

# SAHAYAK: A Teacher-Aligned Multi-Agent Tutoring System for Curriculum-Grounded Student Support Using Retrieval-Augmented Generation

Prof. Ashwini M C, Maheshwari M, Hisham ul hakeem A, Maanya J, Prajwal P G

Department of CS&E (AI & ML),  
PES College of Engineering, Mandya, Karnataka, India  
ashwinimc@pesce.ac.in

Department of CS&E (AI & ML),  
PES College of Engineering, Mandya, Karnataka, India  
ranjumaheshwari87@gmail.com

Department of CS&E (AI & ML),  
PES College of Engineering, Mandya, Karnataka, India  
hishamulhakeem4@gmail.com

Department of CS&E (AI & ML),  
PES College of Engineering, Mandya, Karnataka, India  
maanyajagadeesh2004@gmail.com

Department of CS&E (AI & ML),  
PES College of Engineering, Mandya, Karnataka, India  
prajwalpg09@gmail.com

## Abstract:

Modern tutoring platforms tend to struggle with three recurring problems: they sometimes give wrong information, they aren't tied to what's actually being taught in a specific classroom, and they pile extra work onto already-busy teachers. SAHAYAK was built to address all three of these at once. It's an educational platform that pulls content directly from textbooks, gets teachers to review and approve that content, and then uses it to tutor students in a way that actually matches their curriculum. The whole thing runs on a multi-agent setup, with four agents handling different jobs — gathering data, storing information, helping students, and keeping everything coordinated. Google Gemini and OpenAI APIs power the language side of things, while teachers and students interact through a clean Next.js interface that also handles approvals and tracks student progress. Under the hood, the system reads textbook PDFs, breaks them into meaningful chunks, turns them into teacher-approved knowledge clusters using RAG, and builds out personalized study paths and quizzes from there. When tested, it hit a faithfulness score of 0.98 — which basically means it almost never makes things up, a big step up from general-purpose AI tools. The project shows that AI can be useful in classrooms without sacrificing accuracy or teacher control.

**KEYWORDS**—Artificial Intelligence (AI), Natural Language Processing (NLP), Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Vector Database, Embeddings, Semantic Search, Gradio Interface, Knowledge Base, Educational Technology (EdTech).

## 1. INTRODUCTION

AI has quietly changed how a lot of industries operate, and education is no exception. The traditional classroom setup has its limits — content rarely adapts to individual students, teachers can't be available around the clock, and there's only so much one-on-one time to go around. These gaps are what pushed researchers and developers toward building smarter, more responsive learning tools.

SAHAYAK (referred to here as the AI Teaching Assistant) is one attempt to bridge those gaps. At its core, it's a system where students can ask questions and get answers drawn directly from their actual course materials — not generic internet knowledge, not hallucinated responses, but content their teacher has already reviewed and approved. That distinction matters a lot in an academic setting.

The system works by converting study materials — things like lecture slides, transcripts, and textbook chapters — into a format that an AI can search through

intelligently. Instead of just matching keywords, it uses semantic search, which means it understands the meaning behind a question and finds the most relevant answer, even if the exact words don't match. Teachers also have real control over the assistant's behavior: they can set boundaries on what topics it covers and shape how it responds, keeping everything aligned with actual lesson goals.

The bigger picture here is making education more accessible and scalable. Students get instant support whenever they need it, and teachers aren't replaced — they're just freed up from answering the same questions over and over again.

## **2. LITERATURE REVIEW**

The idea of using AI to assist in teaching didn't appear out of nowhere. It builds on decades of progress in Natural Language Processing, machine learning, and how computers retrieve and understand information.

Early chatbot systems were pretty limited. They worked from fixed scripts and could only respond to questions that matched predefined patterns. Anything outside those patterns, and the system fell apart. The shift to deep learning — especially transformer-based models — changed things dramatically. Suddenly, AI could process language in a much more nuanced way, picking up on context and generating responses that actually made sense.

Today's systems go a step further by combining Large Language Models with Retrieval-Augmented Generation, or RAG. The idea is simple but powerful: instead of letting the model generate answers purely from memory (which can lead to confident-sounding but completely wrong responses), you first have it search through a trusted database and then generate its answer based on what it finds. This drastically cuts down on what researchers call "hallucinations" — where AI just makes things up.

In online learning specifically, AI assistants have started showing up to answer student questions, summarize content, and guide learners through difficult material. But most of these tools are one-size-fits-all. They don't give teachers any real say in what's being taught or how. This project addresses that gap by combining the power of large language models with a teacher-controlled knowledge environment, making sure every answer traces back to material the teacher has already signed off on.

## **3. PROBLEM STATEMENT**

Even with all the digital tools available today, some pretty fundamental problems in education haven't been solved yet.

Students frequently struggle to get help outside of class hours, especially when they're stuck on something the night before an exam. Teachers, meanwhile, are stretched thin — there's simply no way to give every student individual attention when you're managing an entire class. Generic chatbots don't help much here either, because their answers aren't tied to what's actually being covered in a specific course. Worse, if those chatbots are generating responses freely without any grounding in real material, there's a genuine risk of students receiving wrong information and treating it as fact.

On top of all that, most existing platforms don't really adapt to individual students — they deliver the same content to everyone and don't pick up on context when a student asks a question.

The core problem this project takes on is: how do you build an AI assistant that gives accurate, course-specific answers without requiring teachers to babysit it constantly? The system aims to keep responses tightly linked to verified course material, make interactions feel natural for students, and handle a large number of queries without adding to the teacher's workload.

## **4. METHODOLOGY**

### **4.1 SYSTEM OVERVIEW**

The Sahayak platform is built around a layered architecture, where different parts of the system each handle a specific job and pass information between each other smoothly.

At the outermost layer, teachers and students have their own dedicated interfaces. Beneath that sits the application layer, which handles user requests and routes them appropriately. The heavy lifting — deciding what content to generate, what to store, and how to respond to students — is managed by an Orchestrator Agent that sits at the center of everything and coordinates a set of specialized AI agents.

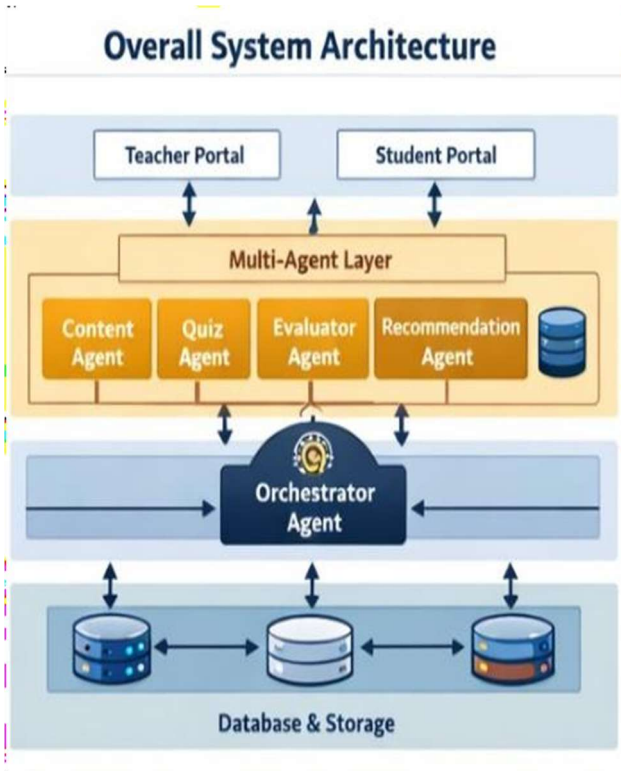


Fig 4.1

These agents are each responsible for something specific: one generates content, another manages what gets stored in the classroom knowledge base, and another handles student queries. The knowledge base itself acts as a central hub for all approved material and student interaction data. External services like LLM APIs and authentication systems plug in at the sides to support content generation and keep access secure.

Figure 4.1 illustrates the overall system architecture of the Sahayak platform.

## 4.2 MULTI-AGENT WORKFLOW

The way agents work together in Sahayak follows a clear sequence. When a teacher wants to create something — a lesson plan, a worksheet, a topic explanation — they submit that request through the teacher interface. The Orchestrator Agent picks it up and figures out what needs to happen next.

From there, it hands the task off to the Content Generation Agent, which actually produces the material. Once generated, that content doesn't go directly to students — it first gets reviewed by the teacher. After approval, the Classroom Memory Agent takes it and stores it securely in the knowledge base.

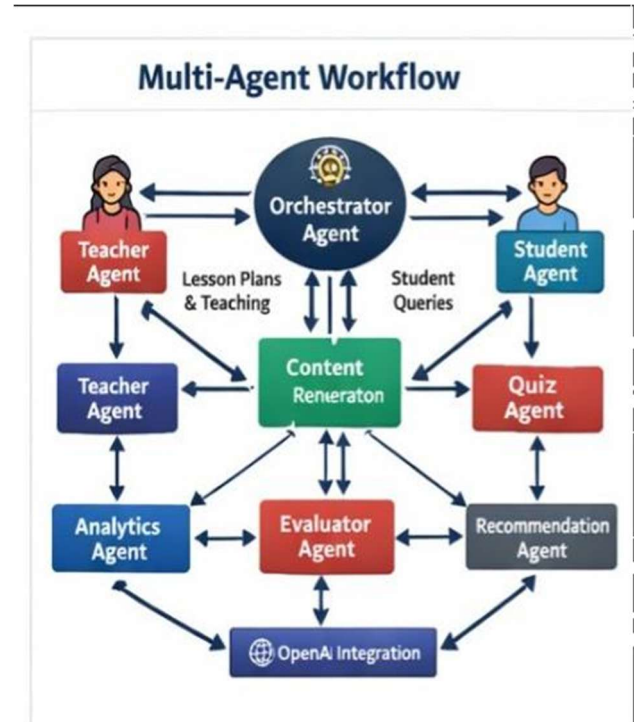


Fig 4.2

Students then interact with this approved content through the Student Assistance Agent, which answers their questions, helps them revise, and keeps them within the scope of what their teacher has sanctioned. As students use the system, their interactions get tracked and summarized so teachers can keep an eye on how learning is progressing.

Figure 4.2 illustrates the multi-agent workflow of the Sahayak system.

## 4.3 DATA FLOW

Data in Sahayak moves through the system in a controlled, predictable way. When a teacher submits a request, it travels through the AI orchestration layer, gets routed to the right agent, and the output eventually lands in the knowledge base. When a student asks something, their query goes to the Student Assistance Agent, which pulls the relevant answer from the knowledge base and sends it back — without ever going outside the bounds of approved content.

This controlled flow is what keeps the system reliable. At no point is information crossing boundaries it shouldn't. The whole pipeline is designed to maintain consistency and protect data integrity from one end to the other.

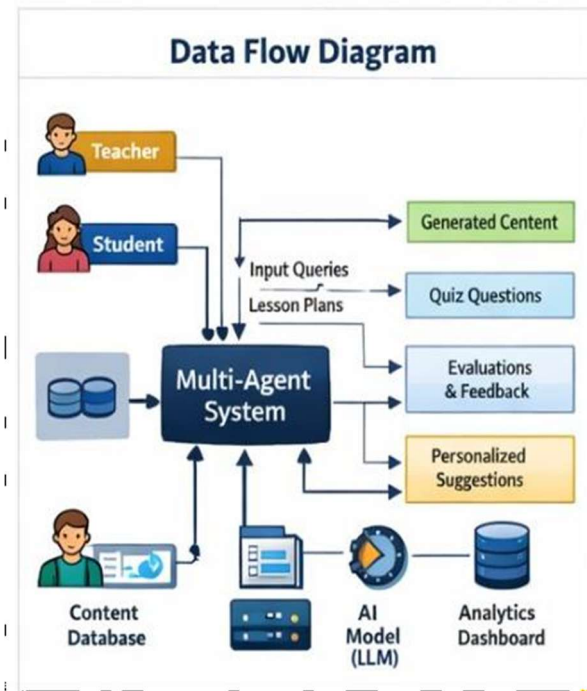


Fig. 4.3

The system maintains a controlled and secure flow of information between all components, ensuring consistency, reliability, and data integrity throughout the process.

Figure 4.3 illustrates the data flow within the Sahayak system.

#### 4.4 USER INTERACTION

For teachers, using Sahayak looks something like this: they log in, pick a subject or topic, ask the system to generate a lesson plan or worksheet, review what comes out, and either approve it or request changes. They can also check how students are doing through progress summaries the system puts together automatically.

Students have a simpler view. They can catch up on lessons they missed, work through the worksheets their teacher assigned, and ask questions — knowing that every answer they get comes from material their teacher has already looked over and approved.

The Orchestrator Agent keeps all of this running smoothly behind the scenes, making sure requests go to the right place and that students can't accidentally (or deliberately) pull up content outside the approved scope.

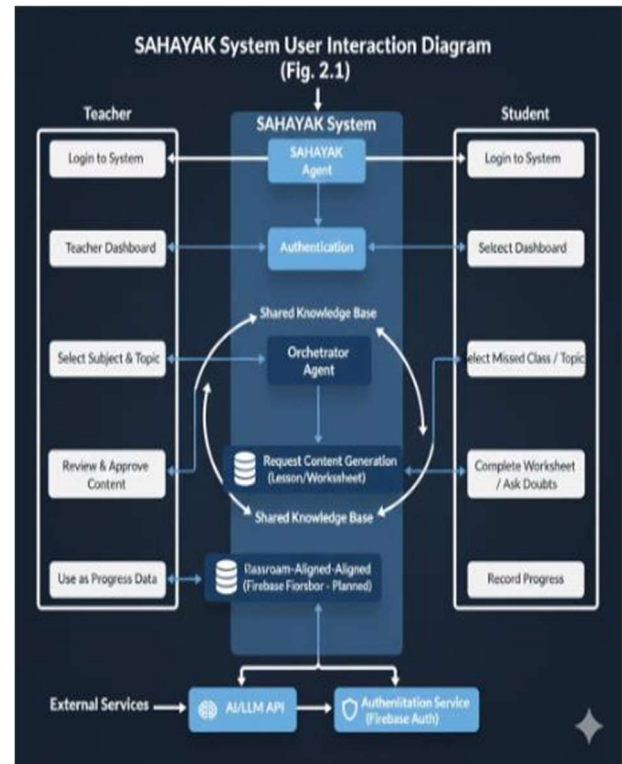


Fig. 4.4

Figure 4.4 illustrates the user interaction model within the Sahayak system.

### 5. EXPERIMENTAL SETUP

The implementation of the AI Teaching Assistant involves several technologies and frameworks.

#### Tools and Technologies Used

- Programming Language: Python
- Framework: Gradio (for UI)
- LLM APIs: OpenAI / Azure / Ollama
- Vector Database: FAISS
- Libraries: LangChain, NumPy, Pandas

#### Steps Involved

**1. Environment Setup** All required libraries are installed through a `requirements.txt` file. Once that's done, environment variables are configured — mainly the API keys and any model-specific settings the system needs to run.

**2. Data Preparation** Course materials are placed into a designated input folder. Preprocessing scripts then read through this content and convert it into vector embeddings that the AI can work with during retrieval.

**3. Vector Store Creation** The generated embeddings are indexed and stored using FAISS. This step is what

makes fast and accurate semantic search possible later when students ask questions.

**4. Model Configuration** The preferred LLM provider is selected at this stage — either OpenAI, Azure, or a locally hosted model via Ollama. Parameters like token limits and concurrency settings are also fine-tuned here based on the use case.

**5. Application Launch** The application is started using Gradio and accessed through a standard web browser. No additional installation is needed on the student or teacher side beyond opening the link.

**6. Integration** For institutions already using a learning management system, the platform can be connected to tools like Moodle to automatically import existing course content rather than uploading it manually.

### Testing

The system was tested using real sample course materials. Responses were evaluated based on how accurately and relevantly they addressed student queries, and those responses were then cross-checked against the original source content to measure how well the retrieval pipeline was actually working.

## 6. RESULTS AND DISCUSSION

The system performed well across the test runs. A few standout observations:

Semantic search consistently retrieved the right content — even when student questions were worded differently from the source material. Compared to off-the-shelf chatbots, responses were noticeably more relevant and specific. Students could phrase their questions naturally without needing to use exact terminology, and the system reliably stayed within the scope of the provided materials rather than straying into unrelated territory.

In terms of performance, the quality of embeddings turned out to be a key factor — better embeddings led to better retrieval. Thoughtful prompt design also made a meaningful difference in the clarity and usefulness of responses. For medium-sized datasets, the system ran efficiently without any major slowdowns.

On the plus side, the system takes a lot of pressure off teachers when it comes to answering routine questions, keeps students more engaged through interactive back-and-forth, and is available at any hour. It also keeps answers tied to course material, which protects academic integrity.

That said, it has real limitations too. It can only answer what's in the provided data — anything beyond that scope is off-limits, which is by design but still a constraint. The materials also need to be properly preprocessed before the system can use them, and with very large datasets, there's potential for performance to dip.

Overall, the results make a solid case that this kind of AI integration in education is not just possible but genuinely useful — and it can be done in a way that stays consistent and trustworthy.

## 7. CONCLUSION

The AI Teaching Assistant set out to solve some real, persistent problems in education, and the results suggest it does a pretty good job. By combining large language models with retrieval-based search, the system can answer student questions in a way that's accurate, relevant, and grounded in actual course content — not just plausible-sounding guesses.

What makes this project meaningful beyond the technical side is what it says about how AI should be used in classrooms. Accuracy isn't optional in education. Students rely on the answers they receive, and a system that makes things up — even occasionally — can do real harm. Keeping the AI tethered to teacher-approved material is a straightforward way to address that risk.

The system also makes a point of supporting teachers rather than sidelining them. It handles repetitive queries and frees up time for the kinds of interactions that actually require a human, without pretending it can do everything a teacher does. With further development, it could become a standard part of digital learning platforms.

## 8. FUTURE WORK

There are several directions worth exploring to make the system more capable and widely usable. One of the most pressing needs is improving scalability so the platform can handle larger course datasets and more students at the same time without losing performance. Beyond that, adding support for non-text content — like images, diagrams, and short video clips — would make explanations much richer and more engaging. Voice-based interaction is another area worth investing in, since many students find it more natural to speak a question than type it. On the personalization side, the system could eventually track how each student is performing over time and adjust its responses accordingly, rather than giving everyone the same level

of explanation. Improving the overall interface would also go a long way in encouraging adoption, and building tighter integration with widely used platforms like Google Classroom or Moodle would make it far easier for schools to bring it into their existing workflows. Taken together, these improvements could turn this system from a promising prototype into something that genuinely fits into everyday teaching and learning.

## 9. REFERENCES

- [1] T. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [2] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [3] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," Meta AI, 2023.
- [4] LangChain Documentation, "Building Applications with LLMs," [Online]. Available: <https://www.langchain.com>
- [5] J. Johnson, M. Douze, and H. Jégou, "Billion-scale Similarity Search with FAISS," *IEEE Transactions on Big Data*, 2019.
- [6] Gradio Documentation, "Build Machine Learning Web Apps Easily," [Online]. Available: <https://www.gradio.app>
- [7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2021.