2026 Volume: 02 Issue: 08 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-22

COLLEGE INFORMATION CHATBOT USING RETRIEVAL-AUGMENTED GENERATION (RAG) ARCHITECTURE: A BEGINNER-FRIENDLY AND COST-EFFECTIVE APPROACH FOR PERSONALIZED EDUCATIONAL ASSISTANCE

Suryavhi Das*[1], Dr. Supantha Mandal[2], Tathagata Roy Chowdhury[3]

[1]B. Tech Students, Department of Computer Science and Engineering (AI & ML), Techno Institute of Engineering and Management, West Bengal, India.

[2]Assistant Professor, Department of Computer Science and Engineering (AI & ML), Techno Institute of Engineering and Management, West Bengal, India.

[3]Assistant Professor, Department of Computer Science and Engineering, Techno Institute of Engineering and Management, West Bengal, India.

Article Received: 09 July 2026
Article Revised: 29 July 2026
Published on: 17 August 2026

***Corresponding Author: Suryavhi Das**
B. Tech Students, Department of Computer Science and Engineering (AI & ML), Techno Institute of Engineering and Management, West Bengal, India.

DOI: https://doi-doi.org/101555/ijrpa.3790

ABSTRACT

Institutional information at academic colleges is typically scattered across static websites, printed brochures, and manually maintained notice boards, forcing students, applicants, and staff to search multiple sources for a single answer. General-purpose Large Language Models (LLMs) can converse fluently but frequently hallucinate when asked about localised, institution-specific facts such as fee structures or placement records, since such details lie outside their training distribution. This paper presents the design, implementation, and evaluation of an AI-powered College Information Chatbot for Techno Institute of Engineering and Management (TIEM), built on the Retrieval-Augmented Generation (RAG) paradigm. A curated institutional knowledge base covering admissions, academic programmes, fee structures, hostel and library facilities, placement statistics, and campus infrastructure is split into overlapping passages using a recursive character-level splitter (chunk size 500, overlap 50) and encoded into 384-dimensional dense vectors with the HuggingFace sentence-transformer model all-MiniLM-L6-v2. These vectors are persisted in a ChromaDB store indexed with Hierarchical Navigable Small World (HNSW) graphs, enabling sub-linear cosine-similarity retrieval. At query time, the top-k most relevant chunks

are injected into a structured prompt and forwarded to a Qwen3-32B model served through the GROQ inference API, which returns a fluent, context-grounded answer that is displayed through a Streamlit-based conversational interface named miro.ai. Experimental evaluation across diverse query categories — admissions, fees, faculty, placements, and facilities — shows that the RAG-grounded responses consistently reproduce verified institutional facts and remain robust to lexical variation in how a question is phrased, while a comparative study against standard prompting and fine-tuning approaches shows RAG offers the best balance of factual accuracy, update cost, and deployment simplicity for a small institution. The resulting system is lightweight, reproducible with open-source and free-tier tools, and readily extensible to voice, multilingual, and authenticated-portal use cases. The work demonstrates that a small academic team can build a production-quality, hallucination-resistant conversational assistant without large computational budgets, and it contributes a practical reference architecture for similar institution-scale deployments.

KEYWORDS: Retrieval-Augmented Generation, Artificial Intelligence, LangChain, Vector Database, ChromaDB, GROQ LLM, Semantic Search, College Chatbot, Educational AI, Sentence Transformers.

I. INTRODUCTION

Higher-education institutions operate within an information ecosystem that is continuously queried by prospective students, enrolled learners, parents, faculty, and administrative staff. The questions asked are diverse in wording but repetitive in substance: what programmes are offered, what the tuition and hostel fees are, how admission works, what the placement record looks like, and what facilities the campus provides. Traditionally this information has been disseminated through static websites, printed brochures, notice boards, and human help desks. These channels remain necessary, but they share a common weakness — they are not conversational. A visitor must know where to look, must read past irrelevant material to find the specific fact they need, and must often wait for a staff member to be free before a nuanced question can be answered.

The rapid maturation of Natural Language Processing (NLP) and Large Language Models (LLMs) has opened an alternative path. Chatbots built on modern LLMs can interpret a question phrased in ordinary language, regardless of its exact wording, and generate a fluent response without being constrained to a fixed decision tree of intents. This capability is a significant advance over the rule-based and keyword-triggered chatbots that dominated

earlier deployments, which broke down whenever a user's phrasing fell outside the anticipated patterns. Conversational AI has already been adopted successfully in banking, healthcare, e-commerce, and customer support, and higher education is a natural next domain: student queries are high in volume, repetitive in nature, and time-sensitive, which makes them well suited to automation.

The obstacle that prevents a general-purpose LLM from being deployed directly as an institutional assistant is hallucination — the tendency of a language model to produce answers that are grammatically fluent and superficially convincing but factually wrong or entirely fabricated. Because an LLM's knowledge is encoded implicitly in billions of trained parameters rather than retrieved from a verifiable source, it cannot distinguish between something it truly knows and something it is merely completing plausibly. For an institution, an incorrect answer about a fee amount, an admission deadline, or a placement statistic is not a harmless inconvenience; it can mislead an applicant's decision and damage the institution's credibility. This risk is amplified for small, less-documented institutions such as Techno Institute of Engineering and Management (TIEM), whose specific data was never part of any foundation model's training corpus and can therefore only be fabricated, not recalled, by a standalone LLM.

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [1], addresses this weakness by decoupling factual knowledge from the generative model. Rather than relying purely on parametric memory, a RAG system first retrieves the passages most relevant to a query from an external, authoritative knowledge base, and then conditions the language model's generation on that retrieved context. This architecture has two practical consequences that make it attractive for an institutional deployment. First, factual accuracy improves because the model is instructed to answer from evidence rather than from memory. Second, and just as important for a resource-constrained team, the knowledge base can be corrected or extended at any time by editing plain documents — no retraining or fine-tuning of the underlying model is required. When TIEM revises its fee structure or publishes a new placement record, the chatbot's answers can be updated within minutes simply by re-indexing the affected document.

Motivated by these considerations, this paper presents an AI-Powered College Information Chatbot for TIEM, deployed under the interface name miro.ai, built entirely on RAG principles. The knowledge base consists of curated institutional documents covering courses, admissions, fees, hostel and library facilities, placement statistics, faculty expertise, and campus rules. Documents are split into overlapping chunks with a recursive character-level

splitter, embedded into a shared 384-dimensional vector space with the HuggingFace sentence-transformer model all-MiniLM-L6-v2, and indexed persistently in ChromaDB using an HNSW approximate-nearest-neighbour graph. At inference time the user's query is embedded into the same space, the most semantically similar chunks are retrieved by cosine similarity, and the retrieved evidence is injected into a prompt that is sent to a Qwen3-32B model served through the GROQ API — chosen for its very low inference latency relative to conventional cloud LLM endpoints. The generated answer is cleaned of any residual internal reasoning tokens and rendered through a Streamlit conversational interface with persistent session state.

The remainder of this paper is organised as follows. Section II reviews prior work on conversational AI and retrieval-augmented systems and identifies the gaps this project addresses. Section III states the problem more precisely, and Section IV lists the project's objectives. Section V details the proposed methodology, including data preparation, the ingestion pipeline, chunking strategy, embedding generation, vector storage, retrieval, prompt engineering, and generation. Section VI describes the overall system architecture, and Section VII lists the algorithms used. Section VIII reports experimental results and a comparative discussion, followed by Sections IX–XI, which discuss the project's innovation, advantages, and limitations. Section XII outlines future scope, and Section XIII concludes the paper.

II. Literature Review

A. Evolution of Conversational AI

Conversational systems have progressed through four broad generations. Early agents such as ELIZA [4] and A.L.I.C.E. [5] relied on hand-written pattern-matching rules; they were transparent and predictable but collapsed on paraphrased or novel input and required constant manual maintenance. Retrieval-based systems that followed used lexical techniques such as TF-IDF [6] and BM25 [7] ranking to select a response from a fixed database, improving scalability but suffering from vocabulary mismatch — a semantically identical question phrased differently could fail to retrieve the correct stored answer. The introduction of the Transformer architecture by Vaswani et al. [2] and large pre-trained generative models such as BERT [3], GPT-2 [8], GPT-3, and T5 [9] enabled open-ended, fluent generation without predefined templates, but these purely parametric models remain vulnerable to hallucination and to knowledge that is stale beyond their training cut-off [10].

B. Retrieval-Augmented Generation

Lewis et al. [1] formalised Retrieval-Augmented Generation as a hybrid architecture that couples a dense retriever with a sequence-to-sequence generator, allowing the model's output to be grounded in passages fetched from an external corpus at inference time. This separates what the model has memorised from what it can look up, which is precisely the property an institutional chatbot needs: the language model supplies fluency and reasoning, while the retrieved documents supply facts that can be corrected independently of the model's weights. Subsequent architectures extended this idea in different directions — REALM [11] integrates the retriever into pre-training itself, Fusion-in-Decoder [12] lets the generator attend jointly over multiple retrieved passages, and RETRO [13] scales retrieval to trillion-token corpora. Sentence-BERT [14] and related sentence-embedding models made dense retrieval practical at low computational cost by producing semantically meaningful fixed-length vectors suitable for approximate nearest-neighbour search, and FAISS [15] demonstrated that such search can be performed efficiently even over billions of vectors, a lineage that ChromaDB's HNSW indexing follows at a smaller institutional scale.

C. Comparative Review of Key Building Blocks

Table I summarises the components most directly relevant to this project, evaluated on purpose, method, advantages, and the limitation or gap each one leaves for a system such as ours to address.

Table I. Comparative summary of foundational works and tools underpinning the proposed system.

Work / Tool	Purpose	Method	Advantage	Limitation / Gap
Lewis et al., RAG [1]	Ground LLM output in external knowledge	Dense retriever + seq2seq generator, jointly conditioned	Reduces hallucination, source-updatable knowledge	Original work targets open-domain QA benchmarks, not a deployed institutional assistant
Vaswani et al., Transformer [2]	Sequence modelling backbone for NLP	Self-attention, no recurrence	Parallelisable, scales to large corpora	No retrieval or grounding mechanism by itself
LangChain framework	Orchestrate LLM application components	Composable loaders, splitters, retrievers, chains	Reduces boilerplate, modular pipeline	Frequent breaking API changes across versions
ChromaDB	Persistent vector storage	HNSW approximate	Fast similarity search, simple	Returns L2 distance, requires manual

	and search	nearest-neighbour index	Python API	similarity conversion
Sentence-Transformers / MiniLM [14]	Generate semantic sentence embeddings	Siamese BERT fine-tuned with contrastive objective	Compact (384-dim), fast, runs locally	Lower ceiling on nuance versus larger embedding models
Enterprise / customer-support chatbots	Automate high-volume repetitive queries	Intent classification or RAG over product docs	Proven at scale in industry	Rarely adapted for small-institution academic data or free-tier budgets
General educational AI assistants	Support learning and campus queries	Mix of rule-based FAQ bots and LLM wrappers	Improves accessibility, 24/7 availability	Frequently lack grounding, leading to fabricated institutional facts

D. Research Gaps

Three gaps recur across the surveyed literature and motivate this project. First, domain specificity: most published RAG and chatbot systems target either open-domain benchmarks or large enterprise deployments, and smaller academic institutions rarely have the technical resources to adapt this research into a working, institution-specific assistant. Second, cost and complexity: many demonstrated architectures assume access to GPU infrastructure or paid, high-latency commercial APIs, which is impractical for a student project or a small institution's operating budget; this work instead relies on a locally run embedding model and a free-tier, low-latency inference API. Third, evaluation realism: much of the RAG literature reports benchmark scores on curated question sets rather than on the messy, paraphrase-heavy queries real users type; this project instead evaluates the system against naturally varied phrasings of the same institutional questions, which is a more faithful test of the semantic robustness a deployed assistant actually needs.

III. Problem Statement

Techno Institute of Engineering and Management manages a large volume of information spanning academic programmes, fee structures, hostel facilities, placement records, campus infrastructure, and institutional rules. In practice this information is distributed across heterogeneous sources — the official website, PDF brochures, administrative circulars, and verbal explanations from staff — and this fragmentation creates five concrete difficulties. Information fragmentation forces a visitor to search multiple documents for a single answer. Static FAQ pages and web content are updated infrequently and can present outdated or contradictory facts, which erodes trust. A general-purpose LLM, lacking institution-specific grounding, is prone to producing plausible-sounding but incorrect answers when asked about

TIEM's specific fees, faculty, or placement figures. Administrative staff spend a disproportionate share of their time answering the same repetitive questions rather than higher-value work. Finally, non-technical users — particularly first-time applicants — may struggle to navigate a complex institutional website or formulate an effective search-engine query. The system proposed in this paper addresses all five difficulties through a single conversational interface that retrieves answers exclusively from a verified, centrally maintained institutional knowledge base.

IV. Objectives

The project is guided by the following academic, technical, research, and innovation objectives.

1. Develop an AI-powered conversational chatbot capable of resolving institution-specific queries covering admissions, academic programmes, campus facilities, and administrative information in natural language.
2. Design and implement an end-to-end Retrieval-Augmented Generation pipeline comprising document processing, recursive text splitting, embedding generation, persistent vector storage, and similarity-based retrieval.
3. Integrate a Large Language Model, accessed through the GROQ inference API, to interpret retrieved context and generate fluent, accurate, and context-grounded responses.
4. Build a modular and maintainable RAG workflow using the LangChain framework, so that document loaders, embeddings, vector stores, and LLM calls can be modified independently.
5. Design and deploy an interactive, user-friendly chatbot interface using Streamlit that hides the complexity of the underlying AI pipeline behind a simple question-and-answer experience.
6. Evaluate the system's retrieval accuracy, response quality, and resistance to hallucination relative to standard prompting and fine-tuning alternatives, in order to quantify the practical benefit of the RAG approach for institutional information systems.

V. PROPOSED METHODOLOGY

This section describes every stage of the pipeline, from raw institutional documents to a displayed chat response. The methodology is organised into ten subsections that mirror the actual data flow of the implemented system: data collection, preprocessing, the ingestion pipeline, text chunking, embedding generation, vector database storage, the retrieval pipeline,

prompt engineering, the generation pipeline, and the user interface.

A. Data Collection

The knowledge base was built by curating institution-specific information directly from official TIEM sources and verifying it against the college's published records. The compiled corpus covers course offerings across Computer Science and Engineering (with specialisations in Artificial Intelligence and Machine Learning), Electronics, and Management streams; the admission procedure; the tuition fee structure (B.Tech: INR 1,10,000; M.Tech: INR 90,000; MBA: INR 1,40,000) and hostel fee (INR 55,000); library holdings of more than 40,000 books with weekday hours of 9 a.m. to 6 p.m. and a four-book borrowing limit; placement statistics (average package INR 4 LPA, highest package INR 10 LPA) together with recruiter names such as TCS, Infosys, Wipro, and Cognizant; faculty expertise across the Computer Science, Electronics, and Management departments; and campus facilities, transportation, annual events, and institutional rules. The material was compiled into a structured plain-text corpus and a supplementary PDF to ensure the retriever always has an authoritative, single-source answer for each fact category.

B. Data Preprocessing

Before indexing, the raw compiled text was cleaned to remove inconsistent formatting, duplicated headings, stray whitespace, and non-informative boilerplate that would otherwise dilute the semantic signal of a chunk. Each fact category (courses, fees, hostel, placements, faculty, events, contact information) was organised under a clear section heading so that the recursive splitter described in Section V-D would tend to keep semantically related sentences together. This preprocessing step is intentionally lightweight — the emphasis of the RAG design is on correct retrieval rather than on heavy natural-language cleaning — but it materially improves chunk coherence, which in turn improves retrieval precision.

C. Ingestion Pipeline

The ingestion pipeline is the offline half of the system: it converts static institutional documents into a queryable vector index exactly once, and the resulting index is then reused for every subsequent conversation. Fig. 2 illustrates both the ingestion pipeline (top row) and the corresponding online query pipeline (bottom row) that reuses the same vector store. Documents are loaded through LangChain's TextLoader and PDFLoader classes

into Document objects that preserve source metadata, passed through the recursive character text splitter, encoded into dense vectors by the embedding model, and finally written to the persistent ChromaDB collection. Because this pipeline is decoupled from query-time inference, re-indexing an updated fee sheet or an updated placement report takes only the time required to re-run these four steps, with no retraining of the language model involved.

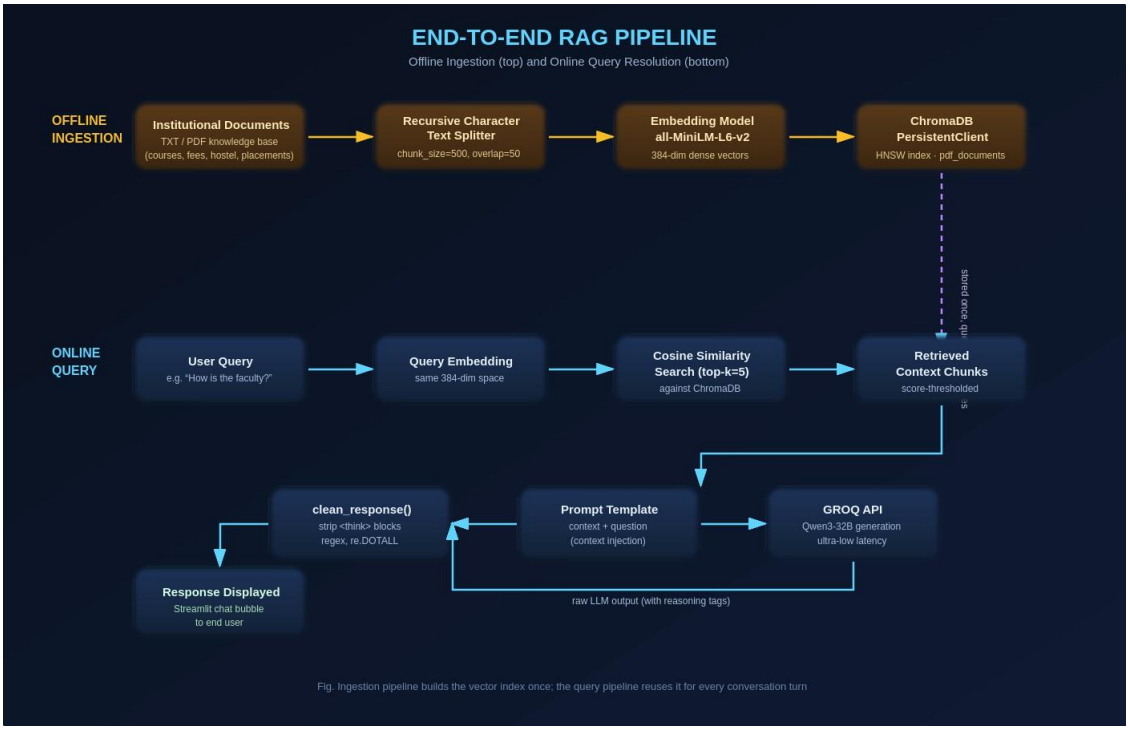

Fig. 2. Offline ingestion pipeline (top) and online query-resolution pipeline (bottom) sharing a single ChromaDB vector store.

D. Text Chunking

Embedding models and LLM context windows both have finite length, and a single institutional document is far too long to embed as one unit without losing retrieval precision. The RecursiveCharacterTextSplitter from LangChain is used to divide each document into shorter, semantically coherent chunks before embedding. This splitter is hierarchical: it first attempts to break text at paragraph boundaries, then at sentence boundaries, then at word boundaries, and only falls back to raw character-level splitting when no natural boundary is available within the target length. The configured parameters are a chunk size of 500 characters and a chunk overlap of 50 characters. The overlap ensures that a sentence spanning a chunk boundary still appears completely in at least one chunk, which prevents the retriever from returning a truncated fact. Smaller chunks were found to fragment context (a retrieved passage answered only part of a question), while larger chunks diluted relevance (a passage

matched the query only loosely, in among unrelated sentences); 500/50 was selected empirically as the balance that gave the most consistently useful retrieved context across the test query set.

E. Embedding Generation

Each chunk is transformed into a dense vector representation using the HuggingFace sentence-transformer model all-MiniLM-L6-v2, a distilled variant of larger BERT-family encoders trained with a contrastive objective on more than one billion sentence pairs. The model outputs 384-dimensional, L2-normalised embeddings, which is a favourable trade-off between semantic fidelity and inference speed for a system that must embed both the entire knowledge base offline and every incoming query online, on modest hardware. Mathematically, the model maps a text span t to a vector $e(t) \in \mathbb{R}^{384}$ such that spans with related meaning are mapped close together under cosine distance, even when they share very little vocabulary — this is what allows the retriever to match a query phrased as “How do I get admitted?” to a chunk headed “Admission Procedure” without any literal keyword overlap between the two.

F. Vector Database

ChromaDB was chosen as the vector store because it is open-source, embedding-first, and offers a persistent client mode that survives application restarts without needing the embeddings to be regenerated on every run. A dedicated collection, pdf_documents, stores each chunk's unique identifier (assigned with Python's uuid4), its raw text, its embedding vector, and metadata such as source file path, chunk index, and content length for traceability. Internally, ChromaDB indexes the collection with a Hierarchical Navigable Small World (HNSW) graph, which gives approximate nearest-neighbour queries logarithmic time complexity with respect to the number of stored vectors, so retrieval remains fast even as the knowledge base grows well beyond its current size.

G. Retrieval Pipeline

At query time, the user's question is embedded with the same all-MiniLM-L6-v2 model used during ingestion, guaranteeing that query and document vectors live in the same semantic space. The query embedding is passed to ChromaDB's query method, which performs an approximate nearest-neighbour search and returns the top-k (default k = 5) most similar chunks. ChromaDB natively reports L2 (Euclidean) distance rather than cosine similarity, so the system converts each result using

$$similarity = 1 - distance \quad (1)$$

which holds because the sentence-transformer model produces L2-normalised embeddings. A configurable score_threshold then discards chunks whose similarity falls below an acceptable relevance level, so that a question with no good match in the knowledge base does not force irrelevant text into the prompt. This retrieval stage is the mechanism by which the system achieves semantic robustness: differently worded questions that share the same underlying intent, for example “What are the courses available?” and “Which programmes does the college offer?”, converge on nearly identical embeddings and therefore retrieve the same relevant passage.

H. Prompt Engineering

The retrieved chunks are concatenated into a single context string and embedded inside a structured prompt template that explicitly instructs the LLM to answer using only the supplied context and the user's original question. This context-injection pattern — context, then question, then an instruction to answer strictly from the given material — is the core mechanism by which RAG constrains generation and suppresses hallucination: the model is not asked to recall a fact from its parameters, but to summarise and phrase a fact that is already present in front of it. Where no sufficiently relevant chunk is retrieved, the prompt design allows the model to state that the information is not available in the knowledge base rather than inventing an answer.

I. Generation Pipeline

The augmented prompt is sent to a Qwen3-32B model served through the GROQ inference API. GROQ was selected over conventional cloud LLM endpoints because of its markedly lower inference latency, which keeps the conversational interface feeling responsive even though every turn involves an embedding call, a vector search, and a full generation call. The raw model output occasionally contains internal chain-of-thought reasoning enclosed in reasoning tags, an artefact of the model's extended-thinking behaviour; a clean_response function using a compiled regular expression with the DOTALL flag strips these blocks before the text reaches the user, so that only the final answer is ever displayed.

J. User Interface

The interface, branded miro.ai — “Your Personalized College Assistant” — is implemented entirely in Streamlit. It uses st.chat_input to capture queries and st.chat_message to render

the ongoing dialogue, with custom avatars and a dark navy theme (background #0f172a, rounded 15 px chat bubbles) that gives the assistant a distinct, professional identity rather than the default framework look. Conversation history persists across turns through Streamlit's session state, so the interface re-renders the full dialogue on every interaction. Fig. 3 shows the landing screen a user sees on opening the assistant.

Fig. 3. Front interface of miro.ai — the landing screen presented before the first query is asked.

VI. System Architecture

The complete system is organised as a modular, five-layer architecture that cleanly separates the concerns of user interaction, query embedding, retrieval, generation, and knowledge storage, as shown in Fig. 4. This separation means each layer can be modified, scaled, or replaced independently: the interface layer could be swapped from Streamlit to a mobile client, the embedding model could be upgraded without touching the retrieval logic, and the GROQ-hosted LLM could be substituted for another provider, all without redesigning the rest of the pipeline.

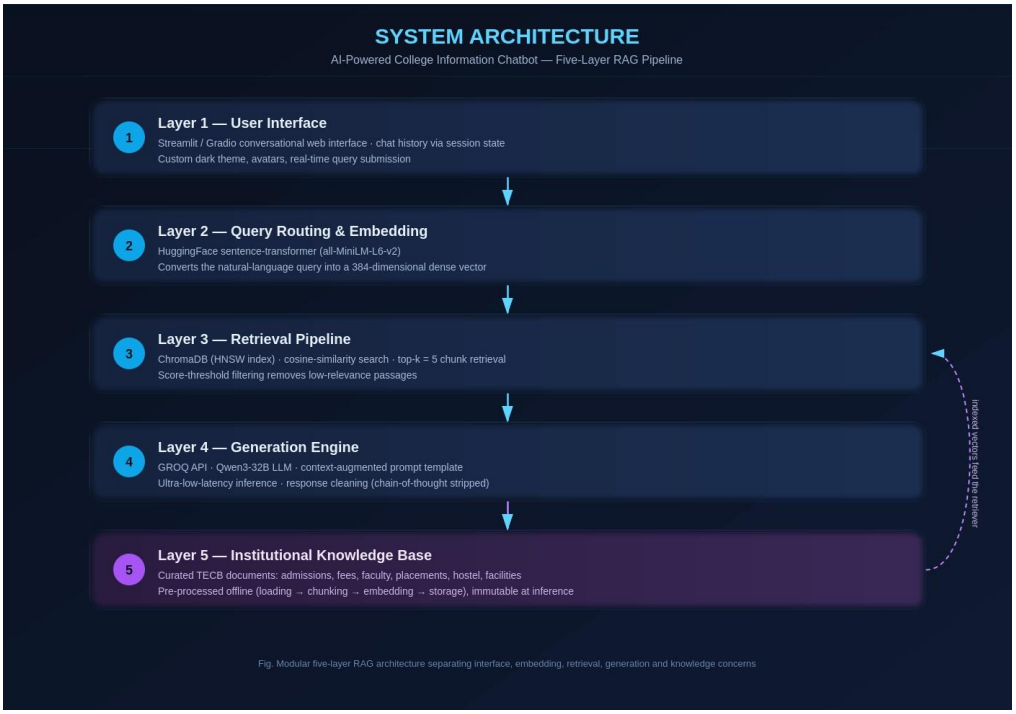

Fig. 4. Five-layer system architecture of the AI-powered college information chatbot

The five layers, and the responsibility of each, are as follows.

- Layer 1 — User Interface: a Streamlit-based conversational web interface that accepts natural-language queries and renders responses with persistent session history.
- Layer 2 — Query Routing and Embedding: converts the incoming query into a 384-dimensional dense vector with the all-MiniLM-L6-v2 sentence-transformer model.
- Layer 3 — Retrieval Pipeline: performs an HNSW-indexed cosine-similarity search over ChromaDB to fetch the top-k most relevant knowledge-base chunks.
- Layer 4 — Generation Engine: forwards a context-augmented prompt to the GROQ-hosted Qwen3-32B model and cleans the returned text before display.
- Layer 5 — Knowledge Base: the curated, offline-indexed corpus of institutional documents that grounds every answer the system produces.

Fig. 5 makes the same architecture concrete at the message level, tracing a single conversational turn as it passes between the user, the Streamlit interface, the embedding model, ChromaDB, and the GROQ-hosted LLM. This communication view highlights that, of the eleven message exchanges required to answer one question, only the first and the last are visible to the user — every intermediate step (embedding, similarity search, prompt construction, and generation) happens within roughly a second, which is what gives the assistant its conversational feel despite the multi-stage pipeline behind it.

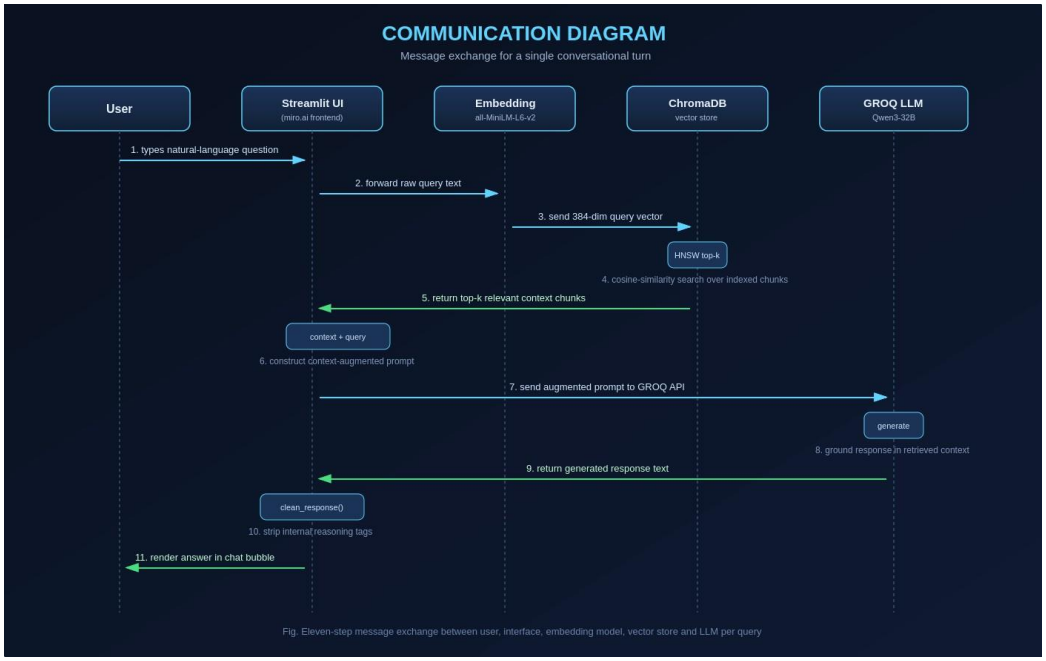

Fig. 5. Communication diagram tracing the message exchange for a single query-response turn.

VII. Algorithms Used

Table II summarises the core algorithms and computational techniques applied at each stage of the pipeline.

Algorithm / Technique	Role in the Pipeline
Recursive Character Text Splitting	Divides documents into overlapping, semantically coherent chunks using a hierarchical separator strategy (paragraph → sentence → word → character).
Dense Embedding (all-MiniLM-L6-v2)	Maps text spans to 384-dimensional, L2-normalised vectors using a distilled Siamese BERT encoder trained with a contrastive objective.
HNSW Approximate Nearest-Neighbour Search	Indexes embeddings in a navigable small-world graph for logarithmic-time similarity search inside ChromaDB.
Cosine Similarity Scoring	Ranks retrieved chunks by similarity = 1 − L2 distance, valid for normalised embeddings, with a configurable relevance threshold.
Retrieval-Augmented Generation	Injects the top-k retrieved chunks into a structured prompt so that the LLM conditions generation on retrieved evidence rather than parametric memory alone.
Prompt Engineering / Context Injection	Structures the prompt as context, then question, then an instruction to answer only from the supplied context.
Regex-based Response Cleaning	Uses a compiled pattern with the DOTALL flag to strip internal chain-of-thought reasoning tags from the raw LLM output before display.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

A. Functional Outcomes

The complete pipeline was exercised against a test suite of questions spanning every knowledge-base category. The system consistently reproduced precise institutional facts, including the exact fee figures (B.Tech INR 1,10,000; MBA INR 1,40,000; M.Tech INR 90,000; hostel INR 55,000), placement statistics (average INR 4 LPA, highest INR 10 LPA) with named recruiters such as TCS, Infosys, Wipro, and Cognizant, library specifications (40,000+ books, 9 a.m.–6 p.m. weekday hours, four-book borrowing limit), and department-wise faculty expertise, without introducing fabricated figures. Fig. 6 shows a representative conversational turn in which the assistant is asked an open-ended question about faculty quality and responds with department-specific, factually grounded detail retrieved from the knowledge base rather than a generic, invented answer.

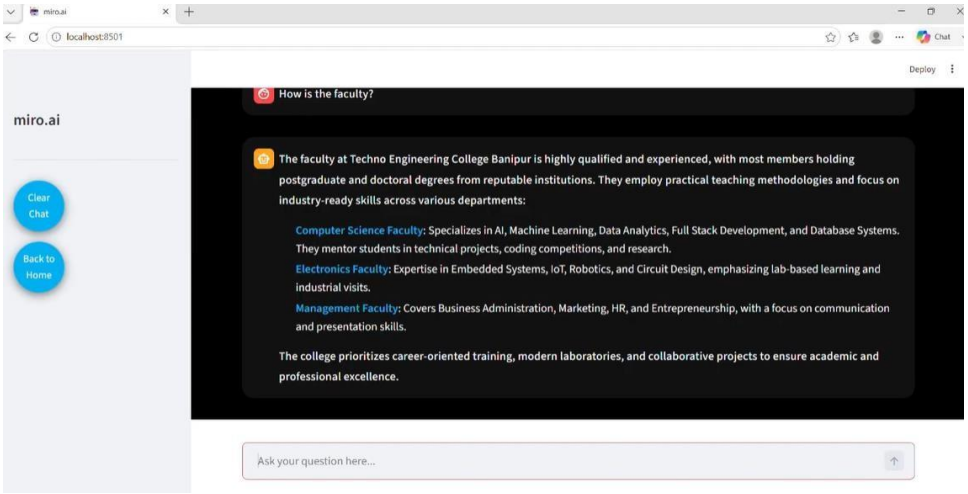

Fig. 6. Sample conversation: miro.ai answering a faculty-related query with retrieval-grounded, department-wise detail.

B. Semantic Query Understanding

Because retrieval operates in embedding space rather than on literal keyword overlap, the system correctly matched questions phrased in markedly different ways to the same underlying knowledge-base section. Queries such as “What are the courses available at TECB?”, “Which programmes does the college offer?”, and “How do I get admitted to the college?” all retrieved the relevant Courses and Admission passages despite sharing little vocabulary, confirming that the embedding-based retriever generalises across paraphrase in a way a keyword-matching system could not.

C. Comparative Study

Table III compares the RAG architecture used in this project against two alternative approaches — standard prompting without retrieval, and fine-tuning an LLM on domain data — across the dimensions most relevant to a small institution's deployment constraints.

Table III. Comparative evaluation of RAG against standard prompting and fine-tuning.

Dimension	RAG (this project)	Standard Prompting	Fine-Tuning on Domain Data
Factual Accuracy	High (grounded in knowledge base)	Moderate (prone to hallucination)	High (post-training)
Hallucination Rate	Low	High	Low
Knowledge Updateability	Real-time (edit knowledge base)	None (retraining required)	Low (retraining required)
Computational Cost	Low (embedding + retrieval)	Very low (inference only)	Very high (GPU training)
Domain Specificity	High (localised knowledge base)	Low	High (after tuning)
Deployment Complexity	Moderate	Low	High
Data Privacy	High (local knowledge base)	Low (data sent to API)	Moderate
Scalability	High (add to knowledge base)	Low	Low

The comparison shows that RAG's decisive advantage for this use case is knowledge updateability: as fees change, new courses are added, or placement figures are revised, the knowledge base can be re-indexed in minutes with no model retraining, whereas fine-tuning would require GPU-accelerated retraining for every such change. RAG also provides substantially higher factual accuracy than prompting without retrieval, since a general-purpose model has no path to TIEM-specific facts other than fabrication.

D. Challenges Faced

- LangChain, langchain-community, langchain-groq, chromadb, and opentelemetry package versions conflicted repeatedly during development and required careful, iterative dependency pinning.
- Chunk size and overlap needed empirical tuning: oversized chunks diluted retrieval precision with irrelevant content, while undersized chunks fragmented answers across multiple retrieved passages.
- ChromaDB reports L2 distance rather than cosine similarity, requiring verification that the embedding model's vectors are normalised before the similarity = 1 − distance conversion

could be trusted.

- The Qwen3-32B model's internal chain-of-thought reasoning had to be reliably stripped from the displayed answer without deleting legitimate response content, which required careful regex testing.
- Retrieving several highly similar chunks could approach the LLM's context window; a top-k cap combined with the similarity threshold keeps the augmented prompt within safe limits.

IX. Innovation

The contribution of this work is not a new algorithm but a complete, working demonstration of how a small academic team can assemble open-source and free-tier components into a reliable, production-style RAG system. Five aspects distinguish the implementation: a fully modular pipeline in which every stage — loading, splitting, embedding, storage, retrieval, prompting, and generation — is independently replaceable through LangChain's abstractions; a beginner-friendly design that keeps each component legible enough for a student team to build, debug, and extend without prior production ML-engineering experience; a cost-effective stack that uses a locally run open-source embedding model together with GROQ's free-tier, low-latency inference rather than any paid enterprise platform; personalised, institution-specific retrieval that grounds every answer in TIEM's own verified documents rather than generic web knowledge; and easy extensibility, since the same pipeline can absorb a new domain of documents (a different department, a different institution) without any retraining of the underlying language model. This section deliberately frames these as engineering and educational contributions rather than claims of algorithmic novelty beyond what has already been established in the RAG literature.

X. Advantages

1. Grounded, verifiable answers reduce hallucination compared with a standalone LLM.
2. The knowledge base can be updated by editing plain documents, with no model retraining required.
3. Local embedding generation keeps institutional data private, since only the final augmented prompt is sent to the external LLM API.
4. GROQ's low-latency inference keeps the conversational experience responsive despite the multi-stage retrieval pipeline.
5. Semantic retrieval generalises across paraphrased queries, unlike keyword-based FAQ

search.

6. The modular LangChain-based pipeline is easy to extend to new document types or knowledge domains.
7. The entire stack relies on open-source or free-tier tools, keeping deployment cost near zero for a small institution.
8. A clean, dedicated Streamlit interface lowers the barrier for non-technical users compared with navigating a full institutional website.
9. Automating repetitive queries reduces the workload on administrative staff.
10. The architecture is transparent and auditable: every answer can be traced back to the specific document chunk that produced it.

XI. Limitations

- Answer quality is entirely dependent on the completeness and accuracy of the curated knowledge base; an unindexed fact cannot be answered correctly.
- The system requires an active internet connection to reach the GROQ inference API, so it cannot operate fully offline.
- Reliability is tied to the availability and rate limits of a third-party API provider.
- The current implementation has no persistent conversational memory; each query is processed independently of prior turns.
- There is no voice interface, which limits accessibility for visually impaired users.
- The knowledge base and interface currently support English only, excluding users more comfortable in Bengali or Hindi.

XII. Future Scope

The present implementation is a functional first version, and several directions can extend it toward a fuller institutional AI assistant. On the interface side, the current Streamlit front end can be rebuilt as a richer, custom-branded web application using HTML, CSS, and JavaScript, giving finer control over layout, animation, and mobile responsiveness than a framework-default theme allows. On the model side, the pipeline's LLM layer is intentionally decoupled from the retrieval layer, so it can be pointed at alternative providers — Gemini, GPT, or open-weight Llama models — allowing a direct comparison of factual accuracy, latency, and cost without touching the retrieval logic at all.

Several capability extensions are also planned. A speech-to-text front end (for example, OpenAI Whisper) paired with a text-to-speech engine would turn the assistant into a fully

voice-enabled interface, improving accessibility for visually impaired users and for those who simply prefer speaking over typing. A language-detection and translation layer, combined with a multilingual embedding model such as paraphrase-multilingual-MiniLM-L12-v2, would let the same knowledge base serve Bengali- and Hindi-speaking users without duplicating content. Integrating the chatbot with TIEM's live Student Information System would allow authenticated, personalised answers about an individual's attendance, fee status, or academic record, rather than only general institutional facts. Cloud deployment on a platform such as AWS, Google Cloud, or Azure, containerised with Docker and orchestrated with Kubernetes, would allow the system to serve many concurrent users reliably, and migrating the vector store to a managed cloud vector database would improve backup and availability at scale.

On the retrieval side, adaptive top-k selection based on query complexity, cross-encoder re-ranking of the initially retrieved chunks, and multi-hop retrieval that can combine evidence from more than one knowledge-base section would together improve answer quality for compound questions. Finally, adding a short-term conversational memory module would let the assistant resolve follow-up questions such as “and what about the hostel fee?” in the context of a preceding question about tuition, producing a more natural, continuous dialogue rather than treating every turn as an isolated query. Collectively, these directions would transform the current prototype into a more accessible, scalable, and genuinely personalised institutional AI assistant.

XIII. CONCLUSION

This paper presented the design, implementation, and evaluation of an AI-powered college information chatbot for Techno Institute of Engineering and Management, built on Retrieval-Augmented Generation. By combining a curated, verified institutional knowledge base with dense semantic embeddings, persistent HNSW-indexed vector storage in ChromaDB, and low-latency generation through a GROQ-hosted Qwen3-32B model, the system delivers answers that are both fluent and factually grounded — directly addressing the hallucination problem that makes a standalone LLM unsuitable for institution-specific information delivery. The evaluation showed that the system consistently reproduces verified facts about fees, placements, faculty, and facilities, generalises correctly across paraphrased questions through embedding-based semantic retrieval, and compares favourably against both plain prompting and fine-tuning on the dimensions that matter most to a small institution: factual accuracy, ease of updating, and cost of deployment.

Beyond its direct utility to TIEM's students, applicants, and staff, the project demonstrates something more general: that a small student-and-faculty team, without access to large computational budgets, can assemble open-source components — LangChain, sentence-transformers, ChromaDB, Streamlit — with a free-tier, low-latency inference API to build a system that behaves like a production-grade conversational assistant. In doing so, the project also delivered practical, end-to-end experience across the modern AI-application stack, from document processing and embedding to vector-database management, prompt design, and interactive interface development. The architecture is deliberately modular, and the future directions outlined in Section XII — voice interaction, multilingual support, authenticated personalisation, and cloud-scale deployment — extend naturally from the current design without requiring it to be rebuilt. The work therefore contributes both a working assistant for TIEM and a reusable reference architecture that other resource-constrained institutions can adapt to their own information-access needs.

XIV. ACKNOWLEDGEMENT

The authors gratefully acknowledge the guidance and support of Dr. Supantha Mandal, Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Techno Institute of Engineering and Management, whose supervision shaped the direction and rigour of this work. The authors also thank the Department of Computer Science and Engineering for providing the academic environment and infrastructure necessary to carry out this project.

XV. REFERENCES

1. P. Lewis, E. Perez, A. Piktus et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.
2. A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is All You Need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
3. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. NAACL-HLT, pp. 4171–4186, 2019.
4. 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.

5. R. Wallace, "The Elements of AIML Style," A.L.I.C.E. AI Foundation, 2001.
6. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.
7. S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.
8. A. Radford, J. Wu, R. Child et al., "Language Models are Unsupervised Multitask Learners," OpenAI Technical Report, 2019.
9. C. Raffel, N. Shazeer, A. Roberts et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1–67, 2020.
10. Z. Ji, N. Lee, R. Frieske et al., "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023.
11. K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," Proc. ICML, vol. 119, pp. 3929–3938, 2020.
12. G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," Proc. EACL, pp. 874–880, 2021.
13. S. Borgeaud, A. Mensch, J. Hoffmann et al., "Improving Language Models by Retrieving from Trillions of Tokens," Proc. ICML, vol. 162, pp. 2206–2240, 2022.
14. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proc. EMNLP, pp. 3982–3992, 2019.
15. J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.
16. T. B. Brown, B. Mann, N. Ryder et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.
17. A. Fan, C. Gardent, C. Braud, and A. Bordes, "Using Local Knowledge Graph Construction to Scale Seq2Seq Models for Complex Question Answering," Proc. EMNLP, 2019.
18. G. Izacard, P. Lewis, M. Lomeli et al., "Atlas: Few-shot Learning with Retrieval Augmented Language Models," Journal of Machine Learning Research, vol. 24, no. 251, pp. 1–43, 2023.
19. W. Shi, S. Min, M. Yasunaga et al., "REPLUG: Retrieval-Augmented Black-Box

Language Models," Proc. NAACL,
pp. 8371–8384, 2023.
20. Chroma Research Inc., "ChromaDB: The Open-Source Embedding Database," GitHub Repository, 2023. [Online].
Available: https://github.com/chroma-core/chroma
21. LangChain Inc., "LangChain: Building Applications with LLMs through Composability," GitHub Repository, 2023. [Online]. Available: https://github.com/langchain-ai/langchain
22. Groq Inc., "Groq API: Ultra-Low Latency AI Inference," Technical Documentation, 2024. [Online]. Available: https://groq.com/docs
23. HuggingFace, "Sentence-Transformers: Multilingual Sentence, Paragraph, and Image Embeddings," Technical Documentation, 2023. [Online]. Available: https://www.sbert.net