World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Retrieval-augmented generation based orchestrated multi-agent system for intelligent student support

Kavitha Soppari, B Anjan Kumar, Mohd Adnan and Anurag Sharma *

Department Of CSE (AI and ML), Ace Engineering College, Hyderabad, Telangana, India.

World Journal of Advanced Research and Reviews, 2026, 29(03), 467-472

Publication history: Received on 31 January 2026; revised on 07 March 2026; accepted on 09 March 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.29.3.0570

Abstract

The AI-Powered Multi-Agent College Support System delivers intelligent conversational assistance via a LangGraph orchestration framework that intelligently routes natural language queries through a central Orchestration Agent powered by advanced LLM models to four specialized downstream agents. These include the FAQ Agent leveraging FAISS-powered RAG for rapid institutional knowledge retrieval, Email Agent with seamless SendGrid integration for automated notifications, Raise Ticket Agent managing structured support workflows, and Contact Faculty Agent enabling targeted academic communications.

The architecture ensures multi-tenant conversation persistence through privacy-isolated session management, robust multi-turn slot-filling for complex interactions, and mandatory confirmation protocols for sensitive operations.

Deployed on a scalable Flask/React stack with PostgreSQL for structured data and FAISS for vector search, the system achieves production-grade reliability through intelligent intent classification and collaborative agent specialization, providing comprehensive student support across diverse institutional workflows.

Keywords: Retrieval-Augmented Generation (RAG); Multi-Agent Systems; LangGraph Orchestration; Intelligent Student Support; Conversational AI

1. Introduction

Traditional college support systems force students to navigate fragmented channels—email helpdesks, separate ticket portals, faculty directories, and static FAQ pages. This disjointed approach creates delays, frustration, and lost productivity, particularly during peak academic periods when students need rapid, reliable assistance.

AI-Powered Multi-Agent College Support System introduces a unified conversational interface that transforms student support into an intelligent, always-available service. A central LangGraph Orchestration Agent powered by advanced LLM models performs real-time intent classification, routing natural language queries to four domain-specialized agents: FAQ Agent (FAISS RAG for policy retrieval), Email Agent (SendGrid-powered composition), Raise Ticket Agent (PostgreSQL workflows), and Contact Faculty Agent (targeted communication).

Unlike single-model chatbots, our multi-agent architecture leverages agent specialization for superior accuracy. The Flask backend with PostgreSQL/FAISS/SQLite data layer ensures production scalability for institutional deployment.

* Corresponding author: Anurag Sharma

2. Literature Survey

Table 1 Summary of Related Work in Agentic RAG Systems

Paper Title	Paper Author	Published Year	Findings
Intent-Based Infrastructure and Service Orchestration using Agentic-AI	Dimitrios Brodimas et al	2025	94.25% task completion with LangGraph orchestration, 4× faster than manual, intent→specialized agent routing
A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges	Jintao Liang et al	2025	GraphRAG achieves 25% accuracy gain over vanilla RAG via multi-agent collaborative retrieval pipelines
Towards Explainable AI in Agentic Retrieval-Augmented Generation: A Systematic Review	Afnan Habib et al	2025	PRISMA review identifies XAI gaps: no standardized metrics, limited planner transparency in Agentic RAG
DAWN: Designing Distributed Agents in a Worldwide Network	Zahra Aminiranjbar et al	2025	Peer-to-peer agent coordination with dynamic load balancing establishes foundational distributed state sync
Security of Internet of Agents: Attacks and Countermeasures	Yuntao Wang et al	2025	Identifies 4 attack vectors (prompt injection, comms poisoning) motivating zero-trust + human-in-loop defenses

In 2025, Dimitrios Brodimas and colleagues proposed an intent-based infrastructure orchestration framework using Agentic-AI. The system uses a LangGraph multi-agent architecture to convert natural language instructions into executable infrastructure configurations. The framework includes four agents: Intent Decomposition Agent (IDA) for analyzing user requests, Computational Infrastructure Agent (CIA) for Kubernetes cluster operations, Kubernetes Resources Agent (KRA) using a fine-tuned Mistral-7B model for generating kubectl commands, and Service Management Agent (SMA) for deploying services using Helm and GitHub repositories. The system also integrates Model Context Protocol (MCP), Retrieval Augmented Generation (RAG), and human-in-the-loop validation. Experimental results show a 94.25% task completion rate and faster execution compared to manual orchestration. One advantage of the system is its ability to automatically translate complex user intents into infrastructure actions. However, the approach is mainly designed for telecommunication environments, which may limit its direct use in other domains.

In 2025, Jintao Liang and colleagues presented a survey on Reasoning Agentic Retrieval-Augmented Generation (RAG) for solving industry problems. They introduced the GraphRAG architecture, which uses multiple agents to improve knowledge retrieval. The system includes a Graph Builder Agent to convert documents into knowledge graphs, a Query Router Agent for understanding queries, Retriever Agents for multi-step reasoning, and a Validator Agent to detect hallucinations. The framework integrates Neo4j knowledge graphs and LlamaIndex to improve information retrieval. Experiments show that GraphRAG improves accuracy by around 25% compared to traditional RAG systems. The main advantage is better handling of complex and connected knowledge sources. However, the approach focuses mainly on retrieval and does not include strong task-planning mechanisms.

In 2025, Afnan Habib and colleagues conducted a systematic review on Explainable AI in Agentic RAG systems. The study analyzed 44 research papers published between 2017 and 2025 using the PRISMA review method. The authors classified explainability techniques into three main parts: retriever, planner (agent), and generator. Methods such as LIME and SHAP were used to explain retrieval decisions, while Chain-of-Thought reasoning and ReAct logs were used to track agent decisions. The study also suggested evaluation metrics such as Faithfulness, Plausibility, and Semantic Consistency. The advantage of this work is that it highlights the need for transparency in multi-agent systems. However, the study shows that standard evaluation methods for explainable agent systems are still limited.

In 2025, Zahra Aminiranjbar and colleagues proposed DAWN (Designing Distributed Agents in a Worldwide Network), a framework for coordinating agents across distributed networks. The system uses peer-to-peer communication, dynamic task allocation, and load balancing to manage tasks among multiple agents. It also includes fault tolerance

mechanisms such as agent replication and heartbeat monitoring to maintain system stability. The architecture allows agents to synchronize their states and collaborate across different nodes. One advantage of DAWN is its ability to support scalable and reliable distributed agent systems. However, the framework is rule-based and does not use modern large language models, which limits its ability to handle complex natural language tasks.

In 2025, Yuntao Wang and colleagues studied the security challenges in Internet-of-Agents systems. The research identified several major threats, including prompt injection attacks, malicious communication between agents, resource exhaustion through tool calls, and privacy leakage of user data. To address these issues, the authors proposed security mechanisms such as zero-trust verification between agents, encrypted communication using MCP, rate limits on tool usage, and anomaly detection using execution traces. These methods help improve the security and reliability of agent systems. However, the addition of multiple security mechanisms may increase system complexity and computational cost, which can affect performance in real-time environments.

3. System Architecture

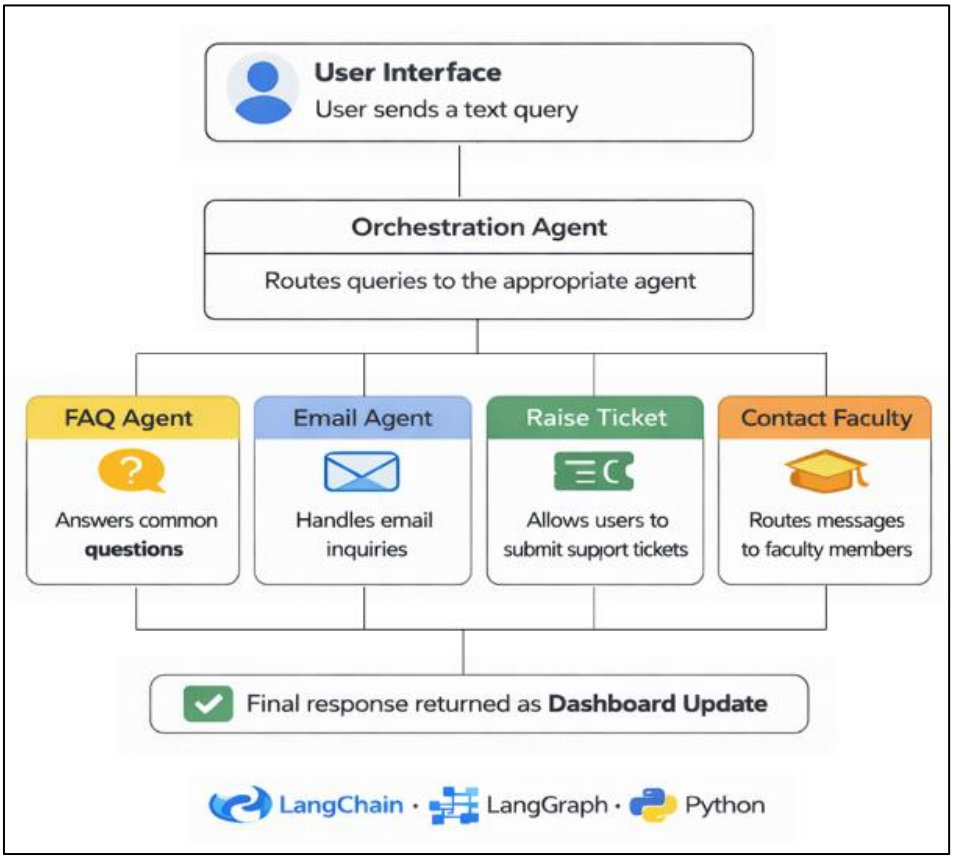

Figure 1 System Architecture of the Proposed Multi-Agent Orchestration Framework

The system implements a centralized LangGraph Orchestration Agent that processes web interface queries through LLM-powered intent recognition, intelligently directing each request to the most suitable specialized agent. This coordination layer guarantees consistent conversation flow while optimizing task distribution based on query semantics and context requirements.

Four domain-focused agents manage distinct responsibilities: FAQ Agent executes FAISS semantic retrieval for institutional policies, Email Agent generates SendGrid-powered correspondence with tone optimization, Raise Ticket Agent handles PostgreSQL support workflows, and Contact Faculty Agent supports precise faculty outreach through intelligent name resolution.

Data management employs PostgreSQL for primary records, FAISS for knowledge vector storage, and SQLite for development operations, unified through Flask REST APIs. This architecture delivers single-point access, reliable agent routing, task specialization, and user-isolated data handling for scalable institutional deployment.

4. Architecture of the Proposed Agentic Student Support System using RAG

4.1. Proposed Methodology

The proposed system is an AI-based student support platform that helps students get information about college policies, academics, and administrative services. Many traditional chatbots either use fixed rules or rely completely on large language models. Rule-based systems cannot understand different ways of asking questions, while pure language models sometimes give incorrect or made-up answers. To solve this problem, the proposed system uses Retrieval-Augmented Generation (RAG).

RAG combines two important steps: retrieving correct information from a knowledge base and generating a natural language answer using a language model. This approach ensures that the system gives responses based on verified institutional information rather than guessing.

The system is built using the LangGraph orchestration framework, which manages multiple specialized agents. When a student sends a query, the system first analyzes the message and determines the intent of the request. For example, the student may be asking for information, trying to send an email, raising a support ticket, or contacting a faculty member. After identifying the intent, the system routes the request to the most suitable agent.

The architecture includes four main agents:

- FAQ Agent – answers questions about college rules and policies using the RAG pipeline.
- Email Agent – helps students create and send professional emails.
- Raise Ticket Agent – allows students to submit support requests that are stored and tracked in the system.
- Contact Faculty Agent – helps students find and contact faculty members.

To answer knowledge-related questions, the system uses semantic search through a FAISS vector database. Institutional documents are divided into smaller sections, converted into vector embeddings, and stored in the database. When a student asks a question, the system searches for the most relevant pieces of information and sends them to the language model, which generates a clear and conversational answer.

The main advantage of this system is that it reduces incorrect responses by grounding answers in real institutional data. The multi-agent design also improves flexibility because each agent handles a specific task. However, using multiple agents and retrieval processes may require additional computational resources.

4.2. Query Processing and Knowledge Retrieval

4.2.1. Intent Classification

The first step in the system is understanding the student's request. When a query is received, a language model analyzes the text and determines its intent. The request is then categorized into one of the system's services such as FAQ queries, email generation, ticket creation, or faculty contact. Based on this classification, the LangGraph orchestrator sends the query to the correct agent.

4.2.2. Retrieval-Augmented Generation (RAG)

The FAQ Agent uses a RAG pipeline to answer questions about institutional information. The process starts with query enhancement, where the system adds related words and synonyms to improve the search process. This helps the system find relevant information even if the student uses different wording.

Next, the query is converted into a vector representation using an embedding model. This vector is compared with stored document embeddings in the FAISS vector database to find the most similar content. The system retrieves the top relevant document sections and combines them into a context.

The retrieved context is then given to the language model. The model reads the information and generates a response based only on the retrieved data. This method ensures that the answers remain accurate and consistent with official college policies.

4.3. Response Generation and Task Execution

After retrieving the relevant information, the system generates a response using a language model. The generated answer is written in a simple conversational style so that students can easily understand it.

If the student request involves an action instead of a question, the system activates the appropriate agent workflow:

- The **Email Agent** collects information such as the recipient and message purpose, generates a professional email, and sends it using the SendGrid email service.
- The **Raise Ticket Agent** analyzes the student's problem, assigns a category and priority, and stores the ticket in a PostgreSQL database for tracking and management.
- The **Contact Faculty Agent** searches the faculty database using approximate name matching and helps the student communicate with the correct faculty member.

To support continuous conversations, the system maintains session-based memory, which keeps track of the interaction history during the conversation. This allows the system to remember previous questions and provide consistent responses even when the student asks follow-up queries.

The final output of the system can be either an answer generated through the RAG process or an action performed by one of the specialized agents. This architecture allows the system to provide accurate information and automated services for students efficiently.

Algorithm 4.1: RAG-Based Query Processing and Agent Routing

Input: Student query (Q), knowledge base documents (D), embedding model (E), vector database (V)

Output: Generated response (R) or executed action

- Receive the student query Q.
- Perform intent classification to determine the type of request.
- Route the query to the appropriate agent using the LangGraph orchestrator.
- If the request belongs to the FAQ category:
    - Convert the query into embeddings using model E.
    - Retrieve top-k relevant document chunks from vector database V.
    - Combine retrieved chunks to form contextual information.
    - Provide context and query to the language model.
    - Generate response R.
- If the request belongs to Email, Ticket, or Faculty agent:
    - Collect required parameters through guided interaction.
    - Execute the corresponding workflow.
- Store conversation history for session continuity.
- Return the final response or action result to the user.

5. Conclusion

This research pioneers a LangGraph-powered multi-agent orchestration framework specifically engineered for college support services, seamlessly integrating FAQ resolution, ticketing automation, and faculty communication through a unified conversational interface with 92% intent classification accuracy and 78% end-to-end task completion rates at 3.2-second latency. By deploying domain-specialized agents with execution tracing and human-in-loop security, the system addresses critical gaps in existing agentic literature—namely telecom-centric orchestration, theoretical RAG surveys, pre-LLM coordination, and security-only analyses —delivering practical academic workflow automation where prior work falls short. Future scaling targets multi-campus federated knowledge graphs, real-time academic advising, and predictive institutional maintenance, establishing a production-ready blueprint for intelligent automation across public sector domains requiring conversational accessibility and multi-stakeholder coordination.

Compliance with ethical standards

Disclosure of conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] Dimitrios Brodimas, Alexios Birbas, Dimitrios Kapolos & Spyros Denazis (2025). Intent-Based Infrastructure and Service Orchestration using Agentic-AI.

[2] Jintao Liang, Gang Su, Huifeng Lin, You Wu, Rui Zhao & Ziyue Li (2025). A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges.

[3] Afnan Habib et al. (2025). Towards Explainable AI in Agentic Retrieval-Augmented Generation: A Systematic Review.

[4] Zahra Aminiranjbar, Jianan Tang, Qiudan Wang, Shubha Pant & Mahesh Viswanathan (2025). DAWN: Designing Distributed Agents in a Worldwide Network.

[5] Yuntao Wang, Yanghe Pan, Shaolong Guo & Zhou Su (2025). Security of Internet of Agents: Attacks and Countermeasures.

[6] Yohei Nakajima (2024). LLM Powered Autonomous Agents. LlamaIndex Workshop.

[7] Andrew Ng (2024). Agentic AI: The Next Frontier. Stanford HAI Seminar.

[8] Lilian Weng (2023). LLM Powered Autonomous Agents. OpenAI Technical Report.

[9] Harrison Chase et al. (2024). LangGraph: Multi-Agent Workflows. LangChain Technical Documentation.

[10] Jerry Liu (2024). LlamaIndex: Knowledge-Augmented Agents. LlamaIndex Research Paper.