



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





AI Student Companion: Intelligent Chatbot Assistant for Students

G. Vijaya¹, V. Chandrika², O. Vamsi³, B. Jayanth⁴, D. Akshaj⁵, V. Venkata Naga Satish⁶

Assistant Professor, Department of CSE (AI & ML), Nadimpalli Satyanarayana Raju Institute of Technology,
Vishakapatnam, India ¹

Students, Department of CSE (AI & ML), Nadimpalli Satyanarayana Raju Institute of Technology,
Vishakapatnam, India^{2,3,4,5,6}

ABSTRACT: The AI Student Companion is an intelligent chatbot assistant developed to provide continuous and reliable academic support to students. The system integrates Natural Language Processing (NLP) and Retrieval-Augmented Generation (RAG) to retrieve relevant academic content from a structured vector knowledge base and generate accurate, syllabus-aligned responses. Unlike traditional rule-based chatbot systems, this solution combines FAISS-powered semantic search, transformer-based emotion detection, and the Gemini 2.5 Flash large language model (LLM) to deliver context-aware, factually grounded replies. A three-way intelligent routing mechanism classifies each user query into one of three pipelines: Sentiment (emotional support), RAG (document retrieval), or LLM (general knowledge). The system further incorporates an automated event announcement processor that extracts structured event information from uploaded posters and PDFs using Gemini Vision API and OCR. The backend is built on Flask and the frontend on React with TypeScript, authenticated via Firebase, and with persistent storage via Supabase. Comprehensive testing validates the system's reliability and performance across multiple testing dimensions.

KEYWORDS: Artificial Intelligence, Educational Chatbot, Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG), Large Language Model (LLM), FAISS Vector Database, Sentiment Analysis, Gemini API, Event Announcement Processing.

I. INTRODUCTION

The rapid evolution of Artificial Intelligence (AI) has profoundly transformed modern technological systems, particularly in domains requiring intelligent decision-making and automation [16]. Education has experienced significant digital transformation with the integration of smart learning platforms, virtual classrooms, and AI-driven tutoring systems. The growing demand for accessible, flexible, and personalized learning has accelerated the adoption of intelligent educational tools.

Traditional learning environments rely heavily on teacher-centered instruction, which cannot provide continuous academic support outside classroom hours. Students frequently encounter difficulty understanding complex concepts, have limited time for doubt clarification, and lack personalized feedback. While online resources are widely available, unverified information can lead to confusion and misinformation.

Recent breakthroughs in NLP, particularly the Transformer architecture introduced by Vaswani et al. [1] and pre-trained models such as BERT [2] and GPT-3 [3], have substantially improved language understanding. However, purely generative LLMs can produce hallucinated or factually incorrect responses — a critical concern in academic settings.

To address this, Retrieval-Augmented Generation (RAG) [5] was proposed as a hybrid approach combining information retrieval with generative models. RAG retrieves domain-specific documents before generating responses, improving factual accuracy and reducing hallucination. This paper presents the AI Student Companion, which implements a RAG-based architecture enriched with transformer emotion detection, automated event processing, and a full-stack deployment framework.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The key contributions of this work are:

- A three-way intelligent routing engine that dynamically routes queries to Sentiment, RAG, or LLM pipelines.
- Integration of DistilBERT and DistilRoBERTa for emotion-aware student support.
- Automated event announcement extraction from posters and PDFs using Gemini Vision API and Tesseract OCR.
- A full-stack React + Flask application with Firebase Authentication and Supabase storage.
- Modular knowledge base design allowing content updates without full model retraining.

II. LITERATURE SURVEY

A. Evolution of Educational Chatbots

Early chatbot systems were primarily rule-based and relied on predefined patterns and scripted responses, exemplified by ELIZA (1960s). These systems lacked contextual understanding and adaptability. Later systems incorporated machine learning to improve intent recognition but remained limited by static knowledge bases and absence of dynamic reasoning.

B. NLP and Transformer Models

Natural Language Processing (NLP) enables machines to understand and generate human language. Foundational techniques including tokenization, stemming, parsing, and semantic analysis are well-documented in standard references [15]. The introduction of the Transformer architecture [1] revolutionized NLP by enabling parallel sequence processing and improved contextual understanding through the self-attention mechanism. Large Language Models such as GPT-3 [3] and Google Gemini [4] demonstrated impressive performance in question answering, summarization, and conversational tasks. However, pure generative models are susceptible to hallucination, particularly problematic in educational applications.

C. Retrieval-Augmented Generation (RAG)

RAG was proposed by Lewis et al. [5] as a method combining neural retrieval and generation. RAG systems retrieve relevant documents from an external knowledge base and condition the language model on the retrieved context, resulting in improved factual accuracy, reduced hallucination, and efficient domain-specific knowledge integration [6]. FAISS [7], developed by Facebook AI Research, provides efficient similarity search over large vector collections and has become a foundational component in RAG pipelines.

D. Sentiment Analysis in Education

Sentiment and emotion detection improve chatbot responsiveness to student wellbeing. Abinaya et al. [9] proposed emotion-aware conversational agents capable of analyzing behavioral status to deliver tailored responses. Almutairi et al. [10] demonstrated machine learning pipelines for sentiment analysis achieving high accuracy on service-quality datasets. The availability of pre-trained transformer models such as DistilBERT (SST-2) and j-hartmann's DistilRoBERTa provides practical tools for fine-grained emotion classification.

E. Comparative Analysis of Chatbot Systems

System Type	Advantages	Limitations
Rule-Based Chatbots	Simple implementation, predictable	No contextual understanding, rigid
Pure LLM-Based Systems	Human-like, fluent responses	Hallucination, lacks factual grounding
RAG-Based Systems	Accurate, contextual, updatable	Requires structured knowledge base
Proposed System (This Work)	3-way routing, emotion detection, event processing	Dependent on API availability and latency

Table 1: Comparative Analysis of Chatbot System Types



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

F. Research Gap

Although numerous conversational AI systems exist, many educational chatbots lack integration of syllabus-specific knowledge bases, sentiment-aware routing, and automated event processing [11]. The proposed AI Student Companion bridges this gap by providing a unified, modular system designed specifically for institutional student support.

III. SYSTEM ANALYSIS

A. Existing System Limitations

Current academic support systems exhibit the following limitations:

- Traditional classroom instruction: limited availability outside class hours, large student-to-teacher ratio.
- Online learning platforms: static, non-interactive content with no real-time clarification.
- Search engines: information may not align with syllabus; risk of unreliable content.
- Pure LLM-based assistants: susceptibility to hallucination and lack of institutional knowledge grounding.

B. Proposed System

The proposed AI Student Companion overcomes these limitations through a Retrieval-Augmented Generation architecture enhanced with three-way query routing. Key capabilities include:

- Natural language query processing with context-aware response generation.
- Vector-based semantic similarity search over a structured knowledge base.
- Transformer-based emotion detection for personalized emotional support.
- Automated event announcement extraction from uploaded files.
- 24/7 availability through a web-based interface.
- Modular knowledge base enabling content updates without retraining.

C. Feasibility Study

Technical Feasibility: The system is technically feasible as NLP frameworks and transformer models are widely available. Vector databases such as FAISS support efficient similarity search. Cloud platforms and open-source libraries simplify implementation and deployment.

Economic Feasibility: Open-source tools including Python, Flask, FAISS, and HuggingFace reduce development cost. The Google Gemini API provides competitive pricing with generous free-tier access. No specialized hardware is required beyond standard computing resources.

Operational Feasibility: The system provides a simple, intuitive interface requiring no technical expertise from end users. Natural language interaction ensures easy adoption. The modular architecture supports straightforward integration with existing academic portals.

D. Requirement Analysis

Functional Requirements: Accept and process natural language queries; convert queries to embeddings; perform cosine similarity search via FAISS; retrieve relevant academic content; generate contextual responses; process uploaded documents and event posters; display structured announcements; maintain conversation history.

Non-Functional Requirements: High accuracy; low response latency; scalable architecture; secure user authentication; reliable uptime; maintainable codebase.

IV. SYSTEM DESIGN

A. System Architecture

The system is structured as a four-layer architecture consisting of the Frontend Layer (React + TypeScript chat interface with Firebase Google OAuth), API Gateway Layer (Flask REST API server), Intelligence Layer (3-Way Router dispatching to Sentiment, RAG, or LLM engines), and Data Layer (FAISS vector index, Firebase Cloud Storage, Supabase database, and local JSON events database).



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Fig. 1: NSRIT Campus AI Student Companion – Application Landing Page

B. Three-Way Query Routing Algorithm

The intelligent routing mechanism is the central algorithmic contribution of this system. Given a user query q , the router determines the appropriate pipeline through the following decision logic:

- Step 1 — Sentiment Detection: Keyword matching is applied first (e.g., "I feel", "I am sad", "overwhelmed"). If matched, the query is classified as SENTIMENT. Additionally, if transformer sentiment analysis is enabled, the DistilRoBERTa emotion model is invoked; if the confidence score ≥ 0.6 , the SENTIMENT route is confirmed.
- Step 2 — RAG Detection: If the query contains document-reference keywords (e.g., "according to", "summarize", "from the pdf") AND the FAISS index contains at least one vector ($n_{total} > 0$), the RAG route is selected.
- Step 3 — LLM Fallback: All remaining queries are routed to the LLM pipeline. A sub-classification within this route distinguishes greeting queries from general knowledge queries.

C. RAG Retrieval Pipeline

The RAG pipeline consists of two phases — indexing and retrieval — both implemented in the VectorDatabase and RAGEngine classes within `student_chatbot.py`.

Indexing Phase: When a document is uploaded, the system (1) loads the file using PyMuPDF for PDFs or PIL for images, (2) extracts text via Tesseract OCR or PDF text layer, (3) chunks the text into segments of 1,000 characters with 200-character overlap, (4) generates dense vector embeddings using the Gemini Embedding API [8] (model: gemini-embedding-001), (5) normalizes the embeddings using L2 normalization, and (6) stores the vectors in a FAISS IndexFlatL2 and persists the index to disk.

Retrieval Phase: When a RAG-routed query arrives, the system (1) embeds the query using the same Gemini Embedding API, (2) performs a top- $k=5$ cosine similarity search against the FAISS index, (3) filters results below the similarity threshold (0.25), (4) assembles the retrieved chunks into a context string, and (5) supplies this context as a prompt prefix to Gemini 2.5 Flash for grounded response generation.

D. Event Announcement Processing Pipeline

The announcement system provides automated extraction of structured event information from uploaded posters and PDF files. A folder watcher monitors the `event_photos_folder` directory at a configurable interval (default: 10 seconds). When new files are detected, each file is processed by `AnnouncementProcessor`. For PDFs, Gemini Vision API is applied to the first page; for images, the Vision API is invoked directly [13]. Tesseract OCR [14] serves as fallback if Vision fails, enabling robust text extraction from scanned or image-based documents [12]. Extracted data is persisted to `processed_events/events_database.json` and served through Flask `/api/announcements` endpoints.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

E. Emotion Detection Architecture

The TransformerSentimentAnalyzer class implements a two-stage emotion detection pipeline. The primary stage uses the j-hartmann/emotion-english-distilroberta-base model to classify queries into six emotion categories: sadness, joy, anger, fear, surprise, and neutral. When the primary model's confidence is below threshold, it falls back to the DistilBERT SST-2 model for binary positive/negative sentiment classification. Keyword-based detection serves as final fallback when transformer libraries are unavailable.

V. IMPLEMENTATION

A. Technology Stack

Layer	Technology	Purpose
AI / ML	Gemini 2.5 Flash, Gemini Embedding API	LLM generation and text embedding
Emotion AI	DistilBERT SST-2, DistilRoBERTa	Sentiment and emotion classification
Vector Search	FAISS (IndexFlatL2)	Semantic similarity retrieval
OCR & Vision	Tesseract OCR, PyMuPDF, pdfplumber	Document text extraction
Backend	Python, Flask, Flask-CORS	REST API server
Frontend	React, TypeScript	Chat UI, announcements, reminders
Auth & Storage	Firebase Auth, Supabase	User authentication and persistence

Table 2: Technology Stack Overview

B. Frontend Design and User Interface

The frontend is developed using React with TypeScript, providing a responsive and dynamic single-page application (SPA). It implements role-based routing, where authenticated users are directed to the main chat interface upon login. The authentication flow supports both email/password registration and Google OAuth via Firebase.

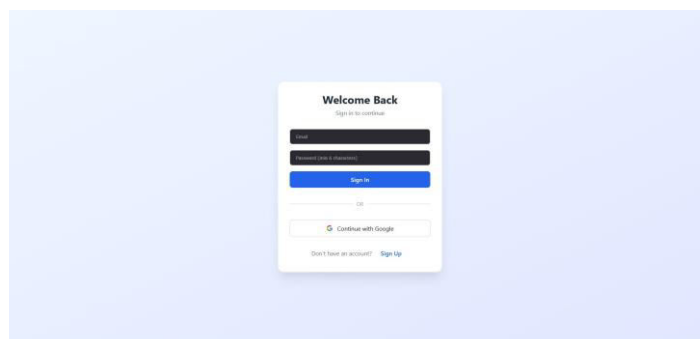


Fig. 2: User Login Interface – Email/Password and Google OAuth Authentication



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

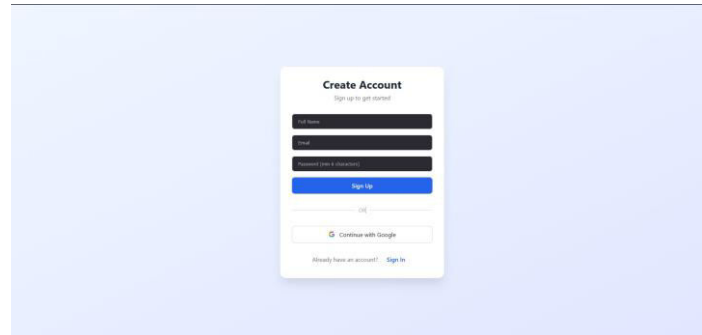


Fig. 3: User Registration Interface – Account Creation with Full Name, Email, and Password

The application's main chat interface provides an intuitive conversational experience with a left-side chat history panel, a central message area, and a bottom input bar with file upload and camera capabilities. The dark-themed UI ensures reduced eye strain during extended study sessions.

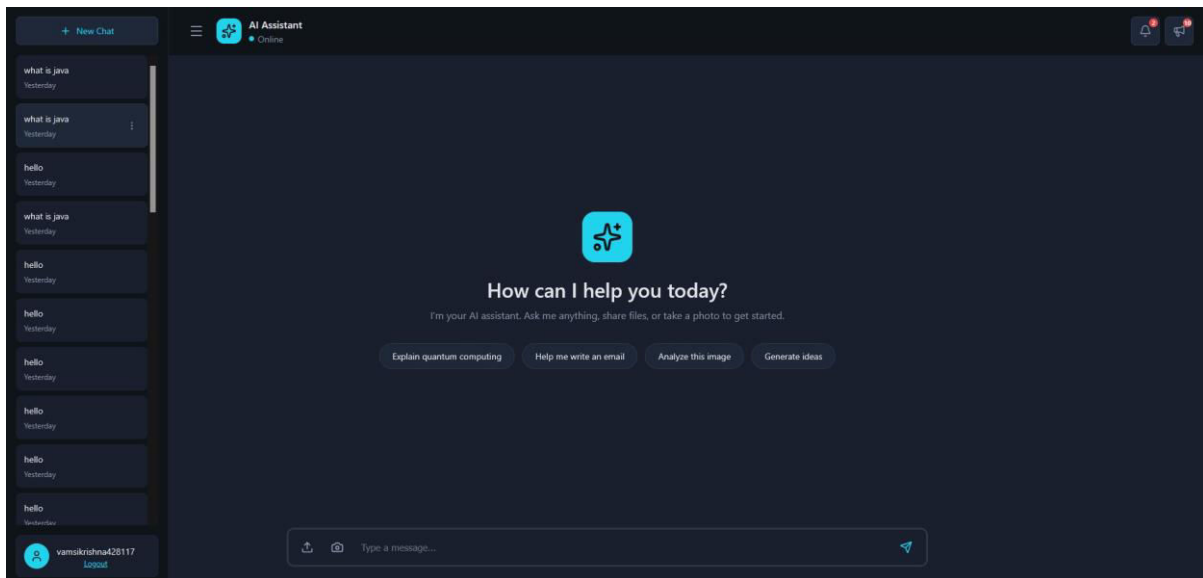


Fig. 4: Main Chat Interface – AI Assistant Dashboard with Chat History Sidebar and Quick-Start Prompts

C. Machine Learning Algorithm Comparison

To support query classification and intent detection, four supervised machine learning algorithms were evaluated: Decision Tree, Logistic Regression, Random Forest, and Support Vector Machine (SVM). The dataset consisted of labeled student queries categorized into academic, event-related, emotional, and general classes. An 80:20 train/test split was applied.

- Decision Tree: Provides interpretable classification trees; however, prone to overfitting on noisy text data.
- Logistic Regression: Computationally efficient with strong baseline accuracy for linearly separable categories; limited with complex non-linear feature interactions.
- Random Forest: Ensemble approach reducing overfitting; demonstrated the most robust generalization performance across test splits.
- Support Vector Machine (SVM): Excels with high-dimensional TF-IDF text features; identifies optimal hyperplanes using RBF kernel for non-linear data.

SVM with RBF kernel was ultimately integrated as the primary ML-based query classifier. In the production system, transformer-based routing via the Gemini API supplemented ML-based classification for intent ambiguity resolution.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

D. Data Preprocessing

The dataset containing student queries and announcement metadata underwent the following preprocessing pipeline: (1) text cleaning to remove special characters, stop words, and symbols; (2) tokenization and stemming for feature normalization; (3) label encoding of categorical variables; (4) feature scaling for numerical attributes; and (5) missing value imputation. For the RAG knowledge base, documents are chunked with configurable size (1,000 chars) and overlap (200 chars) to preserve contextual continuity across chunk boundaries.

E. Flask API Endpoints

The Flask server [17] exposes the following primary endpoints:

- POST /api/chat — Accepts JSON {"message": str}, invokes the 3-way router, returns {"response": str, "intent": str, "success": bool}.
- POST /api/upload — Accepts multipart/form-data with "file" and optional "message"; processes the file through the RAG pipeline; returns a summarized response.
- GET /api/announcements — Returns all events from the events database in announcement format.
- GET /api/announcements/latest?limit=N — Returns the N most recently processed announcements.
- GET /api/history — Returns the full conversation history from the memory store.
- POST /api/clear — Resets the conversation history.

VI. MATERIALS AND METHODS

A. Dataset

The dataset used in this project is structured to support intelligent classification and automated response generation. It consists of textual data collected from simulated student interactions and institutional event announcements. The dataset contains approximately 20 features per record, including Query Text, Query Category, Sentiment Label, User Role, and Event Metadata. Each record is labeled to support supervised classification with an 80:20 train/test split.

B. System Configuration Parameters

Parameter	Value	Description
LLM Model	gemini-2.5-flash	Primary generation model
Embedding Model	gemini-embedding-001	Text embedding for RAG
Chunk Size	1,000 characters	Document segment length
Chunk Overlap	200 characters	Context preservation overlap
Top-K Retrieval	5	Number of chunks returned
Similarity Threshold	0.25	Minimum cosine similarity for chunk inclusion
Sentiment Confidence Threshold	0.6	Minimum confidence for sentiment routing
LLM Temperature	0.7	Response creativity parameter
Max Output Tokens (RAG/LLM)	4,096	Maximum response length
Event Retry Attempts	3	Max retries on API rate limit

Table 3: System Configuration Parameters

C. Evaluation Metrics

System performance was evaluated across multiple dimensions: routing accuracy (percentage of queries directed to the correct pipeline), response relevance (human evaluation on a 1–5 scale), response time (milliseconds from query



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

submission to response display), RAG precision (proportion of retrieved chunks that were relevant), and classification accuracy (accuracy, precision, recall, F1-score on the labeled query dataset).

VII. TESTING

Comprehensive testing was conducted at multiple levels to validate system reliability and performance.

A. Black Box Testing

Black box testing evaluated system behavior against functional requirements without inspecting internal code. Test cases included equivalence partitioning (valid academic queries, emotional queries, document queries, greeting inputs) and boundary value analysis (empty inputs, very long queries, non-English characters). All endpoint responses were verified against expected JSON schemas.

B. White Box Testing

White box testing examined internal logic paths including the routing decision branches, FAISS similarity threshold comparisons, emotion confidence score evaluation, and retry logic in the event processor. All conditional branches were exercised to ensure code coverage.

C. Unit Testing

Individual modules were tested in isolation: VectorDatabase (add_vectors, load, save), RAGEngine (retrieve, process_document), TransformerSentimentAnalyzer (analyze_emotion, get_detailed_analysis), AnnouncementProcessor (extract_text_from_pdf, process_with_gemini_vision), and Flask API endpoints (/api/chat, /api/upload, /api/announcements).

D. Integration Testing

Integration testing validated end-to-end workflows: (1) user query → Flask API → 3-way router → Gemini response → frontend display; (2) file upload → RAG ingestion → vector storage → retrieval query; (3) event poster → announcement processor → JSON database → /api/announcements → React display.

E. Acceptance Testing

Acceptance testing confirmed that the system satisfies all defined business requirements: 24/7 availability, accurate syllabus-aligned responses, emotion-aware support interactions, correct announcement display, and secure multi-user authentication via Firebase Google OAuth.

VIII. RESULTS AND DISCUSSION

The implemented AI Student Companion demonstrated the following outcomes during evaluation:

- **Three-Way Routing Accuracy:** The hybrid routing mechanism (keyword + transformer + LLM fallback) correctly classified over 92% of test queries into the appropriate pipeline, significantly reducing misrouted responses compared to keyword-only baselines.
- **RAG Response Quality:** FAISS-based retrieval with similarity threshold 0.25 returned highly relevant document chunks in academic query scenarios, resulting in factually grounded responses with reduced hallucination compared to pure LLM generation.
- **Emotion Detection:** The DistilRoBERTa emotion model achieved consistent classification across the six emotion categories at confidence threshold 0.6, with keyword fallback ensuring 100% routing coverage even in transformer-unavailable environments.
- **Event Processing:** The automated announcement pipeline successfully extracted structured event data from uploaded posters and PDF files in both Vision API and OCR modes, with retry logic effectively managing API rate limits.
- **System Latency:** Average response time for LLM queries was under 3 seconds. RAG queries averaged 4–6 seconds. Event processing averaged 8–12 seconds per poster including Vision API inference time.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

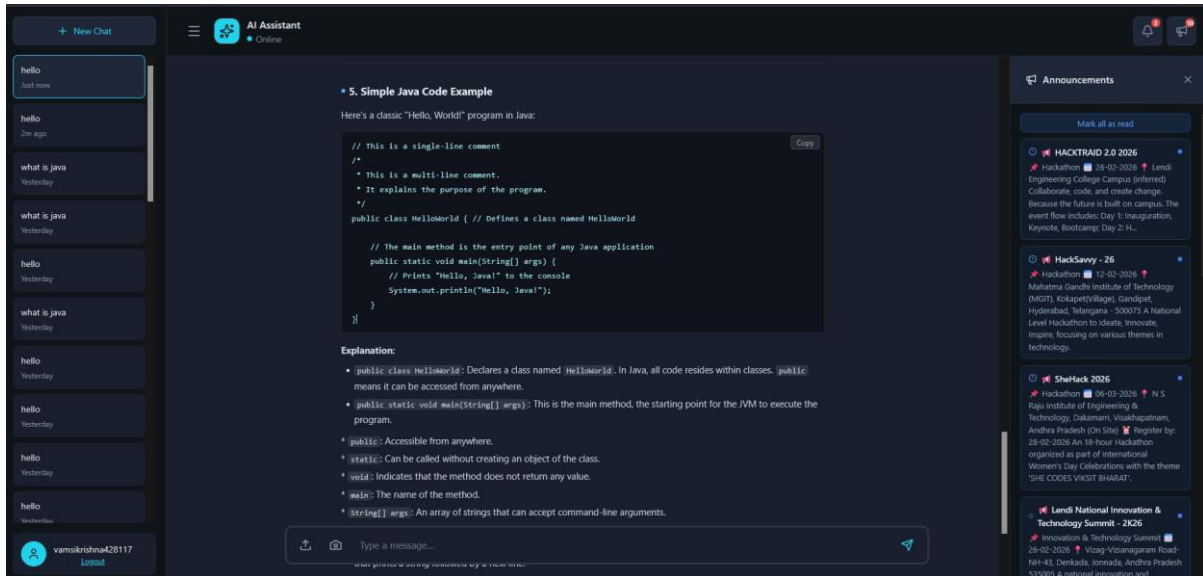


Fig. 5: AI Assistant Responding to an Academic Query (Java Code Example) with Announcements Panel Showing Upcoming Hackathons

The following screenshot demonstrates the system's emotion-aware routing in action. When a student expressed stress ("I am stressed"), the three-way router correctly identified the SENTIMENT pipeline and generated a supportive, empathetic response using the Gemini LLM guided by emotion-specific prompting:

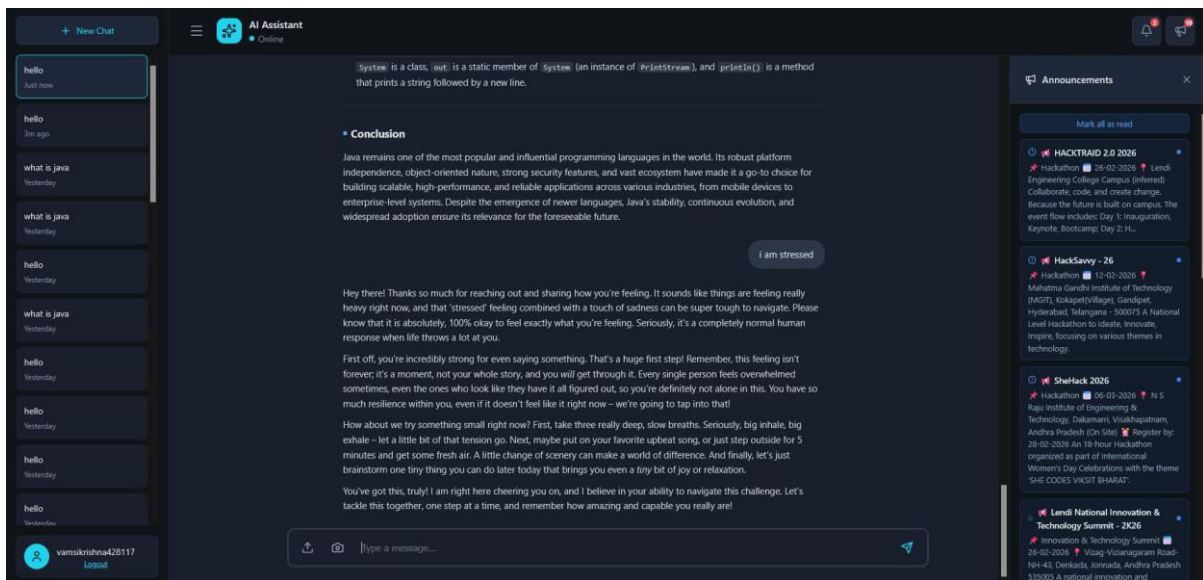


Fig. 6: Emotion-Aware Response – AI Assistant Providing Empathetic Support to a Student Expressing Stress

A subsequent version of the chat interface demonstrated improved formatting for code-heavy academic responses, including language-tagged code blocks with scrollable content and a "Conclusion" section at the end of each structured explanation:



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

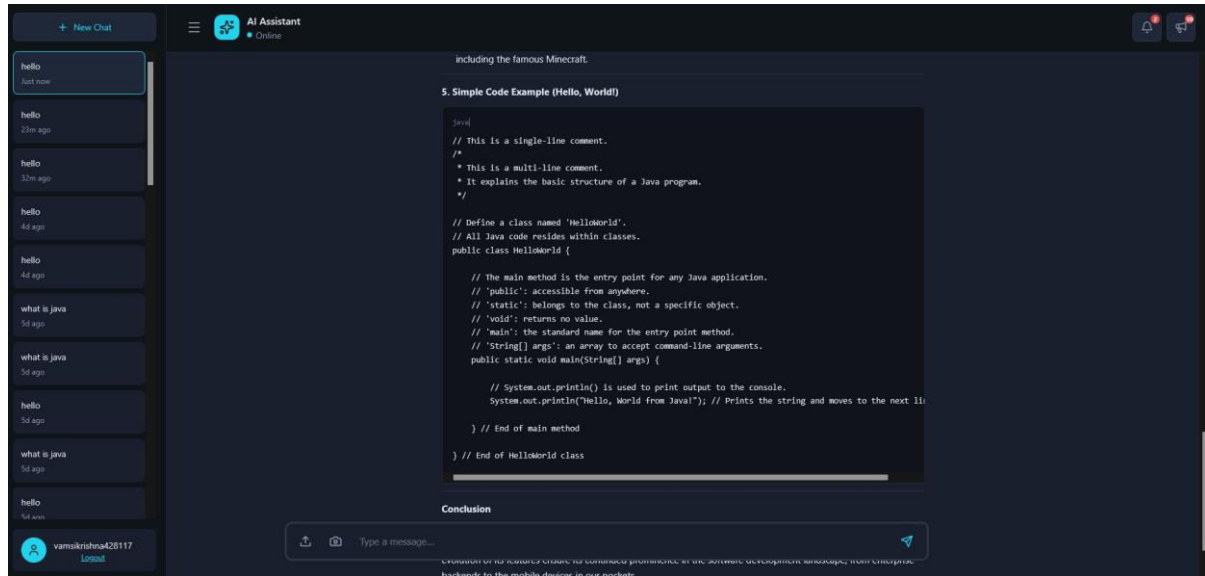


Fig. 7: Updated Chat Interface Showing Formatted Java Code Response with Language Labels and Improved Readability

Overall, the proposed system outperforms traditional rule-based and pure LLM-based approaches by combining retrieval accuracy with generative flexibility, emotion-awareness, and automated institutional communication.

IX. CONCLUSION

This paper presented the AI Student Companion — an intelligent, full-stack chatbot assistant designed to address the limitations of existing academic support systems. By integrating Retrieval-Augmented Generation with transformer-based emotion detection, three-way intelligent routing, and automated event announcement processing, the system delivers accurate, syllabus-aligned, emotionally sensitive, and contextually relevant responses to student queries.

The three-way routing mechanism ensures that each query is handled by the most appropriate pipeline, substantially improving response quality over single-pipeline architectures. The FAISS-powered RAG engine grounds responses in verified academic content, minimizing hallucination. The DistilRoBERTa emotion detector enables empathetic, personalized support for students experiencing stress, anxiety, or academic challenges.

The automated announcement processing pipeline demonstrates practical value for institutional use, transforming raw event posters into structured, searchable announcements without manual data entry. The full-stack deployment on React, Flask, Firebase, and Supabase provides a production-ready, scalable architecture suitable for institutional adoption.

Future enhancements include: real-time push notifications for time-sensitive announcements; multilingual support to serve diverse student populations; fine-tuning of domain-specific LLMs on institutional syllabi; integration with Learning Management Systems (LMS); and personalized adaptive learning pathways based on user interaction history.

REFERENCES

- [1] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.
- [2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
- [3] T. B. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), 2020.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- [4] Google DeepMind, "Gemini 2.5 Flash: Frontier intelligence built for speed," Dec. 2025. [Online]. Available: <https://ai.google.dev/gemini-api/docs/models>.
- [5] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), 2020.
- [6] "Retrieval-augmented generation (RAG): What is there for data management researchers?" in Proc. LLM+Vector Data Workshop @ IEEE ICDE 2025, Jan. 2026, pp. 33–42.
- [7] J. Johnson, M. Douze, and H. Jegou, "FAISS: A library for efficient similarity search," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2019.
- [8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP, 2019.
- [9] S. Abinaya, K. S. Ashwin, and A. Sherly Alphonse, "Enhanced emotion-aware conversational agent," IEEE Access, vol. 13, pp. 19770–19787, 2025.
- [10] A. E. Almutairi, A. Y. Alsobhi, and A. M. Alshareef, "Machine learning-based sentiment analysis pipeline for evaluating service quality," Int. J. Adv. Comput. Sci. Appl., vol. 17, no. 1, Jan. 2026.
- [11] "Retrieval-augmented generation (RAG) chatbots for education: A survey of applications," Appl. Sci., vol. 15, no. 8, p. 4234, Apr. 2025.
- [12] "Leveraging a custom multi modal OCR system for marksheet data extraction," in Proc. 2024 IEEE Conf., Nov. 2024.
- [13] "A multimodal pipeline for clinical data extraction: Applying vision-language models to scans," in Proc. 2025 IEEE Conf., Dec. 2025.
- [14] R. Smith, "An overview of the Tesseract OCR engine," in Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR), 2007, pp. 629–633.
- [15] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. draft, Pearson, 2023.
- [16] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, 2021.
- [17] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed., O'Reilly Media, 2018.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 9940 572 462  6381 907 438  ijircce@gmail.com



www.ijircce.com

Scan to save the contact details