IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

An Intelligent Academic Assistance Framework Using LLM Models

B S Mahalakshmi[1], Shobha Rani B R[2]

Department of MCA, Dr. Ambedkar Institute of Technology, Bengaluru, India[1]

Associate Professor, Department of MCA, Dr. Ambedkar Institute of Technology, Bengaluru, India[2]

Abstract: AcademiAI is an intelligent academic assistance framework designed to support students and educators by unifying study-material comprehension, doubt-resolution, assignment evaluation, and personalized learning planning within a single system. It integrates document processing, Optical Character Recognition, Retrieval-Augmented Generation (RAG), and Generative AI-based conversational tutoring to help learners understand course content, get instant answers grounded in their own study material, and receive structured feedback on assignments and essays. AcademiAI accepts textbooks, lecture notes, syllabi, and handwritten class notes in PDF, DOCX, PPTX, or image form. Scanned and handwritten material is passed through Tesseract OCR to extract text, which is then chunked and embedded into a vector database for retrieval. Extracted content and user queries are routed through the Groq API to two large language models — a lightweight model for fast document summarization/embedding-support tasks and a larger instruction-tuned model (e.g., Llama 3.3 70B / GPT-OSS 120B) for conversational tutoring, essay feedback, and personalized study-plan generation. Core features include a Retrieval-Augmented Q&A engine grounded in the student's own materials, an AI tutor chatbot, an assignment/essay evaluation module, automatic quiz generation, a progress-tracking dashboard, secure login, and chat history. The combination of NLP, OCR, vector retrieval, and generative AI in AcademiAI demonstrates how multimodal, retrieval-grounded LLM systems can make academic support more personalized, scalable, and accessible.

Keywords — Generative AI, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Optical Character Recognition (OCR), Natural Language Processing (NLP), Vector Database, Academic Chatbot, Automated Essay Evaluation, Personalized Learning, Artificial Intelligence.

I. INTRODUCTION

Large language models, in particular, have moved well beyond simple text generation. Survey work on LLM agents in education shows that, unlike rule-based tutoring systems, these agents can deliver interactive, personalized feedback and adapt to a learner's pace without needing constant teacher involvement, though they still struggle with more complex tasks such as evaluating professional-level academic writing [2]. A broader survey on LLMs for education further points out that these models are increasingly expected to support multilingual and non-English-speaking learners, not just English-medium classrooms [3].

This shift shows up across different corners of educational technology. An academic query assistant built on a microservices architecture integrated an LLM API to answer student information-retrieval questions, treating model selection as an engineering trade-off between cost, latency, and capability rather than assuming one "best" model [4]. Retrieval-Augmented didactic chatbots have similarly been built for higher-education settings, grounding LLM answers in course-specific material instead of the model's own pretrained memory, and have reported close to 85% accuracy on course-material questions [5]. Taken together, these efforts point toward the same conclusion: the next generation of academic AI tools needs to reason over a student's own material, not just general pretrained knowledge.

AcademiAI is built in response to this need. It is designed as a full-stack, Flask-based academic assistant that brings together study-material ingestion, a Retrieval-Augmented question-answering engine grounded in the

student's own materials, an AI tutor chatbot, and an assignment/essay evaluation module, all within a single accessible interface. Rather than treating document understanding, doubt-resolution, and assessment as separate problems, AcademiAI is designed around the same principle emerging in the literature above: that meaningful academic support comes from grounding generative AI in the learner's actual material and context [1], [4].

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

II. LITERATURE REVIEW

A knowledge-based LLM reading assistant [6] combined a fine-tuned GPT-4 model with retrieval-augmented techniques to help readers digest academic literature faster, reporting notably better accuracy and lower latency than a plain RAG baseline. While built for research-paper comprehension rather than classroom learning, the underlying retrieval-plus-generation pipeline is directly transferable to student-facing study-material assistants.

An e-learning conference study [7] developed a RAG-based chatbot for university students to answer frequently asked questions, grounding responses in institutional documentation rather than the LLM's own memory. The system demonstrated that combining retrieval with generation meaningfully reduces the kind of vague or generic answers that plain chatbots tend to give, though its knowledge base was limited to a single institution's FAQ material.

Sun et al. [8] introduced an ontology-grounded RAG chatbot for a graduate-level cybersecurity course, adding a domain-specific reasoning layer that constrains and verifies LLM-generated answers before they reach the student. Deployed with over a hundred active student users, the system shows that combining retrieval with a structured knowledge layer can meaningfully reduce hallucinated or unsafe responses in technical subjects.

A university-resource chatbot built on Retrieval-Augmented Generation [9] showed how retrieving the top-k relevant passages from an institutional knowledge base before generation improves both the relevance and accuracy of chatbot answers compared to relying on the LLM's parametric memory alone, reinforcing the importance of retrieval quality over model size alone.

Latif and Zhai [10] evaluated large language models on automated essay grading, testing multiple prompt-engineering strategies and reporting substantial agreement between model-assigned scores and human raters after prompt tuning — while also noting that grading consistency still varies noticeably across essay quality bands and prompt design choices.

***TABLE 2.1** Key Finding of the literature Survey*

Papers	Approach	Dataset Used	Technologies used	Key Result	Limitations
[11] Hasan et al. (2024)	Fine-tuned generative LLM for automatic question–answer generation with instructor-controlled style	RACE dataset	Meta-Llama 2-7B, prompt engineering.	Generated diverse MCQ, conceptual, and factual questions, reducing instructor workload	Focused only on English; not grounded in a specific course's own material
[12] Scaria et al. (2024))	Proposed Evaluated LLMs for generating questions at different Bloom's taxonomy skill levels	Educational text passages	ConvolFive state-of-the-art LLMs, advanced prompting.	High-quality lower-order questions; higher-cognitive questions harder to generate reliably	PQuality of higher-order questions still depends heavily on prompt design
[13] EvalQuiz (2024)	LLM-based automatic generation of self-assessment quizzes from lecture material.	Software engineering lecture slides.	LLM-based quiz generation pipeline	Saved lecturer preparation time; quizzes covered lecture content well	Generated questions sometimes lacked originality and variety.
[14] Educational PLPP (2024))	Prompt-engineered LLMs for personalized learning-path planning	Learner profile / behavior data	GPT-4, Llama-2-70B, prompt engineering	Significant gains in path quality and learner satisfaction vs. baseline	Relies on prompt-only adaptation; no live retraining on student data

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

[15] Second-Classroom LPR (2025	earning-path recommendation using multimodal learner signals.	Student behavioral/physiological data	GPT-4 feature extraction, incremental learning	mproved recommendation accuracy for long-sequence student data	mproved recommendation accuracy for long-sequence student data

From the above literature, existing systems generally address either document-grounded question answering, conversational tutoring, or essay evaluation — but rarely all three within one platform. Retrieval-grounded LLM systems have proven effective at reducing hallucination and improving answer relevance, yet most are limited to a single subject, a single document type, or a single institution's FAQ data. Automated feedback on assignments and essays remains largely a standalone research effort, disconnected from the retrieval and tutoring pipeline. There is a clear need for a unified, secure, and student-friendly platform that combines material-grounded Q&A, conversational tutoring, and assignment feedback into a single academic assistance framework.

III. System Architecture

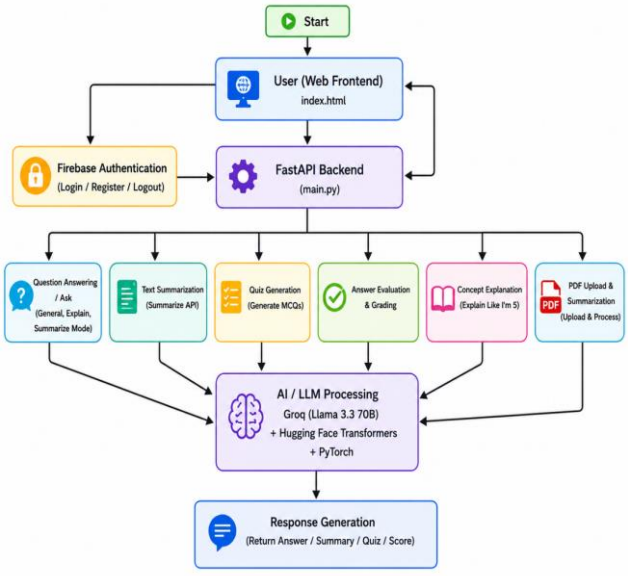

Fig 3.1 System Architecture

The proposed AcademiAI system employs a three-tier architecture: Frontend, Flask Backend, and Database, integrated with AI models for study-material understanding. The frontend, built using HTML, CSS, and JavaScript, allows users to register, log in, upload study material (notes, textbooks, syllabi, assignments), ask questions grounded in their own material, request essay/assignment feedback, and view their personalized study dashboard. The frontend communicates with the Flask backend through RESTful HTTP requests.

The Flask backend acts as the application controller, managing user authentication, file uploads, document parsing, the RAG pipeline, chatbot interaction, essay evaluation, and all database operations. Uploaded PDFs are parsed using PyMuPDF, DOCX files using python-docx, PPTX slides using python-pptx, and scanned or handwritten notes using Tesseract OCR. Extracted text is chunked and converted into embeddings, which are stored in a vector database (e.g., FAISS/Chroma) to enable semantic retrieval. When a student asks a question, the backend retrieves the most relevant chunks from their own uploaded material and passes them, along with the query, to the Groq API, where the selected large language model generates a grounded, context-aware answer rather than a generic response. The same LLM pipeline powers the assignment/essay evaluation module and the personalized study-plan generator.

User data, uploaded material, chat history, saved responses, and evaluation records are stored in an SQLite database. Finally, the backend replies to the frontend with grounded answers, essay feedback, generated quizzes, study-plan recommendations, and dashboard visualizations that let the student track their academic progress over time.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

IV. METHODOLOGY

Below are the Modules of MediAI project:

4.1 User Authentication Module

Protects the framework through secure registration and login. New users provide name, email, and password; returning users authenticate with email and password. Credentials are securely hashed, and sessions are managed and persisted in the SQLite database.

4.2 Study Material Upload and Processing Module

Allows students to upload textbooks, lecture notes, syllabi, and assignments in PDF, DOCX, or PPTX format, as well as scanned or handwritten notes as images. PDFs are processed with PyMuPDF, DOCX files with python-docx, PPTX files with python-pptx, and scanned/handwritten content with Tesseract OCR.

4.3Retrieval-Augmented Question Answering (RAG) Module

Extracted material is chunked and embedded into a vector database. When a student asks a question, the module retrieves the most semantically relevant chunks from their own uploaded material and feeds them to the LLM, ensuring answers are grounded in what was actually taught rather than the model's generic pretrained knowledge

4.4AI Academic Assistant (Tutor Chatbot) Module

The conversational core of the system, connected via the Groq API to an instruction-tuned LLM. It simplifies difficult concepts, answers follow-up questions, supports multilingual interaction, and maintains conversational context across a chat session.

4.5Assignment and Essay Evaluation Module

Accepts student-written assignments or essays and uses the LLM to generate structured feedback — covering clarity, grammar, argument coherence, and content accuracy — along with an indicative score, drawing on prompt-engineering strategies shown effective in recent essay-scoring research [9].

4.6 Personalized Study Plan and Quiz Generation Module

Analyzes a student's uploaded syllabus and topic-wise performance to generate a personalized study schedule and auto-generated practice quizzes targeting weaker topics.

4.7 Progress Dashboard Module

Visualizes topic-wise performance, quiz scores, and study-plan completion using interactive charts (e.g., Chart.js), helping students identify weak areas at a glance.

4.8 Database Management Module

Handles secure storage of user accounts, uploaded material, chat history, saved AI responses, evaluation records, and study-plan data in SQLite, enabling users to revisit past sessions.

4.9 User Interface Module

A responsive interface built with HTML, CSS, and JavaScript supporting registration, login, material upload, chatbot interaction, essay submission, and dashboard viewing in a readable, accessible layout.

V. IMPLEMENTATION

Step 1: Requirement Analysis

Core requirements identified: document-grounded Q&A, AI tutoring, essay feedback, and personalized study planning. Tools, technologies, and overall workflow were finalized before development.

Step 2: Knowledge Base and Embedding Pipeline Setup

Sample syllabi, textbooks, and notes were collected to design and test the chunking and embedding pipeline, and a vector database was configured for semantic retrieval.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

Step 3: Frontend Development

A responsive interface was built using HTML, CSS, and JavaScript, with pages for registration/login, material upload, chatbot interaction, essay submission, and dashboard display.

Step 4: Backend Development

Built using Python and Flask, managing authentication, file uploads, document parsing, RAG retrieval, and chatbot/essay-evaluation logic, modularized for maintainability.

Step 5: RAG Pipeline Integration

Extracted text from uploaded material is chunked, embedded, and indexed; retrieved chunks are passed to the Groq API alongside user queries for grounded response generation.

Step 6: Database Implementation

SQLite was chosen to store user credentials, uploaded material metadata, chat history, and evaluation records.

Step 7: API Integration

The Groq API was integrated for both the tutoring chatbot and the essay-evaluation module, with API keys managed via environment variables.

Step 8: System Testing

The complete system was tested across authentication, document upload, retrieval accuracy, chatbot responses, essay feedback, and dashboard rendering under varied and invalid inputs.

Step 9: Deployment and Evaluation

The integrated system was deployed locally and assessed for retrieval accuracy, response quality, usability, and performance.

VI. MODEL EVALUATION AND TECHNOLOGY SELECTION

Table 6.1 Comparative Evaluation of AI Models and Technologies

Component	Models / Technologies Evaluated	Selected Model / Technology	Reason for Selection
Document Text Extraction	EasyOCR, Rule-based extraction, Tesseract OCR	Tesseract OCR	Reliable extraction from scanned notes and handwritten material with easy Python integration
Embedding / Retrieval	TF-IDF, Sentence-BERT, OpenAI embeddings	Sentence-BERT-based embeddings + FAISS	Strong balance of retrieval accuracy, speed, and low resource cost for local deployment
Question Answering (RAG)	GPT-3.5, Gemma, Qwen, Llama 3.3 70B	Llama 3.3 70B (via Groq API)	Strong contextual grounding, low-latency inference, and reliable multi-turn reasoning
AI Tutor / Conversational Assistan	Smaller instruction-tuned LLMs, GPT-OSS 120B	GPT-OSS 120B (via Groq API)	More natural conversation, stronger reasoning, and better context retention for tutoring dialogue
Essay / Assignment Evaluation	GPT-3.5, GPT-4, fine-tuned smaller LLMs	GPT-4-class model with prompt engineering	Higher agreement with human raters on scoring and feedback quality, per recent AES research [9]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

LLM Inference Platform	Local Inference, Groq API,OpenAI API	Groq API	EHigh-speed, low-latency inference improving chatbot and evaluation response times
PDF Processing	Generic PDF Libraries, PyMuPDF	PyMuPDF	Fast, accurate text extraction with minimal preprocessing
DOCX Processing	Manual Parsing, python-docx	python-docx	Efficiently extracted structured content from Word documents and integrated easily with the application workflow.

Table 6.1 presents the comparison of models and technologies considered during development. Each was evaluated on accuracy, response quality, latency, and ease of integration with the proposed system.

VII. RESULTS AND DISCUSSION

The proposed AcademiAI framework was evaluated by testing each module independently with different study materials, essays, and user queries, verifying that the system correctly processes multiple input formats, retrieves relevant material, and generates meaningful academic support.

7.1 User Authentication

Registration and login functioned securely; sessions and credentials were correctly encrypted and persisted.

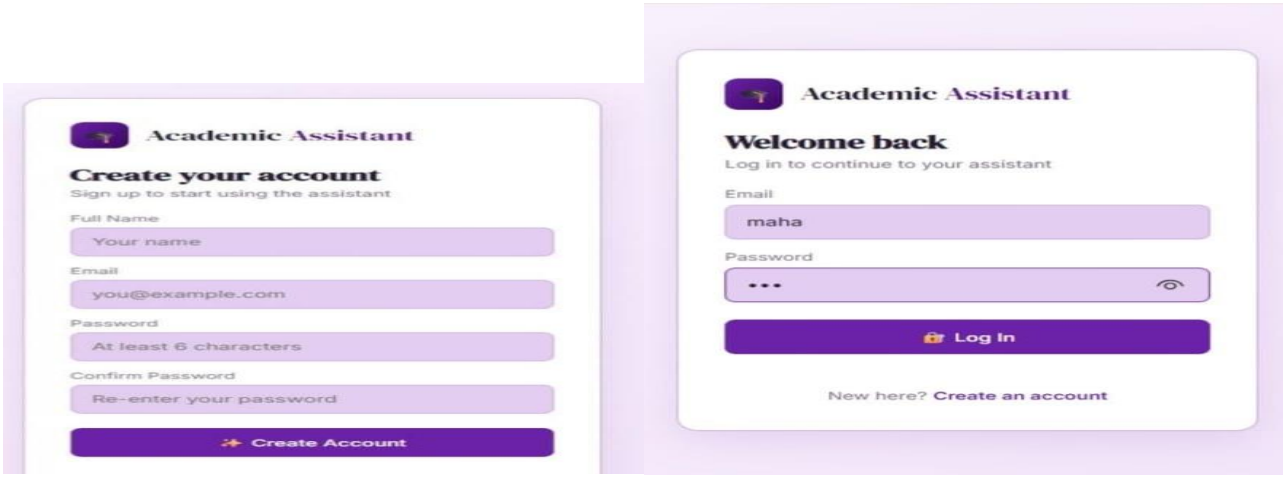

Fig. 7.1. User Registration and Login Interface

7.2 Study Material Processing

Evaluated using PDF, DOCX, PPTX, and scanned/handwritten notes. Digital files were processed via PyMuPDF/python-docx/python-pptx; scanned and handwritten notes were converted via Tesseract OCR and successfully indexed for retrieval.

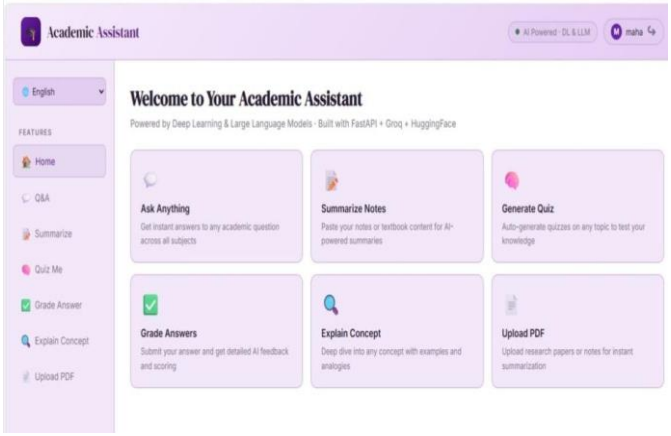

Fig 7.2. Study material Upload and Processing

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

7.3Retrieval-Augmented Question Answering

Tested with questions spanning multiple uploaded documents. The system retrieved contextually relevant chunks and generated grounded answers rather than generic responses, closely matching source material.

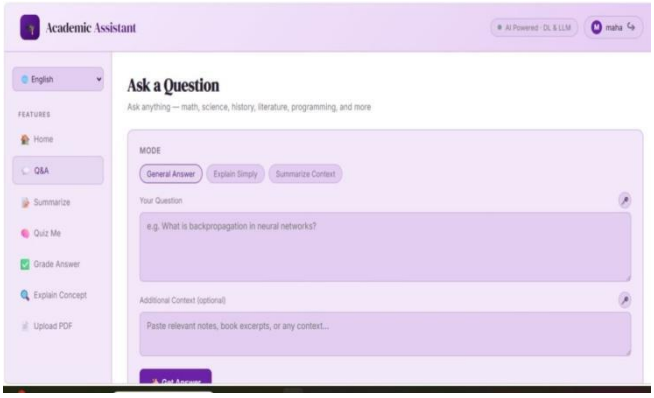

Fig. 7.3.RAG-based Question Answering Interface

7.4 AI Academic Assistant (Tutor Chatbot)

Evaluated with varied academic questions across subjects. The chatbot explained concepts clearly, maintained conversational context, and supported multilingual queries.

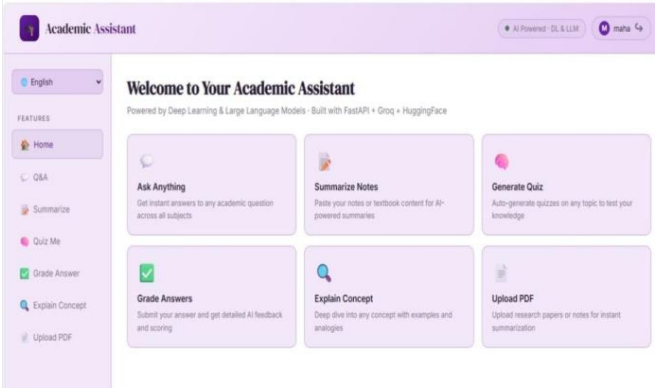

Fig. 7.4. AI Academic Assistant Interface

7.5 Assignment and Essay Evaluation

Tested with sample essays of varying quality. The module generated structured, consistent feedback on grammar, coherence, and content, broadly aligned with the score-agreement trends reported in recent AES literature [9]. and the AI assistant, making the system more inclusive and user-friendly.

7.6 Lab Report Dashboard

The laboratory dashboard successfully extracted laboratory parameters from uploaded reports and presented them using interactive charts. The visualization enabled users to quickly identify abnormal test values and compare different health parameters without manually reading the entire report.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

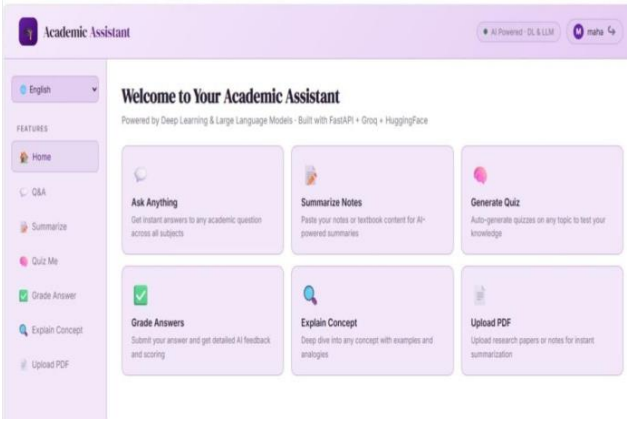

Fig. 7.6. Dashboard Visualization

7.7 Discussion

The evaluation demonstrates that AcademiAI effectively combines document processing, OCR, retrieval-augmented generation, and large language models into a single academic support platform. Unlike conventional tools that focus on either chatbot tutoring or essay grading in isolation, AcademiAI grounds its responses in the student's own uploaded material, reducing generic or irrelevant answers. The integration of personalized study planning and progress tracking further distinguishes it from single-purpose academic chatbots, confirming the framework's value as an efficient, secure, and student-friendly academic assistance solution.

VIII. CONCLUSION

AcademiAI demonstrates the integration of Artificial Intelligence, retrieval-augmented generation, and web technology to make academic support more personalized and accessible. Students can upload study material in multiple formats, ask questions grounded in their own coursework, receive structured feedback on assignments, generate personalized study plans and quizzes, and track their progress — all from a single interface. Through a retrieval-augmented pipeline combined with large language models accessed via the Groq API, AcademiAI aims to reduce the gap between generic AI chatbots and genuinely useful, material-grounded academic assistance. The architecture consists of User Authentication, Study Material Processing, Retrieval-Augmented Question Answering, AI Academic Assistant, Assignment/Essay Evaluation, Study Plan and Quiz Generation, Progress Dashboard, Database Management, and User Interface modules.

The application is built on a Flask, SQLite, HTML, CSS, and JavaScript stack, providing a secure, maintainable, and extensible environment. AcademiAI is beneficial to students seeking on-demand, grounded academic help; educators who can use it as a supplementary teaching-support tool; and institutions aiming to scale personalized academic assistance without proportionally scaling staff. Future work can include integration with Learning Management Systems (LMS), support for voice-based interaction, deeper analytics on learning patterns, and cloud deployment for institution-wide access.

REFERENCES

[1] VC. Feng et al., "Improving Collaborative Learning Performance Based on LLM Virtual Assistant," *Proc. IEEE Int. Conf.*, 2024

[2] [Author names], "PaperHelper: Knowledge-Based LLM QA Paper Reading Assistant," arXiv:2502.14271, 2025.

[3] [Author names], "LLM Agents for Education: Advances and Applications," arXiv:2503.11733, 2025.

[4] pAuthor names], "Academic Query Assistant: Integrating LLM API into an Academic Assistant Using a Microservices Architecture," *J. Computing Science and Engineering*, vol. 18, no. 2, pp. 125–133, 2024.

[5] [Author names], "Retrieval-Augmented Chatbots for Scalable Educational Support in Higher Education," 2025

[6] [Author names], "Retrieval Augmented Generation in Large Language Models," *Proc. 15th Int. Conf. e-Learning*, Belgrade, Serbia, 2024

[7] F. Xing, B. Meyer, M. Harandi, T. Drummond, and Z. Ge, [medical vision task], IEEE Access, vol. 11, pp. 50165–50179, 2023, doi: 10.1109/ACCESS.2023.3278376.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15849

[8] Z. Sun et al., "CyberBOT: Towards Reliable Cybersecurity Education via Ontology-Grounded Retrieval Augmented Generation," arXiv:2504.00389, 2025.

[9] E. Latif and X. Zhai, "On Automated Essay Grading Using Large Language Models," *Proc. 8th Int. Conf. Computer Science and Artificial Intelligence (ICCSAI)*, 2024.

[10] A. S. M. M. Hasan, M. A. Ehsan, K. B. Shahnoor, and S. S. Tasneem, "Automatic Question & Answer Generation Using Generative Large Language Model (LLM)," B.Sc. thesis, Brac University, 2024, arXiv:2508.19475.

[11] N. Scaria, S. Dharani Chenna, and D. Subramani, "Automated Educational Question Generation at Different Bloom's Skill Levels Using Large Language Models: Strategies and Evaluation," arXiv:2408.04394, 2024

[12] Author names], "EvalQuiz — LLM-based Automated Generation of Self-Assessment Quizzes in Software Engineering Education," *Proc. GI*, 2024.

[13] [Author names], "Educational Personalized Learning Path Planning with Large Language Models," arXiv:2407.11773, 2024.

[14] [Author names], "A Second-Classroom Personalized Learning Path Recommendation System Based on Large Language Model Technology," *Applied Sciences (MDPI)*, vol. 15, no. 14, p. 7655, 2025.

[15] [Author names], "Large Language Models for Education: A Survey andOutlook," arXiv:2403.18105, 2024.