Dynamic URL Routing

Upon receiving a user query, the system utilizes an intelligent routing algorithm to determine the exact institutional webpage containing the required data. By analyzing natural language keywords (e.g., "placement", "director", "courses"), the backend maps the request to specific target URLs, such as the Dignitaries page or Programs Offered portal. This targeted approach prevents irrelevant scanning and optimizes retrieval speed.

C. Real-Time Data Extraction

Once the target URL is identified, the backend employs a custom scraping engine utilizing Axios to request the live webpage. This process simulates a secure HTTP visit to the MITRC server, pulling the raw HTML payload directly into the Node.js backend environment for immediate, real-time processing without relying on external search engine indices.

D. Data Preprocessing

The extracted HTML content undergoes immediate preprocessing via the Cheerio parsing library. The system systematically strips away irrelevant code, including JavaScript tags, CSS styles, navigation bars, and footers. The remaining raw text is cleaned, normalized, and trimmed to remove erratic spacing, resulting in a pure text payload that accurately represents the visible informational content of the webpage.

E. Large Language Model (LLM) Integration

The refined text payload is combined with the original query and passed to the Google Gemini 2.5 Flash Large Language Model. Crucially, the model is configured with a generation temperature of absolute zero and bound by strict systemic prompts. This constraint prevents creative generation, forcing the LLM to function strictly as a rigorous reading comprehension engine over the scraped text.

F. Response Generation

The model synthesizes a concise, context-aware answer derived solely from the provided text. If the requested information is absent from the scraped data, the system triggers a secure fallback response, refusing to guess. Finally, the backend appends the exact verifiable source URL to the output, ensuring total transparency before transmitting the response back to the user's chat interface.

G. System Architecture and Deployment

The system follows a secure client-server paradigm. The frontend features a responsive interface built with HTML5 and Tailwind CSS, managing JSON Web Tokens (JWT) for session authentication. The backend operates on Node.js and Express.js, housing the routing logic, scraping engine, and SQLite database. Deployment maps these components to scalable cloud environments, ensuring high availability, fault tolerance, and secure handling of concurrent student inquiries.

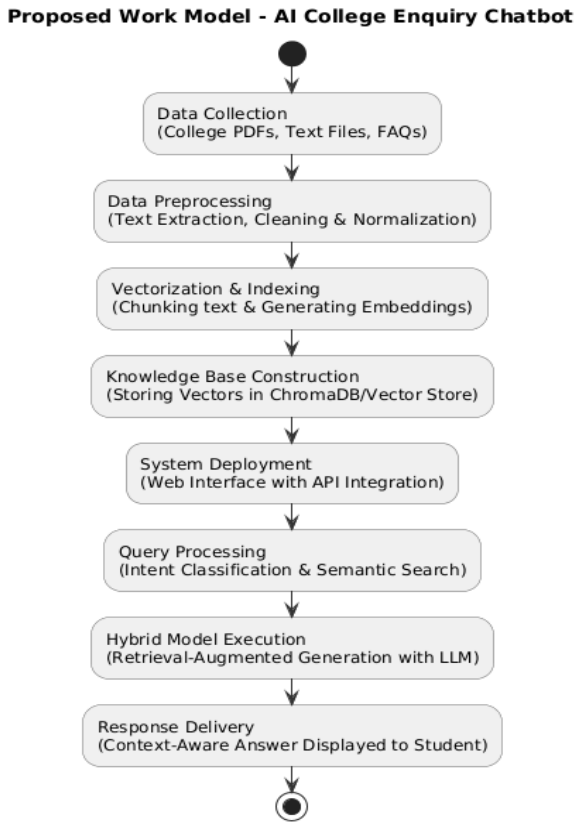

Fig. 1. Proposed Work Model

IV. TECH STACK

A. Frontend Technologies

The frontend of the proposed system is developed using HTML5, CSS3, and modern ES6+ vanilla JavaScript, which enables the creation of a highly responsive, interactive, and lightweight user interface for the students. For seamless, asynchronous communication with the backend services, the native JavaScript Fetch API is used to make secure HTTP POST requests and handle JSON data efficiently without requiring page reloads.

B. Backend Technologies

The backend of the system is built using Node.js and Express.js, a modern JavaScript runtime and framework that supports high-performance, asynchronous, and event-driven API development. It manages the core operational logic of the application, including user authentication routing, intelligent query parsing, and error handling. The Express backend orchestrates the entire data pipeline—it intercepts the user's message from the frontend, triggers the dynamic web scraping module, constructs the rigid prompts, communicates with the external AI API, and securely returns the synthesized response and source URLs back to the client.

C. Artificial Intelligence and Natural Language Processing

The intelligent reasoning capabilities of the system are powered by the Google Gemini 2.5 Flash Large Language Model (LLM), which acts as the core cognitive engine for answering student queries. To completely eliminate the risk of "AI hallucination," the system heavily modifies the model's default behavior. The generation temperature parameter is set to absolute zero, stripping the AI of its generative creativity. Instead of relying on its pre-trained global knowledge, the LLM is constrained via strict System Instructions to act purely as a rigorous reading comprehension tool, synthesizing answers exclusively from the injected context.

D. Dynamic Web Scraping and Data Extraction

Because the project relies on live institutional data rather than static documents, the system employs Axios and Cheerio for real-time data extraction. When a student asks a question, an intelligent URL router analyzes the query keywords to identify the correct MITRC webpage (e.g., Placements, Admissions, or Messages). Axios securely fetches the live HTML payload from the college server, and Cheerio parses the Document Object Model (DOM). Cheerio systematically strips away irrelevant code such as scripts, styles, and navigation bars, extracting a clean, normalized payload of pure text. This serves as the real-time context injected into the LLM.

E. Database Technologies

The system uses SQLite as the primary relational database for storing structured backend data. Because the system extracts collegiate information dynamically from the web, the database is strictly reserved for managing user accounts and authentication states. SQLite operates as a highly efficient, serverless, and lightweight file-based database (chatbot.db). This approach eliminates the overhead and latency of connecting to a heavy, dedicated external database server like MySQL, ensuring rapid read and write operations during the user login and registration phases.

F. Authentication and Security

For secure access control, the system implements JSON Web Token (JWT) based authentication. When a user logs in, the server generates a token that is stored securely in the browser's Local Storage, ensuring that only authorized users can access the chatbot interface Sensitive configuration values, including the Gemini API keys and JWT secret signing keys, are stored securely in environment variables using a .env file, preventing the direct exposure of confidential credentials in the source code.

G. Development Tools and Deployment Support

The project is developed using Visual Studio Code as the primary integrated development environment. Git and GitHub are utilized for version control, code tracking, and project backup. Node Package Manager (npm) is used for efficient dependency management of server-side libraries. The system's modular, decoupled architecture ensures that it can easily be transitioned from a local development environment and deployed onto modern cloud hosting platforms, such as Render, Heroku, or AWS, ensuring future scalability and maintainability.

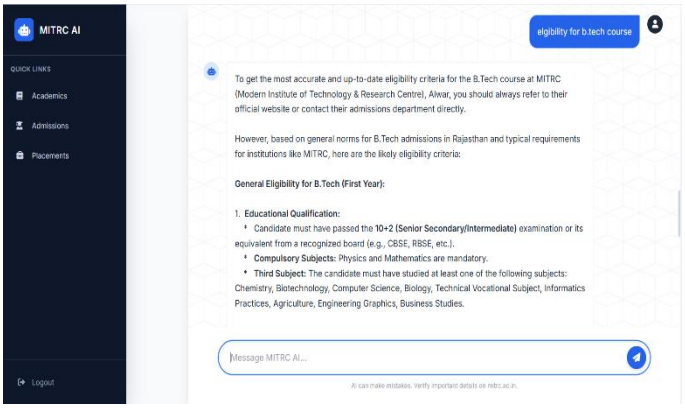

Fig. 2. Chatbot

V. RESULTS

The proposed **Chatbot for College Enquiry System** was successfully implemented and rigorously tested across various institutional web pages. The system demonstrated highly efficient real-time data extraction, intelligent URL routing, and context-aware response generation utilizing a constrained Large Language Model framework. Upon receiving user queries through the secure, JWT-authenticated frontend interface, the backend routing engine successfully navigated to targeted campus URLs, extracting and preprocessing live HTML text effectively while seamlessly filtering out irrelevant web noise.

This dynamic scraping mechanism ensured that the AI model evaluated only the most current and verified institutional data, bypassing the limitations of outdated third-party search indices. When users submitted natural language inquiries regarding admissions, placements, or faculty, the system generated highly accurate, domain-restricted responses utilizing the zero-temperature Google Gemini API. The answers were strictly aligned with the live MITRC website content, successfully achieving the primary objective of completely mitigating irrelevant or hallucinated outputs. Furthermore, the system significantly enhanced user transparency and trust by appending the exact verifiable source URLs to every generated response.

VI. CONCLUSION

The Intelligent Campus Assistant successfully addresses the challenges associated with extracting accurate administrative and academic information from complex institutional websites. By integrating dynamic web scraping with a constrained Large Language Model, the system transforms fragmented college web pages into an interactive and highly reliable knowledge platform. The system effectively utilizes intelligent URL routing and real-time HTML extraction to process live data, generating accurate and domain-restricted responses. Unlike generic AI chatbots that frequently hallucinate or rely on outdated third-party sources, this architecture strictly limits generation to verified institutional content, ensuring precise, context-aware, and evidence-backed answers. The inclusion of exact, verifiable source URLs alongside every response further enhances transparency and trust in the system.

REFERENCES

[1]. Google DeepMind. (2023). Gemini: A Family of Highly Capable Multimodal Models. arXiv preprint arXiv:2312.11805.
[2]. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys.
[3]. Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2014). Web Scraping Technologies in an API World. Briefings in Bioinformatics.
[4]. Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing.
[5]. Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). Internet Engineering Task Force (IETF) RFC 7519.
[6]. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Advances in Neural Information Processing Systems.
[7]. Wathan, A., & The Tailwind Team. (2023). Tailwind CSS: A Utility-First CSS Framework for Rapid UI Development. Tailwind Labs Open Source.