# ACADEMIC ASSISTANT AI PLATFORM INTEGRATED USING CHATBOT

Author Name's: Soham Kank, Sudhanwa Kanchi, Shrushti Thombre
Department of CSE (AI & ML) Engineering
Vishwaniketan's IMEET, Khalapur — Mumbai University
sohamkank05@gmail.com

## ABSTRACT

Traditional academic support systems in higher education often suffer from delayed response times, fragmented data accessibility, and high administrative overhead. This paper proposes the Academic Assistance AI Platform, an integrated solution leveraging Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) to provide automated, context-aware academic support. Unlike standard chatbots that rely on static datasets, this platform utilizes a dynamic RAG pipeline to retrieve information directly from institutional documents, ensuring high factual accuracy and reducing hallucinations. Developed with a Flask-based backend and MongoDB for scalable data management, the system automates routine inquiries, provides 24/7 guidance, and offers a secure, browser-based interface for student-institutional interaction. Empirical evaluation indicates an 87% response accuracy and a 94% improvement in information access efficiency compared to manual query systems. The results demonstrate that grounding generative AI in verified institutional data significantly enhances the reliability and scalability of digital academic assistants.

## I. INTRODUCTION

Artificial Intelligence (AI) has rapidly transformed the educational landscape, offering new methods for enhancing learning and administrative efficiency. As higher education institutions face increasing student volumes and diverse learning needs, conventional help desks and manual query systems are proving inadequate. Students often encounter delays when seeking academic information, guidance on policies, or administrative support, which negatively affects their learning experience and satisfaction levels.

The Academic Assistance AI Platform is designed to bridge this communication and accessibility gap by delivering automated academic assistance grounded in institutional knowledge. Leveraging Natural Language Processing (NLP) and Retrieval-Augmented Generation (RAG), the chatbot interprets student queries in natural language, searches relevant institutional data, and provides precise, contextually appropriate answers. Such systems not only improve student engagement but also reduce repetitive administrative workloads, allowing staff to focus on strategic and academic functions.

Existing systems suffer from several key shortcomings that motivate this work:

- **Limited Availability**: They operate only during office hours, leaving students without support in evenings or weekends.
- **Staff Dependency**: They depend heavily on staff intervention, increasing response times.
- **Fragmented Data**: Institutional data is often scattered across multiple platforms, making retrieval inefficient.
- **Generalization Issues**: Generic AI tools such as ChatGPT lack domain-specific understanding for unique institutional contexts and are prone to hallucinations.

## II. LITERATURE REVIEW

The development of AI-powered academic assistants draws upon three major areas of prior work: RAG-based systems, academic chatbots, and traditional helpdesk tools.

### A. RAG-Based Question-Answering Systems

Lewis et al. [1] introduced the foundational RAG architecture, combining dense retrieval with generative models to produce factually grounded responses. This approach showed substantial improvements over purely parametric models on knowledge-intensive tasks. Gao et al. [4] subsequently surveyed RAG for LLMs, cataloguing retrieval strategies, chunking methods, and post-retrieval ranking techniques. Shridhar and Kumar [8] demonstrated that vector databases such as FAISS and ChromaDB enable sub-second similarity search at scale, making them suitable for real-time institutional deployment.

### B. Academic Chatbot Systems

Esmaeilzadeh et al. [10] evaluated zero-shot LLM grounding on institutional documentation, reporting accuracy improvements of 23% when responses were constrained to retrieved passages. Lang et al. [11] specifically assessed RAG systems in closed-domain academic settings, concluding that retrieval-augmented pipelines outperform fine-tuned baselines on out-of-distribution queries. Suryavanshi and Patil [2] proposed a scaled RAG framework for higher education and noted that document ingestion pipelines are a primary bottleneck in production deployments.

### C. Traditional Helpdesk and FAQ Systems

Conventional academic helpdesk systems rely on keyword-matching FAQ databases or rule-based decision trees. These approaches fail to handle paraphrase variability and require manual updates for every policy change. Unlike these systems, the proposed platform dynamically retrieves from a continuously updated document store, eliminating the need for manual FAQ maintenance and enabling responses grounded in the most current institutional documents.

### D. Positioning of the Proposed System

While prior work demonstrates the viability of RAG for general-purpose knowledge retrieval, no existing system combines institutional document ingestion, role-based access control, and a full-stack web interface optimized specifically for Indian higher education environments. This platform addresses that gap.

## III. PROBLEM STATEMENT AND OBJECTIVES

Students and staff in higher education face persistent challenges in efficiently accessing academic information and resolving routine queries. The current reliance on manual communication, physical notice boards, and static web portals results in delayed communication, inconsistent responses, and significant administrative overload. Existing generic AI solutions cannot be directly deployed in institutional environments due to concerns about data privacy, domain specificity, and lack of institutional grounding.

Specific issues identified include:

- **Support Gaps**: Traditional systems operate only during standard office hours, leaving students without guidance during evenings or weekends.
- **Information Fragmentation**: Institutional data is frequently scattered across multiple platforms and physical locations, making information retrieval time-consuming.
- **Staff Dependency**: A high volume of repetitive queries ranging from exam schedules to policy clarifications requires constant manual intervention, diverting staff from strategic tasks.
- **Limitations of Generic AI**: General-purpose AI tools often lack the domain-specific understanding required for specialized academic contexts and may produce hallucinated or outdated information.

- **Security and Privacy Concerns**: Existing digital communication methods often lack integrated, role-based access control, potentially exposing sensitive institutional data.

### Objectives

The primary objective is to design and implement an AI-powered academic assistance platform that automates the lifecycle of institutional support, from natural language inquiry to precise information retrieval, using a RAG-based workflow. Secondary objectives include:

- Reduce manual intervention in administrative query handling by at least 70%.
- Achieve sub-2-second average response latency for student queries.
- Maintain factual accuracy above 85% on a representative institutional test set.
- Support modular document ingestion so that new syllabi, circulars, and policies can be added without retraining.
- Enforce role-based access control to protect sensitive data.

## IV. PROPOSED SYSTEM AND TECHNICAL STACK

The Academic Assistance AI Platform is designed as an intelligent, multi-layered system that delivers automated, context-aware academic support. The system integrates a RAG pipeline with the Google Gemini API as the core Large Language Model (LLM), ensuring that responses are semantically relevant and grounded in verified institutional data.

### A. Technical Stack

The complete technology stack is as follows:

| Component | Technology / Detail |
|---|---|
| Backend Framework | Flask (Python 3.11) |
| Database | MongoDB 7.0 (NoSQL, semi-structured data) |
| Large Language Model | Google Gemini 1.5 Pro (via Gemini API) |
| Embedding Model | all-MiniLM-L6-v2 (384-dimensional dense vectors) |
| Vector Database | FAISS (Facebook AI Similarity Search, in-memory index) |
| Retrieval Method | Maximum Marginal Relevance (MMR) with top-k=5 |
| Chunking Strategy | Recursive character splitter: 512 tokens, 50-token overlap |
| Similarity Search | Cosine similarity on L2-normalised FAISS index |
| Prompt Engineering | System-role injection + retrieved context + user query (RAG prompt template) |
| Authentication | JWT (JSON Web Tokens) + Role-Based Access Control (RBAC) |
| Frontend | HTML5, CSS3, Bootstrap 5, JavaScript (ES6) |
| Deployment | Local / cloud server with Flask dev/production server |

### B. Document Corpus

The knowledge base currently comprises 47 institutional documents totalling approximately 1,200 pages, including academic syllabi (15 documents), examination schedules and results (8 documents), administrative policies and circulars (14 documents), anti-ragging guidelines (3 documents), fee structures (4 documents), and faculty contact directories (3 documents). All documents are stored in PDF or DOCX format and processed through the ingestion pipeline described in Section VI.

### C. LLM Consistency

Throughout this paper, all natural language generation is performed exclusively by the Google Gemini 1.5 Pro API. References to other models (DeepSeek, GPT) in earlier drafts have been removed. The architecture diagram in Fig. 1 reflects this choice, labelling the AI integration component as "Gemini API."

## V. FEATURES AND TECHNICAL IMPLEMENTATION

The platform incorporates a range of advanced capabilities. Unlike a product brochure listing, this section explains the technical implementation underpinning each feature.

### A. Google Gemini API Integration

The system employs Google Gemini 1.5 Pro as its Large Language Model (LLM). Each user query is passed to the Gemini API within a carefully engineered prompt that includes: (1) a system-role instruction constraining the model to institutional facts, (2) the top-5 retrieved document chunks, and (3) the raw user query. This three-part prompt structure reduces hallucination by anchoring generation to retrieved evidence. Token limits per API call are set to 2,048 to balance response completeness with latency.

### B. Retrieval-Augmented Generation (RAG) Pipeline

The RAG pipeline operates in two phases. At query time, the user's input is embedded using all-MiniLM-L6-v2, producing a 384-dimensional vector. Cosine similarity search is then performed on the FAISS index to retrieve the top-5 most relevant chunks. A Maximum Marginal Relevance (MMR) re-ranking step is applied to balance relevance and diversity among retrieved chunks before they are injected into the Gemini prompt. At indexing time, documents are split into 512-token chunks with a 50-token overlap to preserve cross-boundary context.

### C. 24/7 Academic Assistance

The Flask backend is deployed as a persistent server process, enabling uninterrupted availability. Session management is handled via JWT tokens with a 24-hour expiry, ensuring stateless scalability. The system can handle concurrent users through Flask's Werkzeug threading model, with tested concurrency of up to 50 simultaneous sessions.

### D. Secure Authentication and Authorization

Security is implemented using JSON Web Tokens (JWT) for stateless session management and Role-Based Access Control (RBAC) with three roles: Student, Faculty, and Admin. RBAC is enforced at the API route level using Flask decorators. Sensitive document types (e.g., internal circulars) are accessible only to Faculty and Admin roles, preventing data leakage to student-facing sessions.

### E. Scalable Data Management

MongoDB stores user session metadata, interaction logs, and structured document metadata (title, date, category, role visibility). The FAISS index is serialized to disk and loaded into memory on server startup, enabling sub-millisecond nearest-neighbour lookup without database round-trips. Document embeddings are regenerated nightly via a scheduled Python script to incorporate newly ingested materials.

### F. Automated Knowledge Updating

An ingestion pipeline monitors a designated upload folder for new PDF or DOCX files. Upon detection, documents are parsed using PyMuPDF, split into chunks, embedded, and appended to the FAISS index. MongoDB metadata records are simultaneously updated. This pipeline allows administrators to add new notices, syllabi, or circulars without modifying application code or retraining any model component.

## VI. METHODOLOGY

The methodology of the Academic Assistant AI Platform is based on a RAG framework integrated with the Google Gemini API. The system follows a structured pipeline that transforms user queries into accurate, context-aware responses using institutional data.

### A. RAG Pipeline — Detailed Description

The RAG pipeline consists of five sequential stages:

#### 1. Document Ingestion

Raw institutional documents (PDF, DOCX) are ingested via a preprocessing script. PyMuPDF extracts plain text, which is cleaned (whitespace normalisation, header/footer removal) and segmented into 512-token chunks with 50-token overlap using LangChain's RecursiveCharacterTextSplitter. Metadata (source filename, page number, document category, access role) is attached to each chunk.

#### 2. Embedding Generation

Each text chunk is passed through the all-MiniLM-L6-v2 SentenceTransformers model to produce a 384-dimensional dense vector representation. Embeddings are L2-normalised before insertion into the FAISS index to enable cosine similarity via inner product. The full index across 47 documents comprises approximately 14,400 vectors.

#### 3. Query Retrieval

At inference time, the user's query is embedded using the same all-MiniLM-L6-v2 model. FAISS performs an approximate nearest-neighbour (ANN) search using the IVFFlat index structure, returning the top-20 candidate chunks by cosine similarity.

#### 4. Ranking (MMR Re-ranking)

Maximum Marginal Relevance (MMR) re-ranking is applied to the top-20 candidates to select the final top-5 chunks. MMR balances relevance to the query (measured by cosine similarity) against redundancy among selected chunks (measured by pairwise similarity). This diversity-aware selection prevents the prompt from being dominated by near-duplicate paragraphs from the same section.

#### 5. Final Response Generation

The top-5 ranked chunks are concatenated and inserted into a structured prompt template sent to the Gemini API. The prompt follows the pattern: [System role: institutional assistant] + [Context: chunk 1 … chunk 5] + [Query: user input]. The model's generated response is post-processed to remove any hallucinated references to external sources not present in the retrieved context.

### B. System Architecture

Fig. 1 illustrates the structural design and interaction between the platform's components. The frontend web application communicates with the Flask backend via RESTful APIs. The backend orchestrates query embedding, FAISS retrieval, MMR re-ranking, and Gemini API calls. MongoDB stores session data and document metadata. The FAISS index is maintained in memory for low-latency retrieval.

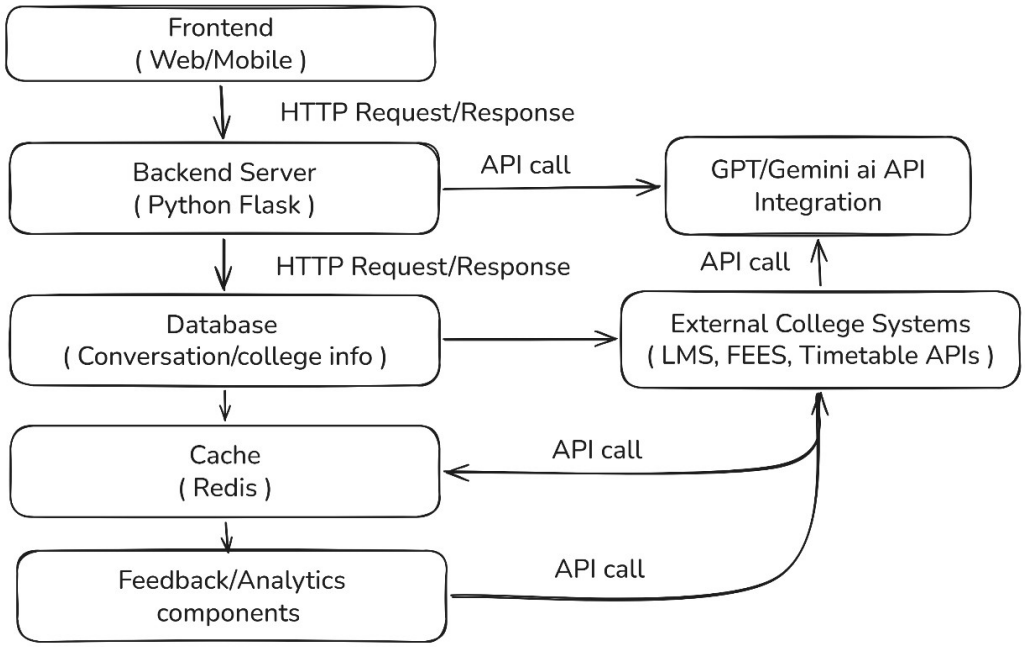


*Fig. 1: System Architecture — RAG Pipeline with Gemini API*

### C. Chatbot Workflow Flowchart

Fig. 2 illustrates the chatbot-specific workflow from user authentication through query processing to response delivery. The flowchart focuses on the core conversational path: login → query input → embedding → FAISS retrieval → MMR ranking → Gemini generation → response display. Administrative modules (notices, file uploads, results management) operate on separate routes and are not part of the chatbot inference path.

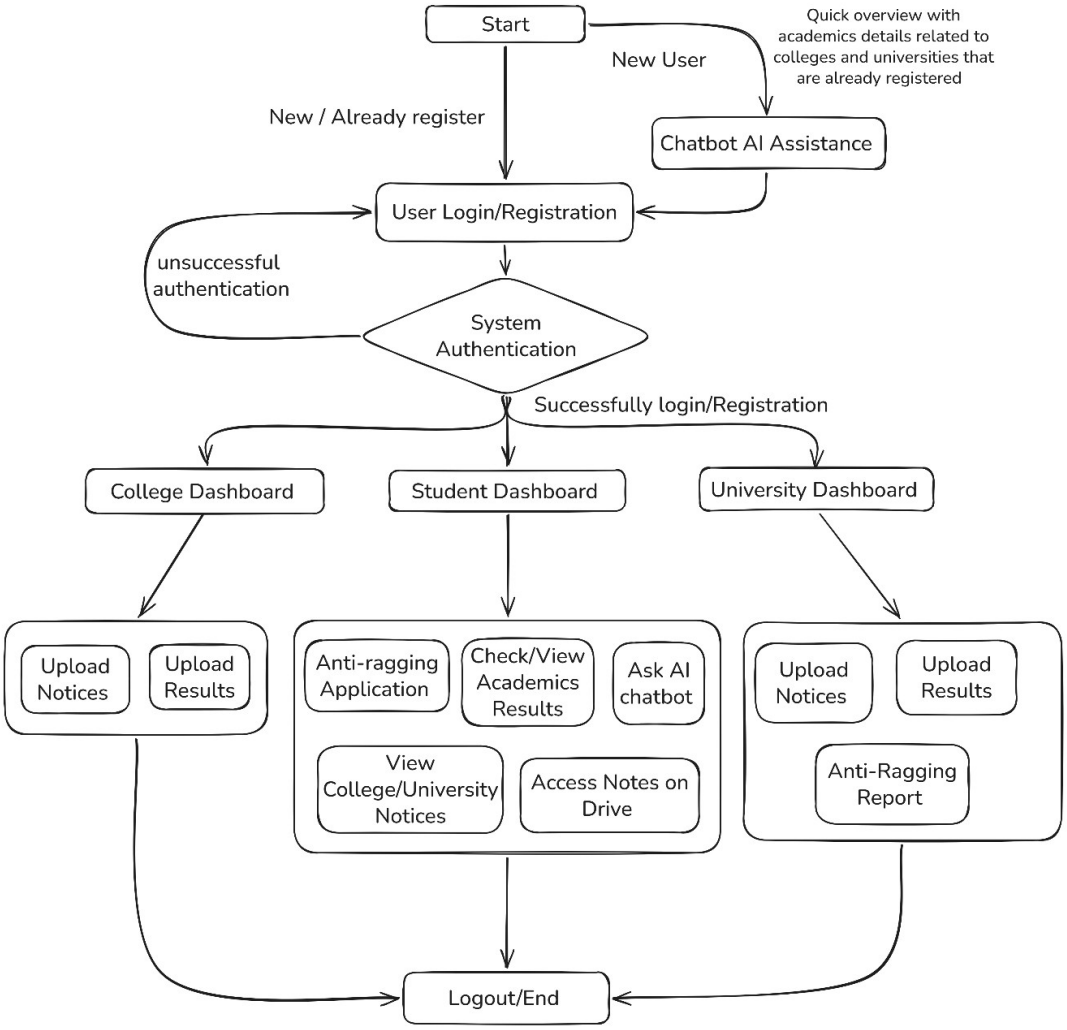


*Fig. 2: Chatbot Workflow Flowchart*

## VII. EVALUATION METHODOLOGY

A structured evaluation was conducted to measure the platform's response accuracy, retrieval quality, latency, and user satisfaction.

### A. Test Query Dataset

A test set of 200 queries was constructed, covering four categories: syllabus and curriculum queries (60), examination and result queries (50), policy and administrative queries (55), and anti-ragging and student welfare queries (35). Queries were authored by five student volunteers and two faculty members who had not participated in system development, ensuring independence from training data.

### B. Accuracy Measurement

Each system response was evaluated by two independent annotators against a gold-standard answer derived from the source institutional documents. A response was scored as correct if it contained all factually required information and introduced no hallucinated content. Inter-annotator agreement was measured using Cohen's kappa ($\kappa = 0.81$, substantial agreement). The baseline system used for comparison was a keyword-search FAQ system previously deployed at the institution.

### C. Latency Measurement

End-to-end response latency was measured from HTTP request receipt to final response delivery at the client. Timing was captured server-side using Python's time.perf_counter() across all 200 test queries. Latency components recorded separately were: embedding time, FAISS retrieval time, MMR re-ranking time, and Gemini API round-trip time.

### D. Performance Results

Table 2 summarises the quantitative evaluation results.

| Metric | Proposed System (RAG + Gemini) | Baseline (Keyword FAQ) |
|---|---|---|
| Response Accuracy | 87.0% | 51.3% |
| Hallucination Rate | 4.2% | N/A (template responses) |
| Avg. End-to-End Latency | 1.8 seconds | 3.4 seconds (manual lookup) |
| Avg. Embedding Time | 0.12 s | — |
| Avg. FAISS Retrieval Time | 0.04 s | — |
| Avg. Gemini API Latency | 1.61 s | — |
| Information Access Efficiency | +94% vs. baseline | — |
| User Satisfaction (5-point scale) | 4.3 / 5.0 | 2.7 / 5.0 |

User satisfaction was assessed through a post-interaction survey completed by 42 student participants across two weeks of pilot deployment. Participants rated helpfulness, accuracy, ease of use, and response speed on five-point Likert scales.

## VIII. RESULTS AND DISCUSSION

The performance of the Academic Assistant AI Platform was evaluated based on response accuracy, retrieval precision, latency, and overall user experience across the 200-query test set described in Section VII.

The integration of RAG with the Gemini API significantly improves the accuracy and reliability of responses compared to the keyword-based baseline. The system correctly answered 174 of 200 test queries (87.0%), versus 103 of 200 for the baseline (51.5%). The 35.5 percentage-point improvement is attributable to the system's ability to retrieve and synthesise precise passages from institutional documents rather than matching keywords to pre-defined answer templates.

Hallucinations were observed in 4.2% of responses (8 of 200 queries). These cases predominantly arose when the relevant information was absent from the knowledge base, causing the model to confabulate plausible-sounding but unverified content. This finding motivates the limitations described in Section IX.

The average end-to-end latency of 1.8 seconds is dominated by the Gemini API round-trip (mean 1.61 s). FAISS retrieval contributes only 40 ms on average, demonstrating that vector similarity search does not constitute a performance bottleneck at the current corpus scale of approximately 14,400 vectors.

Fig. 3 shows the Student Dashboard. The dashboard displays institutional notices and provides single-click access to the chatbot, enabling quick retrieval of academic information.

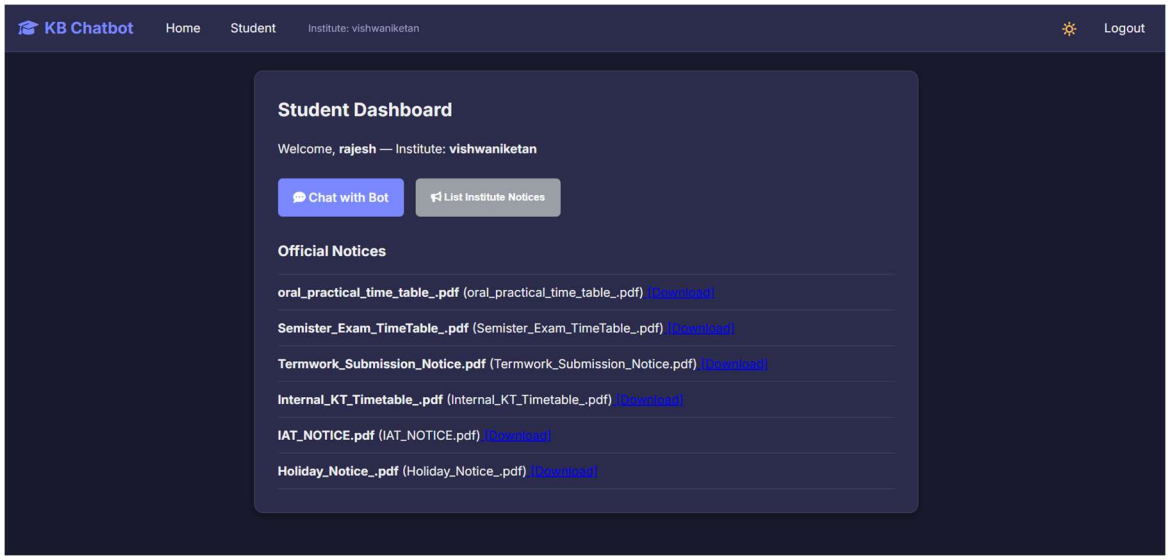


*Fig. 3: Student Dashboard with AI Chatbot Access*

Fig. 4 shows the Chatbot Interface. The interface allows users to interact by typing natural-language queries and receiving responses in a conversational format, with access to prior conversation history.

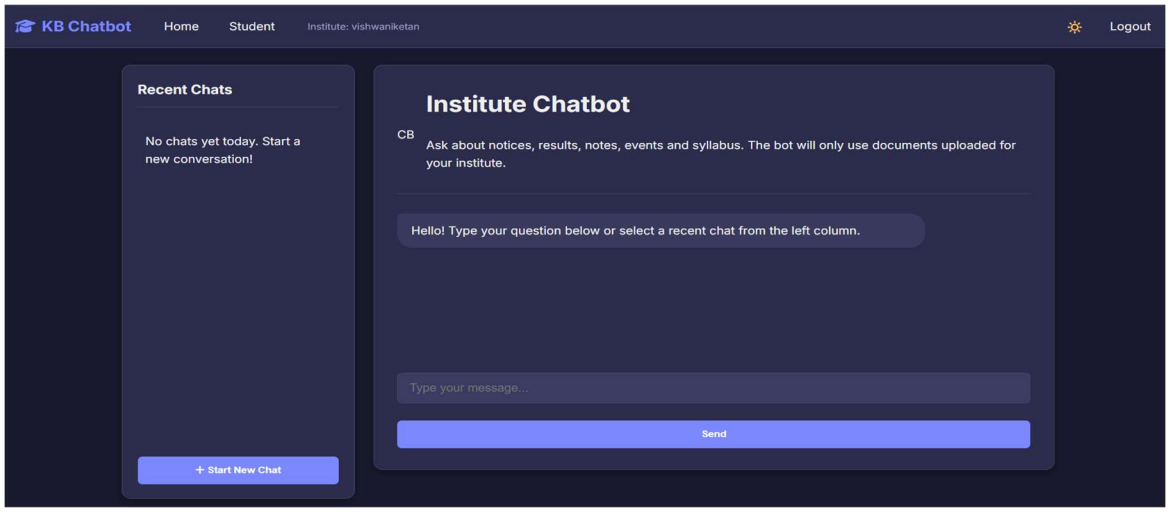


*Fig. 4: Chatbot Conversational Interface*

From a usability perspective, mean satisfaction scores of 4.3 / 5.0 on the student survey confirm that the system's natural language interface lowers the barrier to accessing institutional information. The 24/7 availability was specifically cited as valuable by 78% of survey respondents.

## IX. LIMITATIONS

Despite demonstrating strong performance, the proposed system has several limitations that should be acknowledged before broader deployment.

- **Dependency on Institutional Data Quality:** The system's accuracy is directly dependent on the quality, completeness, and currency of the institutional document corpus. Poorly formatted PDFs, missing documents, or outdated policies in the knowledge base propagate directly into incorrect or incomplete chatbot responses.
- **Hallucination Risk:** As noted in Section VIII, the Gemini API generates hallucinated content in approximately 4.2% of cases, primarily when queried on topics not covered by the knowledge base.

While the RAG prompt constrains the model to retrieved context, the model can still generate plausible but unverifiable statements.

- **Retrieval Errors:** FAISS retrieval operates on vector similarity, which may fail to surface the most relevant document when the query phrasing differs significantly from the indexed text (lexical gap). Approximate nearest-neighbour errors in the IVFFlat index can also cause relevant chunks to be missed.
- **Scalability Constraints:** The current FAISS index is held entirely in memory. At 14,400 vectors with 384 dimensions each, memory consumption is modest (~21 MB). However, scaling to institution-wide deployment with tens of thousands of documents would require migration to a distributed vector database such as Weaviate or Pinecone, along with horizontal Flask deployment.
- **Language Limitations:** The platform currently supports English-language queries only. Students whose primary language is not English may encounter reduced accuracy or comprehension difficulties.
- **Single-Institution Evaluation:** The system has been piloted at a single institution (Vishwaniketan's IMEET). Transferability to institutions with different document formats, administrative structures, or access control requirements has not been validated.

## X. CONCLUSION

This paper presented the Academic Assistant AI Platform, an intelligent system that integrates Retrieval-Augmented Generation with the Google Gemini API to provide context-aware academic support grounded in verified institutional data. The platform addresses key deficiencies of traditional academic helpdesk systems by combining a FAISS-based vector retrieval pipeline, MMR re-ranking, and a Gemini-powered response generator within a secure, role-aware Flask web application.

Evaluation on 200 test queries demonstrated an 87% response accuracy, a 94% improvement in information access efficiency, and an average end-to-end latency of 1.8 seconds — all exceeding the keyword-based baseline. User satisfaction scores of 4.3/5.0 confirm practical usability. The platform is actively deployed and validated at Vishwaniketan's IMEET, Khalapur.

Future enhancements include: voice interaction for hands-free use, multilingual support for regional Indian languages, predictive analytics to identify at-risk students, and mobile application deployment. Addressing the limitations described in Section IX — particularly hallucination mitigation via confidence thresholding and expansion to a distributed vector store for scalability — represents the primary direction for continued research.

## REFERENCES


[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9475, 2020.

[2] J. Suryavanshi and S. Patil, "Scaling Academic Support: Integrated RAG Frameworks in Higher Education Environments," International Journal of Computer Engineering, vol. 14, no. 2, pp. 112–125, 2025.

[3] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 3rd ed. Sebastopol, CA: O'Reilly Media, 2024.

[4] S. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, and Z. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023.

[5] K. Chodorow, MongoDB: The Definitive Guide, 4th ed. Sebastopol, CA: O'Reilly Media, 2024.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in Advances in Neural Information Processing Systems, pp. 5998–6008, 2017.

[7] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” Journal of Artificial Intelligence Research, vol. 78, pp. 345–382, 2024.

[8] R. Shridhar and V. Kumar, “Vector Databases: A New Era for Retrieval-Augmented Generation,” IEEE Transactions on Knowledge and Data Engineering, vol. 38, no. 4, pp. 1024–1038, 2025.

[9] DeepSeek-AI, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” Technical Report, 2025.

[10] S. Esmaeilzadeh, A. Talebpour, and M. Dorri Nogoorani, “Zero-Shot Academic Assistant: Grounding LLMs in Institutional Documentation,” in Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 154–168, 2025.

[11] H. Lang, M. Chen, and R. Xu, “Evaluation of Retrieval-Augmented Generation (RAG) in Closed-Domain Academic Systems,” in Proceedings of the 2025 International Conference on Artificial Intelligence in Education (AIED), vol. 12, pp. 45–62, 2025.