



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

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



Impact Factor: 9.935

Volume 15, Issue 10, October 2026

VidyaAI: A Personalized AI Learning Assistant Using Hybrid Retrieval-Augmented Generation for Syllabus-Aligned Question Answering

Basantha Kumari, Chinmayi M K, Dilip M P, Chidananda G, Abhinaik N

Associate Professor, Department of Computer Science and Engineering, S J M Institute of Technology, Chitradurga,
Karnataka, India¹

U.G. Student, Department of Computer Science and Engineering, S J M Institute of Technology, Chitradurga,
Karnataka, India²⁻⁵

ABSTRACT: Large Language Models (LLMs) can answer academic questions, but general-purpose chatbots respond from knowledge learned during pre-training, so their explanations often differ from a student's prescribed syllabus, lecture notes or textbooks. This paper presents VidyaAI, a web-based personalized learning assistant that generates answers directly from study materials uploaded by the student. The system accepts PDF, DOCX and image files, extracts text (using Tesseract OCR for scanned content), divides it into overlapping chunks and stores Sentence-Transformer embeddings in a custom subject-wise SQLite vector database. At query time, a hybrid retrieval pipeline combines semantic search with BM25 keyword ranking, merges the results using Reciprocal Rank Fusion (RRF) and reorders them with a BGE cross-encoder reranker before the retrieved context is passed to a Groq-hosted LLM. The platform adds secure authentication, persistent conversation history, multilingual responses and several answer formats (detailed, simplified, essay, MCQ and code). The frontend is built with React and the backend with FastAPI. Unit, integration, functional, retrieval, database, security and usability testing showed that all implemented modules behaved as specified. VidyaAI demonstrates that a lightweight, open-source RAG stack can give engineering students reliable, document-grounded and syllabus-oriented academic assistance.

KEYWORDS: Retrieval-Augmented Generation, Hybrid Retrieval, BM25, Reciprocal Rank Fusion, Cross-Encoder Reranking, Sentence Transformers, Large Language Models, Personalized Learning, Educational AI.

I. INTRODUCTION

Artificial Intelligence (AI) has transformed healthcare, finance, manufacturing and education. In education, Large Language Models (LLMs) now provide question answering, content generation and personalized explanations on demand. Engineering students, however, study from many different sources: textbooks, lecture notes, laboratory manuals, previous examination papers and handwritten notes. Searching through these resources to find a relevant passage is time-consuming, and it reduces learning efficiency, especially during examination preparation.

General-purpose AI assistants answer from knowledge learned during pre-training. Their replies may use terminology different from the classroom, include concepts beyond the prescribed syllabus, or contain statements that are simply incorrect. Retrieval-Augmented Generation (RAG) addresses this limitation by retrieving relevant passages from an external knowledge base before the language model writes its answer, so that the response is grounded in source documents and hallucination is reduced [1]. Many existing RAG implementations, however, are research prototypes or developer tools. They often lack features that a learner needs, such as secure user accounts, persistent conversation history, support for scanned notes, multilingual interaction and flexible answer formats.

This paper presents **VidyaAI**, an AI-powered personalized learning assistant. A student uploads academic documents in PDF, DOCX or image format; the system extracts the text (using OCR for scanned content), splits it into overlapping chunks, converts the chunks into vector embeddings and stores them in a subject-wise SQLite vector database. When the student asks a question, a hybrid retrieval pipeline combining semantic search and BM25 keyword search, followed by Reciprocal Rank Fusion (RRF) and BGE cross-encoder reranking, selects the most relevant chunks. These are

supplied as context to a Groq-hosted LLM that produces a syllabus-aligned answer in the format and language the student prefers.

The main contributions of this work are:

- A personalized, document-grounded learning assistant based on RAG, with secure authentication, profile management and persistent chat history.
- A custom SQLite-based vector database that keeps embeddings in separate subject collections and removes the need for an external vector-database service.
- A multi-stage hybrid retrieval pipeline that combines semantic search and BM25, fuses the results with RRF and reranks them with a BGE cross-encoder.
- OCR-based ingestion of scanned notes and images, automatic language detection and multilingual response generation.
- Several educational output formats (detailed explanation, simplified explanation, essay, multiple-choice questions, source-based answer and code assistance) within one platform.

The paper is organized as follows. Section II reviews related work and identifies the research gap. Section III states the problem and objectives. Section IV describes the requirements and feasibility of the system. Section V presents the system architecture and design. Section VI explains the proposed methodology. Section VII describes the implementation. Section VIII reports testing and results, Section IX discusses advantages and limitations, and Section X concludes the paper with future scope.

II. LITERATURE SURVEY

Lewis et al. [1] introduced the Retrieval-Augmented Generation framework, which combines a neural retriever with a sequence-to-sequence generator. Instead of relying only on parameters learned during training, the model retrieves relevant documents from an external knowledge base and conditions its answer on them. The authors showed that this approach improves performance on knowledge-intensive tasks and reduces hallucination. The same principle forms the core of the VidyaAI question-answering pipeline, where the knowledge base is the set of documents uploaded by the student rather than general internet knowledge.

Reimers and Gurevych [2] proposed Sentence-BERT (SBERT), which adapts the BERT architecture [6], itself based on the transformer [8], to produce semantically meaningful sentence embeddings that can be compared efficiently using cosine similarity. This makes meaning-based document retrieval practical, because a query and a passage can be matched even when they share few words. VidyaAI uses Sentence-Transformer embeddings to place document chunks and user queries in the same vector space.

Sparse retrieval remains valuable for exact terms. The BM25 ranking function [3] scores documents using term frequency, inverse document frequency and document length normalization, and it handles specific keywords, symbols and names very well. Dense and sparse retrieval are complementary: dense retrieval captures paraphrases and concepts, while sparse retrieval captures precise vocabulary. Combining them gives higher retrieval accuracy than either alone. Cormack et al. [4] showed that Reciprocal Rank Fusion, which merges ranked lists using only the rank positions of documents, outperforms several more complex rank-learning and fusion methods. A cross-encoder reranker can then re-score a short list of candidates by reading the query and each passage together, giving a finer relevance judgment than comparing independently computed embeddings.

VanLehn [5] analyzed the relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems, and found that well-designed intelligent tutoring systems can approach the effectiveness of human tutors and do better than instruction without such support. This evidence supports the development of AI-driven learning platforms that give personalized feedback and continuous assistance. Large language models such as GPT-3 [7] can produce fluent and context-conditioned answers from a few examples or instructions, but the quality and reliability of their answers depend on the context they are given, which again motivates retrieval-based grounding.

A. Research gap

The surveyed literature shows that the individual techniques are mature. The following gaps remain in practical educational systems:

- Most AI chatbots rely on pre-trained knowledge instead of institution-specific study materials.
- Few learning tools combine semantic and keyword retrieval in one pipeline.
- Secure authentication and persistent conversation history are rarely combined with document-grounded retrieval.
- Multilingual support together with personalized document retrieval is uncommon.
- Many systems depend on heavy external vector-database services and do not exploit lightweight local storage for embeddings.

VidyaAI is designed to address these gaps in a single integrated platform.

III. PROBLEM STATEMENT AND OBJECTIVES

Students often receive answers from AI assistants that differ from their university syllabus, lecture notes or recommended textbooks, which leads to confusion during examination preparation. Existing educational platforms generally provide either static learning resources or generic question answering, without using the individual study material uploaded by the learner. Students also use heterogeneous resources such as scanned notes, PDF textbooks, Word documents and images, and ordinary chatbots cannot extract, organize and retrieve knowledge from them consistently. The core problem is therefore to design and develop an AI-powered personalized learning assistant that enables students to upload their academic resources, retrieves relevant information using hybrid Retrieval-Augmented Generation techniques, and generates accurate, syllabus-oriented, multilingual responses while maintaining secure user authentication and persistent learning history.

A. Objectives

The specific objectives of the project are:

- To build a RAG-based learning assistant that answers academic questions using user-uploaded study materials.
- To implement secure registration, login, session handling and profile management.
- To store conversations so that users can revisit and continue earlier learning sessions.
- To support PDF, DOCX, JPG, JPEG and PNG files for knowledge-base creation and to extract text from scanned documents using OCR.
- To generate contextual embeddings with multilingual Sentence-Transformer models and store them in a custom subject-wise SQLite vector database.
- To combine semantic search and BM25 ranking, and to improve results with RRF and BGE cross-encoder reranking.
- To integrate the Groq LLM API for context-aware, syllabus-oriented response generation with minimal hallucination.
- To support automatic language detection, multilingual answers and several answer formats.
- To provide a responsive web interface that works across devices.

B. Scope

The system includes secure authentication, profile management, document upload and extraction, OCR, language detection, subject-wise knowledge bases, hybrid retrieval, LLM-based answering, multiple response formats, source-aware answers and local storage of embeddings, conversations and credentials in SQLite. The current implementation deliberately excludes real-time internet search, cloud-storage synchronization, native Android or iOS applications, video lecture and audio analysis, handwritten equation recognition beyond OCR, collaborative multi-user workspaces, automatic grading and performance analytics, integration with institutional Learning Management Systems (LMS) and real-time online classroom features.

IV. REQUIREMENT ANALYSIS

Requirement analysis identifies the functional and non-functional requirements that the system must satisfy, together with the hardware and software needed and the feasibility of the project.

A. Functional requirements

The system is organized into the following functional modules. The authentication module allows registration, secure login, session maintenance, profile updates and logout. The document upload module accepts PDF, DOCX, JPG, JPEG and PNG files, validates them and organizes them by subject. The document processing module extracts text from PDF and DOCX files, performs OCR on images, detects the language, divides text into overlapping chunks and generates

embeddings. The knowledge base module stores embeddings subject-wise in the SQLite vector database, maintains metadata for every document and allows new documents to be added without affecting existing data. The question answering module accepts natural language questions, performs semantic and BM25 retrieval, merges results with RRF, reranks with the BGE cross-encoder and generates the answer with the Groq LLM. The learning support module produces detailed explanations, simplified explanations, essays, multiple-choice questions and code. The conversation history module stores, searches and reopens earlier chats, and the multilingual module detects the query language and answers in the supported language while keeping the formatting consistent. The frontend communicates with the backend through REST APIs, and the backend validates every request, returns structured JSON and handles exceptions gracefully.

B. Non-functional requirements

The system should respond within a few seconds under normal conditions (performance); keep uploaded documents and conversations safe and consistent (reliability); store passwords as hashes, validate authentication tokens and prevent one user from accessing another user's data (security); support multiple users, large document collections and additional AI models (scalability); remain available and recover gracefully from temporary service interruptions (availability); offer simple navigation, easy upload and a minimal learning curve (usability); follow a modular architecture with reusable components and independent services (maintainability); and run on Windows and Linux in modern web browsers (compatibility).

C. Hardware and software requirements

TABLE I. HARDWARE AND SOFTWARE REQUIREMENTS

Component	Specification
Processor	Intel Core i5 or higher
RAM	Minimum 8 GB (16 GB recommended)
Storage	10 GB free space
Internet	Required for LLM inference and updates
Display	Minimum 1366 × 768 resolution
Operating system	Windows 10/11 (Linux also supported)
Frontend	React.js, Vite
Backend and languages	FastAPI; Python 3.10+, JavaScript
Databases	SQLite; custom SQLite vector database
AI and NLP components	Groq LLM, Sentence Transformers, Tesseract OCR
Tools	Visual Studio Code, Git and GitHub

D. Feasibility study

Technical feasibility: the project uses mature, well-documented open-source technologies (React, FastAPI, SQLite, Sentence Transformers and the Groq API), so implementation is technically feasible. Economic feasibility: free and open-source software is used wherever possible, SQLite is the local database and the AI service offers usage tiers suitable for academic projects, so the cost is minimal. Operational feasibility: the interface lets students upload materials, ask questions and receive answers without technical expertise. Schedule feasibility: the modular architecture allows analysis, design, implementation, testing and documentation to proceed in phases within the academic timeline.

E. System constraints

Answer quality depends on the quality and relevance of the uploaded documents; internet connectivity is required to reach the Groq LLM; very large document collections may increase indexing and retrieval time; OCR accuracy varies with image quality; and overall performance depends on the available hardware and network conditions.

V. SYSTEM DESIGN AND ARCHITECTURE

VidyaAI follows a modular client–server architecture that integrates web technologies, Natural Language Processing and hybrid information retrieval. The system has six layers: the presentation layer (React frontend), the application layer (FastAPI REST services), the authentication layer, the document processing layer, the hybrid retrieval layer and

the AI response generation layer. Modules communicate through REST APIs returning structured JSON, which keeps them loosely coupled and easy to maintain. Fig. 1 shows the overall architecture.

Two workflows run on this architecture. In the document ingestion workflow, an authenticated user uploads a file; the text is extracted and cleaned, divided into overlapping chunks, embedded and stored in the vector database under the selected subject. In the question answering workflow, the user's query is retrieved against the selected subject collection, the top chunks are placed into a prompt, and the LLM's answer is returned to the user and saved to the chat history.

A. Module design

The user authentication module handles registration, login, logout, profile and session management so that only authenticated users can reach personal resources. The document upload module uploads PDF, DOCX and image files, validates them and records the subject. The document processing module performs PDF and DOCX parsing, OCR, language detection and text cleaning. The knowledge base module stores subject-wise vectors, metadata and collections, which prevents retrieval across unrelated subjects. The hybrid retrieval module performs semantic search, BM25 retrieval, RRF and BGE reranking. The AI response module constructs prompts, injects context, calls the LLM and formats the answer. The conversation history module stores, retrieves, continues and deletes chats.

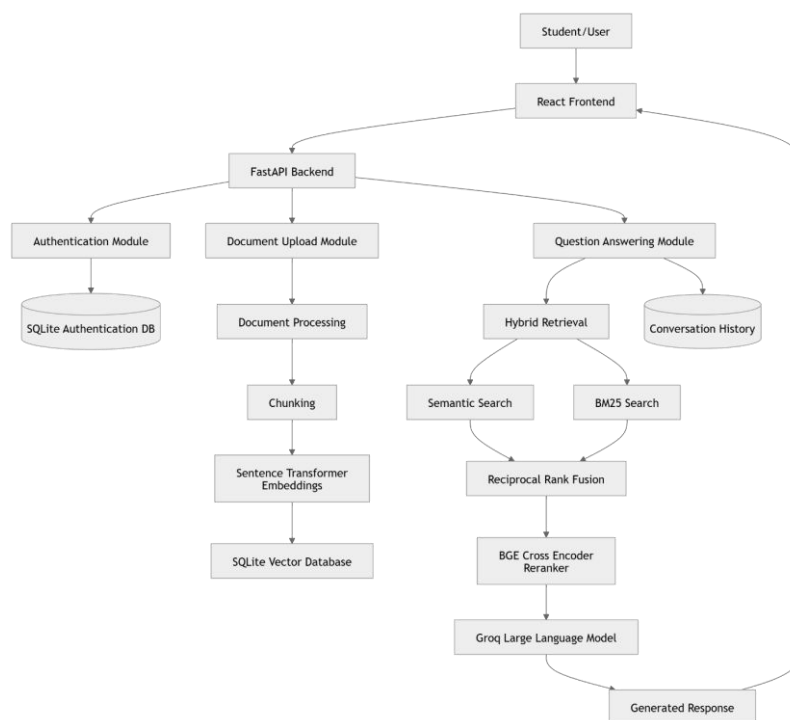


Fig.1. Overall system architecture of VidyaAI

B. Use case and retrieval pipeline

The primary actor is the student, who can register, log in, upload study material, ask questions, view chat history, update the profile and log out. Fig. 2(a) shows the use case diagram and Fig. 2(b) shows the hybrid retrieval pipeline that supplies context to the language model.

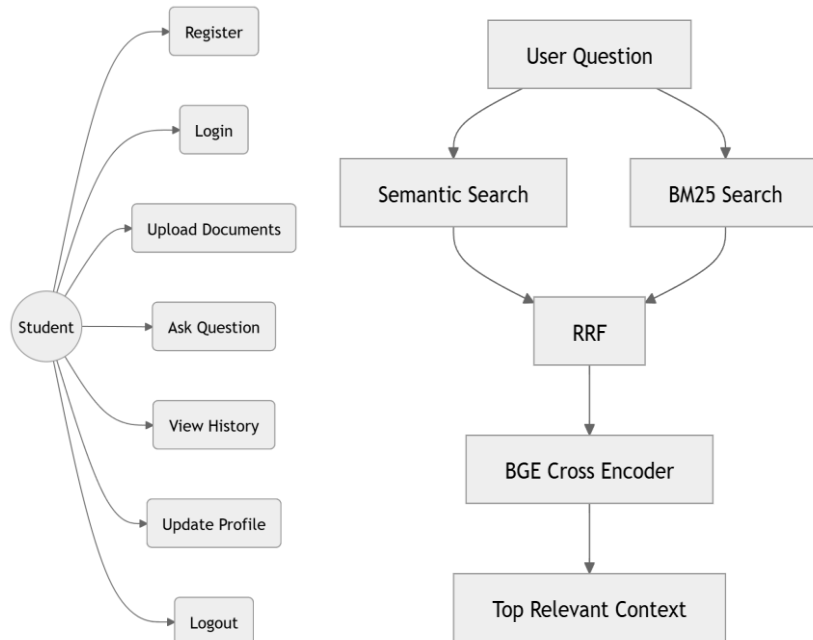


Fig. 2. (a) Use case diagram (b) Hybrid retrieval pipeline

C. Database design

Two SQLite databases are used. The authentication database stores users, sessions and conversation history. The vector database stores subject collections, document metadata, text chunks and embeddings. The main entities (Fig. 3) are USER, PROFILE, SUBJECT, DOCUMENT, DOCUMENT_CHUNK, CHAT_SESSION and CHAT_HISTORY. A user has one profile and can create many subjects and many chat sessions; a subject contains many documents; a document is divided into many document chunks; and a chat session contains many chat history records.



Fig.3.Entity-relationship diagram of VidyaAI

VI. PROPOSED METHODOLOGY

The methodology follows a Retrieval-Augmented Generation architecture and consists of the stages: user authentication, document upload, document processing, text chunking, embedding generation, vector storage, hybrid retrieval, prompt construction, LLM inference, response generation and chat-history storage. Every response is grounded in the user's uploaded materials, which improves accuracy and reduces hallucination.

A. User authentication

Every user must register and log in before using the application. The authentication module verifies credentials and creates a secure session. Authenticated users can upload documents, create subject collections, ask questions, view previous conversations and manage their profile. Authentication also keeps each user's study materials and conversations private and isolated from other users.

B. Document upload and processing

Uploaded files (PDF, DOCX, JPG, JPEG, PNG) are validated before processing. Text is parsed directly from PDF and DOCX files, while image-based and scanned documents are converted to text using Tesseract OCR. The extracted text is cleaned of unnecessary whitespace, special symbols and formatting inconsistencies, and the document language is detected.

C. Text chunking and embedding generation

Large documents cannot be passed directly to embedding models because of token limits, so the cleaned text is divided into smaller overlapping chunks. Chunking preserves contextual information, improves retrieval accuracy, enables efficient vector generation and reduces computation. Each chunk carries metadata such as document name, subject and chunk number. Every chunk is then converted into a dense vector by a multilingual Sentence-Transformer model. Because the embedding captures meaning rather than exact wording, conceptually similar questions and passages can be matched even when their vocabulary differs.

D. SQLite vector database

The embeddings are stored in a custom SQLite-based vector database together with document information, subject collections, chunk identifiers and metadata. This lightweight design supports efficient semantic retrieval and removes the dependence on external vector-database services, which simplifies deployment for student and small-institution use.

E. Hybrid retrieval

When a question is submitted, retrieval within the selected subject runs in four stages:

- Semantic search: the query is embedded and the chunks with the most similar vectors are found, which matches conceptually related content even when the wording differs.
- BM25 keyword search: chunks containing the important terms of the query are ranked, which handles exact terms, names and symbols well.
- Reciprocal Rank Fusion: the two ranked lists are merged. A chunk d receives the score $RRF(d) = \sum 1 / (k + \text{rank}(d))$, summed over the lists in which it appears, where k is a smoothing constant. Chunks ranked well by both methods rise to the top without any score normalization [4].
- BGE cross-encoder reranking: the fused short list is re-scored by a cross-encoder that reads the query and each chunk together, and the best chunks are kept as context.

F. Prompt construction and response generation

The retrieved chunks and the user's question are assembled into a structured prompt. The prompt instructs the Groq-hosted LLM to answer only from the supplied academic content, in the requested format (detailed, simplified, essay, MCQ, source-based or code) and in the language of the user. The generated answer is formatted and returned to the user through the web interface. Providing relevant context to the model is what keeps answers aligned with the uploaded syllabus and limits hallucination. The sequence of calls for one question is shown in Fig. 4.

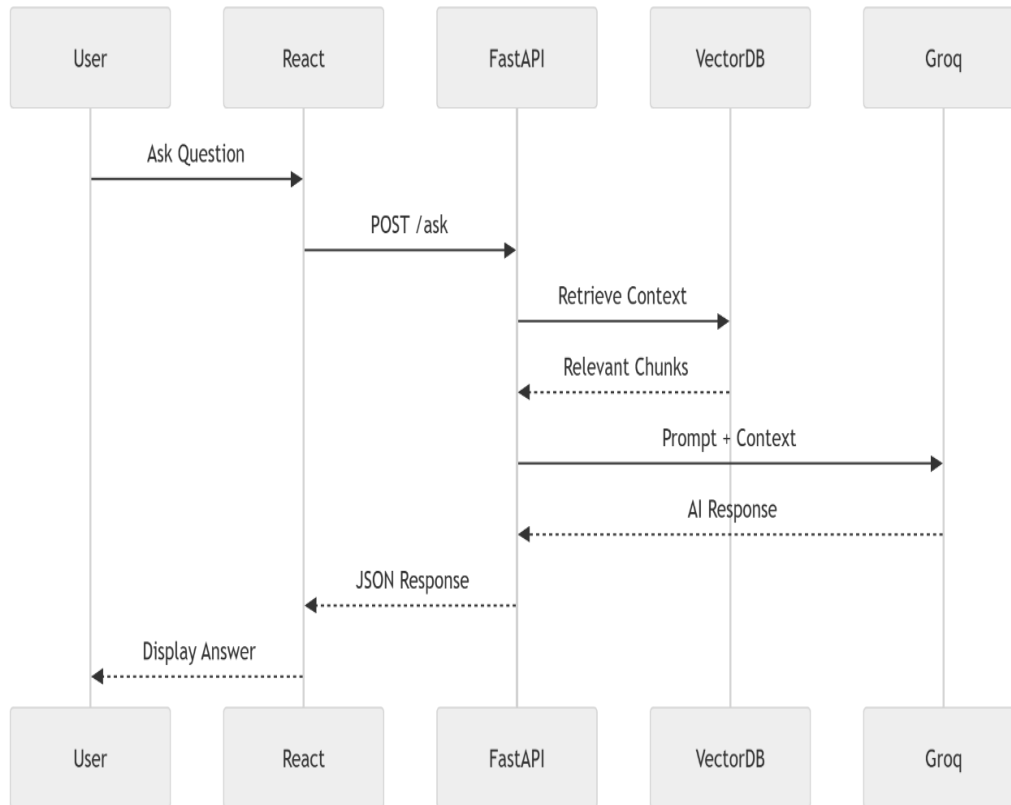


Fig.4. Sequence diagram of the question-answering flow

G. Conversation history management

Each question, the AI-generated answer, the session identifier and a timestamp are stored in the database. Users can therefore revisit earlier discussions and continue learning without repeating queries, which supports revision and long-term retention.

VII. IMPLEMENTATION

The proposed design was implemented as a modular web application using the technologies listed in Table II. Each module was developed independently and integrated through REST APIs to ensure maintainability and extensibility.

TABLE.II. DEVELOPMENT ENVIRONMENT

Component	Technology used
Programming languages	Python 3.10+, JavaScript
Frontend	React.js, Vite
Backend	FastAPI
Database	SQLite
Vector database	Custom SQLite vector database
AI model	Groq Large Language Model
Embedding model	Sentence Transformers
OCR engine	Tesseract OCR
Version control	Git and GitHub
IDE	Visual Studio Code

A. Frontend

The frontend uses React.js, whose component-based architecture simplifies the development of reusable user-interface elements. It provides the home page, registration, login, dashboard, subject management, document upload, AI chat interface, chat history and user profile screens. Each screen follows a consistent layout and responsive design principles, so the application works across different screen sizes, and the frontend communicates with the backend through RESTful APIs.

B. Backend

The backend is implemented with FastAPI, which provides high-performance asynchronous request handling and automatic API documentation. It is responsible for user authentication, file management, document processing, vector generation, hybrid retrieval, AI response generation and chat history management. User credentials are stored as password hashes in the SQLite authentication database, and every request is validated before it is processed.

C. Document processing and embeddings

After a document is uploaded, the system performs PDF text extraction, DOCX parsing, OCR for scanned images, text cleaning and metadata generation. The processed text is divided into overlapping chunks, and each chunk is converted into a dense vector by the Sentence-Transformer model. The vectors, chunk text and metadata are stored in the custom SQLite vector database, with a separate collection for every subject.

D. Retrieval and response generation

For each question the backend runs semantic vector search and BM25 retrieval, merges the results with RRF and reranks them with the BGE cross-encoder. The final context is combined with the question into a prompt and sent to the Groq API. The response is returned to the frontend as JSON, displayed to the user and saved to the conversation history.

VIII. TESTING AND RESULTS

Testing was performed on individual modules as well as on the integrated application. Its aim was to verify that all functions work as specified, that authentication and authorization are enforced, that documents are processed correctly, that retrieval is accurate and that the system is stable and usable.

A. Types of testing

Unit testing verified registration, login, document upload, OCR processing, text chunking, embedding generation, semantic search, BM25 retrieval, API endpoints and chat-history storage; each module produced the expected output. Integration testing validated the links between the React frontend and the FastAPI backend, the backend and both SQLite databases, the backend and the Groq API, the backend and the OCR engine, and hybrid retrieval and response generation. Functional testing confirmed registration, login, upload, subject creation, question answering, chat history, profile and logout. Performance testing observed upload time, text processing time, embedding time, retrieval time, AI response time and database query performance, and the application performed satisfactorily under normal operating conditions. Usability testing checked navigation, the responsive interface, simple document upload, response speed, the chat interface and the dashboard. Security testing verified secure authentication, password hashing, session management, prevention of unauthorized access, and user-specific isolation of documents and chat history.

B. Test cases

TABLE.III. SUMMARY OF TEST CASES

Test ID	Test scenario	Expected result	Status
TC-01	User registration	User account created successfully	Pass
TC-02	User login	User logged in successfully	Pass
TC-03	Invalid login	Error message displayed	Pass
TC-04	Upload PDF	Document uploaded successfully	Pass
TC-05	Upload DOCX	Document uploaded successfully	Pass
TC-06	Upload image	OCR extracts text correctly	Pass
TC-07	Create subject	Subject created successfully	Pass
TC-08	Ask question	AI generates context-aware response	Pass
TC-09	View chat history	Previous conversations displayed	Pass
TC-10	Logout	User session terminated successfully	Pass
RT-01	Semantic search	Relevant chunks retrieved	Pass
RT-02	BM25 retrieval	Keyword matches returned	Pass
RT-03	RRF ranking	Combined results ranked correctly	Pass
RT-04	BGE reranking	Most relevant chunks prioritized	Pass
RT-05	Prompt construction	Retrieved context added correctly	Pass
DB-01	User registration (database)	Record inserted	Pass
DB-02	Login validation	User verified	Pass
DB-03	Subject creation	Subject stored	Pass
DB-04	Document upload	Metadata saved	Pass
DB-05	Chat history	Conversation stored	Pass

C. System validation and error handling

The system was validated with realistic learning scenarios: uploading subject notes, PDF textbooks and scanned notes; asking conceptual, descriptive and multiple-choice questions; retrieving previous conversations; and accessing multiple subject collections. The generated responses were relevant to the uploaded study materials. The application also handles runtime errors gracefully: invalid login credentials, unsupported file formats, empty uploads, missing queries, AI-service unavailability, database connection failures and network interruptions all produce clear validation messages instead of crashes.

D. User acceptance and performance evaluation

In user acceptance testing the evaluation focused on ease of navigation, accuracy of the generated answers, the upload process, the learning experience and overall usability. Users reported that the application answered questions based on their uploaded documents and offered a smooth, intuitive experience that did not require technical training. Table IV summarizes the evaluation of the implemented parameters.

TABLE.IV. PERFORMANCE EVALUATION

Parameter	Observation
User registration and login	Successful
Document upload and OCR processing	Successful
Text chunking and embedding generation	Successful
Semantic and hybrid retrieval	Successful
AI response generation	Successful
Chat history	Successfully stored
Overall system stability	Stable

E. Achievement of objectives

Every planned objective was implemented and verified: user authentication, document upload, PDF/DOCX/image processing, OCR text extraction, embedding generation, the SQLite vector database, hybrid retrieval, Groq LLM

integration, AI question answering, chat history, user profile management and multilingual support. The results indicate that combining semantic and keyword retrieval, fusing the rankings with RRF and reranking with a cross-encoder supplies the language model with more relevant context than a single retrieval method, and that grounding the prompt in this context keeps answers tied to the student's own materials.

IX. DISCUSSION: ADVANTAGES AND LIMITATIONS

Advantages. VidyaAI provides personalized, syllabus-aligned answers generated from the student's own documents. It supports heterogeneous inputs including scanned notes, offers multilingual interaction and several answer formats, protects user data and history with per-user isolation, and has a lightweight deployment based on SQLite with no external vector-database service. The hybrid design also compensates for the weaknesses of its parts: semantic search can miss exact terms that BM25 catches, while BM25 can miss paraphrases that embeddings capture.

Limitations. Response quality depends on the quality and relevance of the uploaded documents. OCR accuracy decreases for low-quality scans. An internet connection is required for the Groq API. Indexing and retrieval time may increase with very large document collections. The evaluation reported in this paper is functional and qualitative; a quantitative benchmark of retrieval quality (for example precision, recall or answer-faithfulness scores on a standard dataset) has not yet been carried out and is planned as future work.

X. CONCLUSION AND FUTURE SCOPE

This paper presented VidyaAI, a personalized learning assistant that answers students' questions from their own uploaded study materials using hybrid Retrieval-Augmented Generation. By combining Sentence-Transformer embeddings, BM25 keyword retrieval, Reciprocal Rank Fusion and BGE cross-encoder reranking with a Groq-hosted LLM, and by adding OCR, multilingual support, secure accounts and persistent chat history, the system delivers reliable, context-aware and syllabus-oriented assistance through a lightweight open-source stack. The implementation and testing show that the planned objectives were achieved and that the modular architecture supports scalability, maintainability and future development.

Future work includes: integration of additional LLMs with user-selectable models; cloud-based vector databases for large-scale collections; voice-based interaction using speech recognition and text-to-speech; native Android and iOS applications; collaborative sharing of subject collections; integration with LMS platforms such as Moodle and Google Classroom; automatic generation of quizzes, assignments, summaries, flashcards and study plans; analytics dashboards for learning progress; support for audio, video and presentation files; containerized cloud deployment for concurrent users; advanced security such as multi-factor authentication and role-based access control; and a quantitative evaluation of retrieval quality with advanced embedding models.

XI. ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, S J M Institute of Technology, Chitradurga, Karnataka, India, for the facilities and support provided during this work.

REFERENCES

- [1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.
- [2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT Networks," in *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, pp. 3982–3992, 2019.
- [3] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
- [4] G. V. Cormack, C. L. A. Clarke and S. Büttcher, "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods," in *Proc. 32nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Boston, MA, USA, pp. 758–759, 2009.

- [5] K. VanLehn, "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011.
- [6] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, pp. 4171–4186, 2019.
- [7] T. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details