JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PRISM: AI-POWERED LEARNING AND PREPARATION PLATFORM

[1]Dr. G Narender, [2]Srikar Veluvali, [3]Sesha Sai Pratiek Yeggina, [4] Anpur Phani Charan, [5]Srikar Narsingoju
[1]Head of Department, [2]Student, [3]Student, [4]Student, [5]Student
[1]Information Technology,
[1]Keshav Memorial Institute of Technology, Hyderabad, India

Abstract **:** A personalized learning platform that employs Retrieval-Augmented Generation (RAG) to enable students to interact with their study materials through an AI-powered assistant. The system comprises a web-based interface built with React that allows users to upload educational documents in various formats including PDF, DOCX, TXT, and YouTube video transcripts, which are then processed through a text extraction and chunking pipeline. Document chunks are encoded into 768-dimensional vector embeddings using the all-mpnet-base-v2 sentence transformer and stored in a Pinecone vector database for efficient semantic retrieval. When users pose questions, the system retrieves relevant passages from their uploaded documents and conditions a language model's response generation on this retrieved context, ensuring factually grounded and document-specific answers. The platform includes eight integrated learning modes: interactive chat-based question answering, automated quiz generation, AI-scored mock tests, virtual interview sessions, a swipeable flashcard feed (Doomscroll), per-page PDF analysis with graded questions, in-browser PDF annotation with AI-powered Q&A on highlighted text, and rich note-taking. The backend implements a FastAPI server with asynchronous HTTP endpoints, JWT-based authentication, and hybrid data persistence using MongoDB and Pinecone. The system is designed to improve learning outcomes through retrieval augmentation, reduce hallucination compared to ungrounded language models, and support concurrent users with low-latency responses, thereby democratizing personalized tutoring for students regardless of geographical or economic constraints.

Index Terms – *Retrieval-Augmented Generation, Educational Technology, Large Language Models, Personalized Learning, Vector Database, Sentence Transformers, FastAPI, React, MongoDB, Pinecone.*

I. INTRODUCTION

This paper presents PRISM, a personalized learning platform that utilizes Retrieval-Augmented Generation (RAG) technology to enhance self-study capabilities. The system enables students to upload educational documents and interact with an AI-powered assistant that provides contextually relevant answers, generates assessments, and facilitates active learning through quiz-based reinforcement. The platform addresses the challenge of providing personalized tutoring at scale, making high-quality educational support accessible to learners regardless of geographical or economic constraints.

A. Motivation

The democratization of quality education remains one of the most pressing challenges in modern society. While digital learning resources have become increasingly available, the lack of personalized guidance and interactive engagement continues to create barriers to effective self-study. Traditional educational approaches rely heavily on one-to-one tutoring or classroom instruction, which cannot scale efficiently to meet global demand. Students often struggle with passive learning methods such as reading static documents, which leads to poor retention and understanding.

The advent of large language models has created new possibilities for educational technology, enabling natural language interaction and content generation at unprecedented scales. However, general-purpose AI systems often lack the contextual grounding necessary to provide accurate, document-specific guidance. Students need systems that can understand their specific study materials, answer questions based on those materials, and generate relevant assessments to reinforce learning.

B. Limitations of Current Solutions

Existing educational platforms typically fall into two categories: content delivery systems that provide static materials without interactivity, and generic AI chatbots that lack document-specific context. Content delivery platforms such as traditional Learning Management Systems offer organized course materials but require students to passively consume information without active engagement mechanisms. While they may include quiz features, these assessments are manually created and cannot adapt to individual student materials.

Generic AI chatbots suffer from hallucination problems where they generate plausible-sounding but incorrect information when they lack relevant context. Without grounding in the specific documents a student is studying, these systems cannot guarantee accuracy or relevance of their responses. Furthermore, most existing solutions do not integrate multiple learning modalities effectively. Research in cognitive science demonstrates that active recall through testing significantly improves retention compared to passive review.

C. Prior Art

Retrieval-Augmented Generation represents a significant advancement in addressing the hallucination problem in AI systems. The RAG framework, introduced by Lewis et al. (2020), combines parametric knowledge from language models with non-parametric knowledge retrieved from external documents, demonstrating improved factual accuracy in knowledge-intensive tasks. Intelligent tutoring systems such as AutoTutor and Carnegie Learning's cognitive tutors have shown effectiveness in improving student outcomes through adaptive interaction, but typically operate within predefined knowledge domains and lack flexibility to work with arbitrary user-provided documents.

Question generation from text has emerged as an active research area, with transformer-based models showing promise in automatically creating assessment items, though ensuring pedagogical quality and appropriate difficulty level remains challenging. Vector database technologies such as Pinecone, Weaviate, and FAISS have made semantic search at scale practical, forming the foundation for effective document retrieval in RAG systems.

II. System Architecture

The platform implements a modern three-tier architecture separating presentation, business logic, and data persistence concerns. The frontend layer is built using React 18 with functional components and hooks for state management. The application employs state-based mode switching within a single-page architecture, enabling seamless navigation between document upload, chat interface, quiz modules, and other learning features without full page reloads.

The backend server is implemented in Python using the FastAPI framework, chosen for its high performance, automatic API documentation, and built-in support for asynchronous operations. The server exposes RESTful endpoints for document upload, chat interactions, quiz generation, and user authentication. Asynchronous HTTP endpoints enable efficient processing of AI responses during chat sessions, providing responsive feedback as the system processes queries. **"Fig.1"** provides a clear understanding of the system architecture.

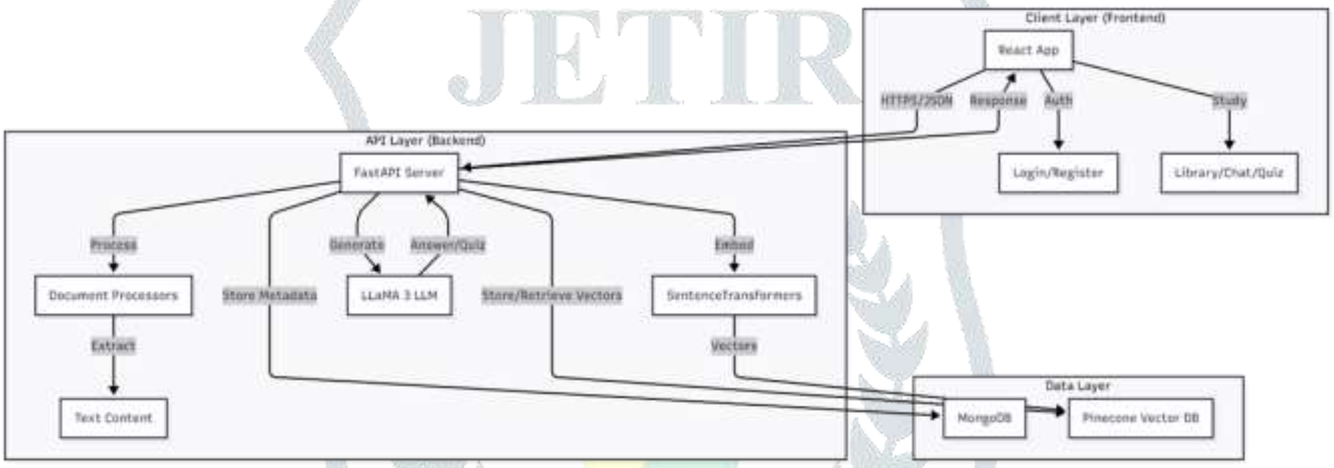


Figure 1 Architecture Diagram

Data persistence employs a hybrid approach: MongoDB stores structured data including user profiles, uploaded document metadata, conversation histories, notes, and quiz results, while Pinecone vector database handles document embeddings for semantic retrieval. This separation allows optimal performance for each type of data access pattern—traditional queries for structured data and similarity search for document retrieval. The frontend application deploys on Vercel, while Pinecone and MongoDB Atlas provide managed, auto-scaling cloud infrastructure.

III. Implementation

A. Document Processing Pipeline

When a user uploads a document through the web interface, the system initiates a multi-stage processing pipeline. File validation occurs first, checking file size limits, format compatibility, and scanning for malicious content. Supported formats include PDF, DOCX, TXT, and other common document types. The system employs format-specific parsers: PyPDF2 for PDF extraction, python-docx for Word documents, and standard file readers for plain text.

Text extraction preserves document structure where possible, maintaining section boundaries and logical grouping. The extracted text undergoes cleaning to remove artifacts from formatting, normalize whitespace, and handle special characters. Long documents are chunked into segments of approximately 1,000 characters with 200-character overlap to ensure context preservation across chunk boundaries, balancing retrieval precision with contextual completeness. **"Fig.2"** provides a clear understanding of the Document Processing Pipeline.

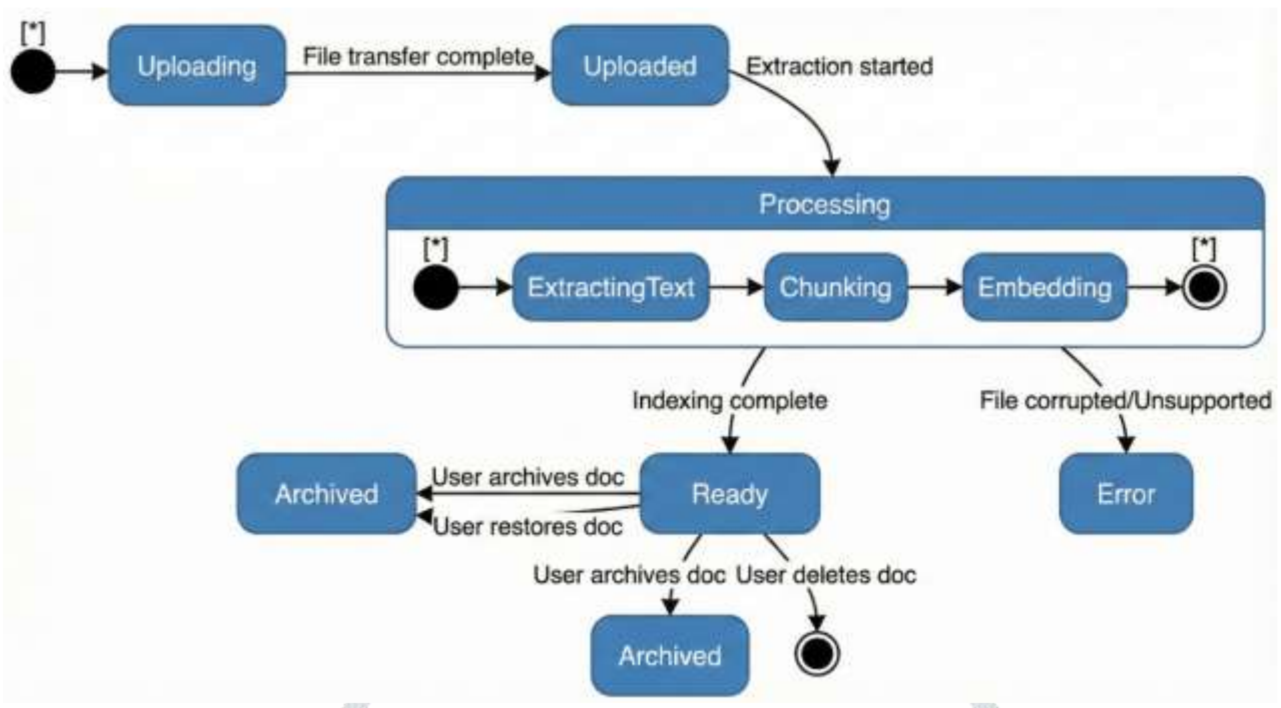


Figure 2 State Chart Diagram

Each chunk is processed through the all-mpnet-base-v2 sentence transformer model to generate a 768-dimensional dense vector embedding. These embeddings capture semantic meaning, enabling retrieval based on conceptual similarity rather than keyword matching. The embeddings are uploaded to Pinecone with metadata including the original text, chunk position, source document ID, and user ID. An index structure optimized for similarity search enables sub-second retrieval even from large document collections.

B. Retrieval-Augmented Generation

The RAG implementation follows a multi-stage process optimized for educational context. When a user submits a question, the system first embeds the query using the same sentence transformer model used for document indexing, ensuring vector space compatibility. The query embedding is used to perform similarity search in Pinecone, retrieving the top-k most relevant document chunks (typically k=5–10 depending on query complexity).

Retrieved chunks undergo relevance filtering to remove passages below a similarity threshold, ensuring only truly relevant context is included. The system constructs a prompt for the language model that includes the user's question, retrieved passages as context, and instructions to ground the response in the provided context. The prompt explicitly instructs the model to cite specific passages when making claims and to acknowledge when retrieved context is insufficient to answer the question definitively. **"Fig.3"** provides a clear understanding of the RAG Sequence.

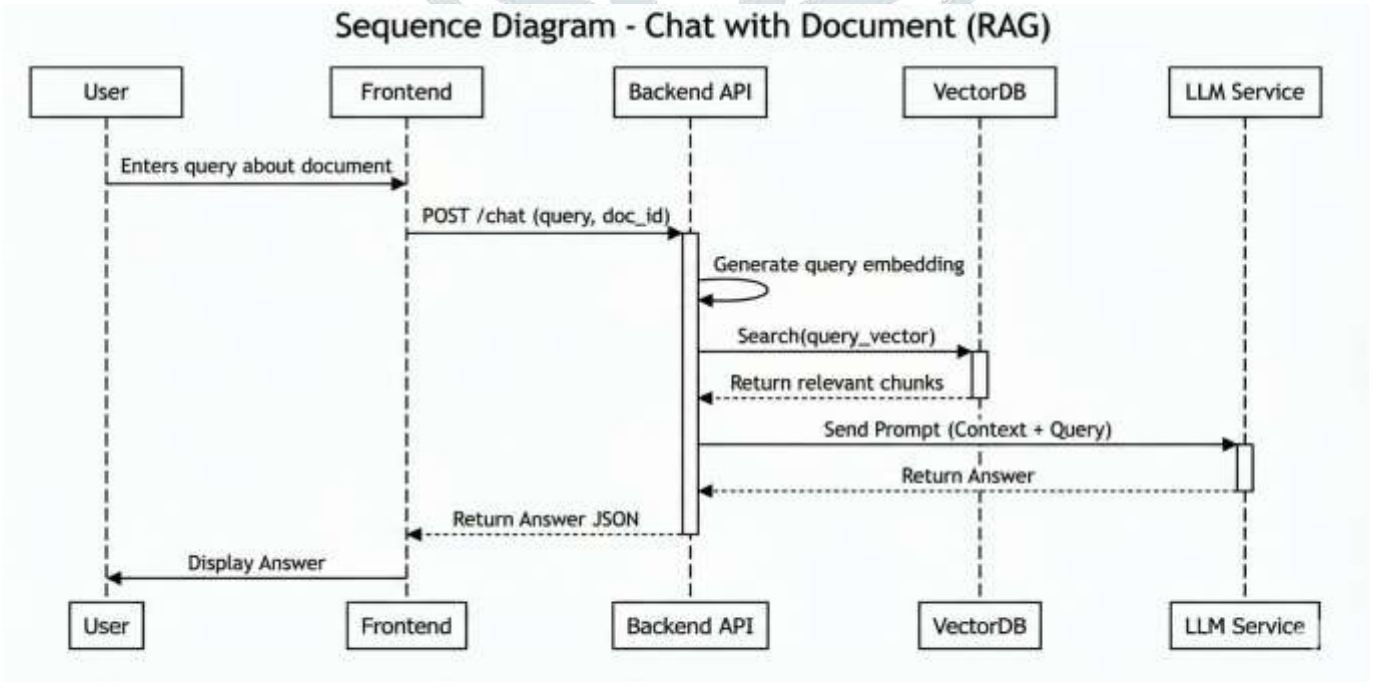


Figure 3 RAG Sequence Diagram

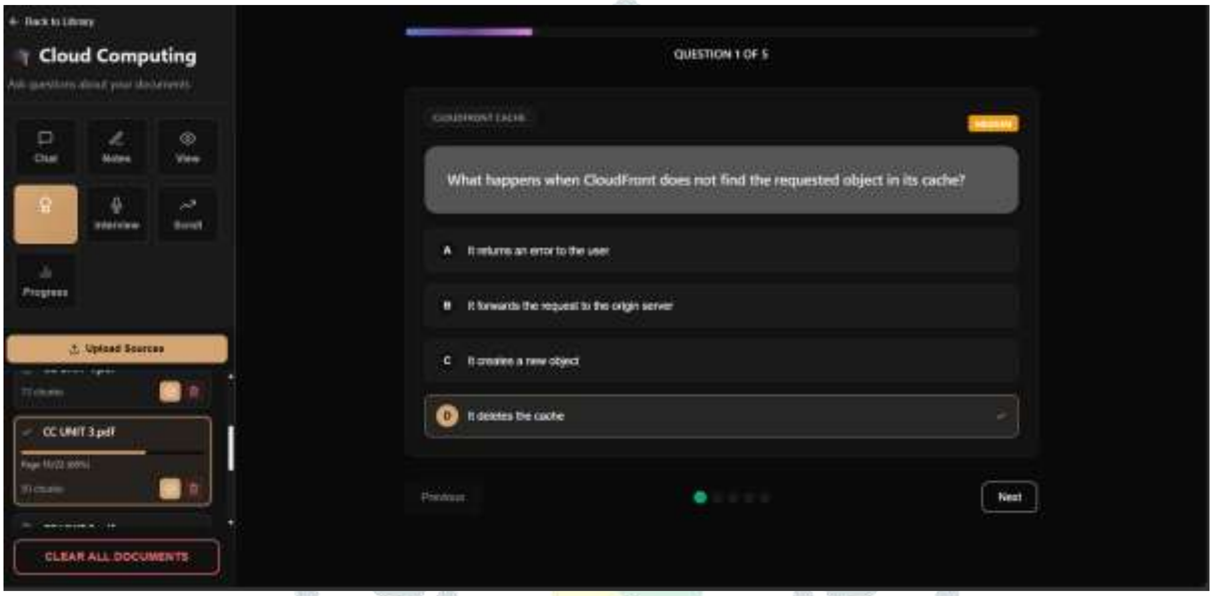
Response generation employs asynchronous processing to provide efficient feedback. As the language model generates responses, they are transmitted to the frontend via asynchronous HTTP endpoints, creating a responsive conversation flow. Post-processing validates that claims are supported by the provided context, monitoring for signs of hallucination by checking for statements that contradict or go beyond the retrieved context.

C. Automated Quiz Generation

Quiz generation employs a sophisticated pipeline that ensures pedagogical quality while maintaining automation. The system first analyzes the uploaded document to identify key concepts, facts, definitions, and relationships using named entity recognition and dependency parsing. For each identified concept, relevant passages are retrieved from the document to provide sufficient context for question generation.

Question generation uses few-shot prompting with the language model, providing examples of well-formed multiple-choice questions to guide generation. The prompt specifies question format (stem, options, correct answer, explanation), difficulty level, and cognitive level (recall, understanding, application, analysis). Quality filtering checks that questions are unambiguous, have exactly one clearly correct answer, include plausible distractors, and test important concepts rather than trivial details.

The resulting quiz is stored in MongoDB with associations to the source document and user. During quiz-taking, the frontend presents questions sequentially, records user responses, provides immediate feedback on correctness, and displays explanations for both correct and incorrect options. Quiz results are analyzed to identify patterns in student understanding and areas requiring additional study. **"Fig.4"** shows the Automated Quiz Generation UI.



What happens when CloudFront does not find the requested object in its cache?

A It returns an error to the user

B It forwards the request to the origin server

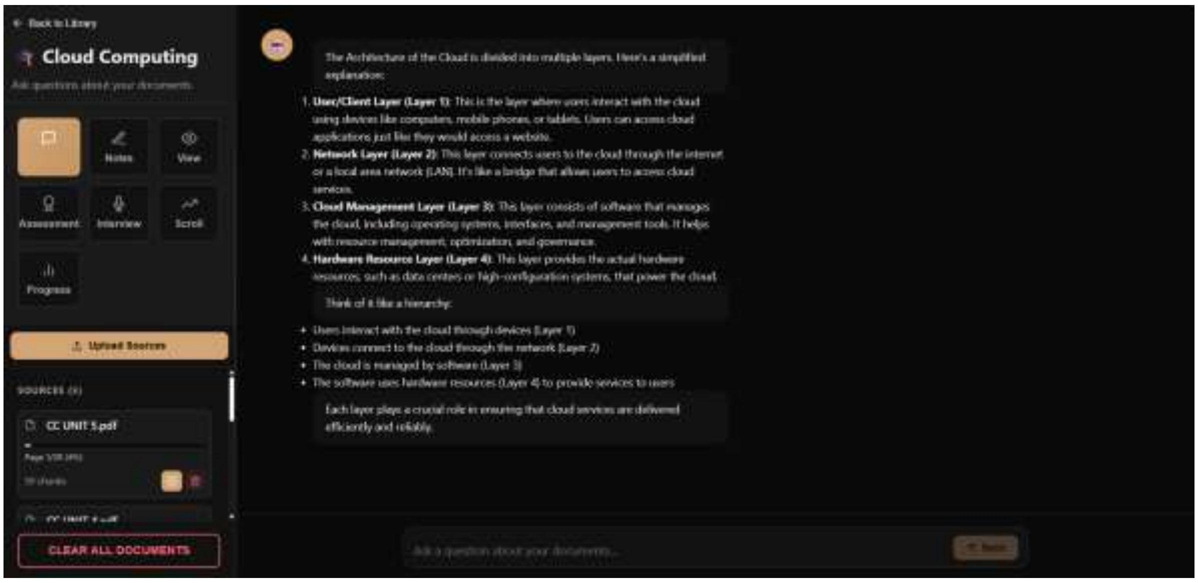
C It creates a new object

D It deletes the cache

Figure 4 Automated Quiz Generation UI

D. Learning Modes and User Experience

The user interface prioritizes simplicity and intuitive interaction. The home screen provides access to eight learning modes: chat-based Q&A, quizzes, mock tests, virtual interviews, Doomscroll flashcards, PDF Analyzer, PDF annotations, and notes. Document upload uses drag-and-drop with visual feedback during processing. The chat interface resembles familiar messaging applications, reducing cognitive load for users, and supports markdown formatting including code blocks, mathematical notation, and lists. **"Fig.5"** shows the user-friendly chat interface.



The Architecture of the Cloud is divided into multiple layers. Here's a simplified explanation:

1. **User/Client Layer (Layer 1):** This is the layer where users interact with the cloud using devices like computers, mobile phones, or tablets. Users can access cloud applications just like they would access a website.
2. **Network Layer (Layer 2):** This layer connects users to the cloud through the internet or a local area network (LAN). It's like a bridge that allows users to access cloud services.
3. **Cloud Management Layer (Layer 3):** This layer consists of software that manages the cloud, including operating systems, interfaces, and management tools. It helps with resource management, optimization, and governance.
4. **Hardware Resource Layer (Layer 4):** This layer provides the actual hardware resources, such as data centers or high-configuration systems, that power the cloud.

Think of it like a hierarchy:

- Users interact with the cloud through devices (Layer 1)
- Devices connect to the cloud through the network (Layer 2)
- The cloud is managed by software (Layer 3)
- The software uses hardware resources (Layer 4) to provide services to users

Each layer plays a crucial role in ensuring that cloud services are delivered efficiently and reliably.

Figure 5 Chat Interface

The mock test module provides full-length assessments encompassing theory, coding, and reorder question types, all AI-scored with detailed feedback. The virtual interview feature conducts multi-turn question-and-answer sessions simulating technical and behavioral interviews, with end-of-session scoring and performance analysis. The Doomscroll feature presents a swipeable flashcard feed with eight card types including fun facts, mnemonics, and definitions organized into custom folders for later review.

PDF annotation support enables in-browser highlighting with configurable colors, position and page number tracking, and AI-powered question answering on highlighted text selections. The PDF Analyzer generates per-page study questions at 2-mark, 5-mark, and 10-mark difficulty levels, with 30-day caching to avoid redundant regeneration. The platform also accepts YouTube URLs as document sources by extracting video transcripts for indexing and retrieval. Per-page bookmarks with user notes and reading progress tracking with time-spent analytics provide additional study management capabilities.

E. Security and Technical Details

Authentication employs JWT tokens with secure HTTP-only cookies to prevent XSS attacks. Password storage uses bcrypt hashing with per-user salts. Each user's uploaded documents and generated embeddings are strictly isolated through user-specific indexing in the vector database and access control in document storage, ensuring that users never receive information from other users' materials. User data encryption at rest in MongoDB ensures privacy compliance.

The system implements comprehensive error handling and logging to ensure reliability. All API endpoints include request validation using Pydantic models, rate limiting to prevent abuse, and detailed logging. Performance optimization includes caching of frequently accessed data, lazy loading of large document lists in the frontend, pagination of quiz results and conversation histories, and connection pooling for database connections.

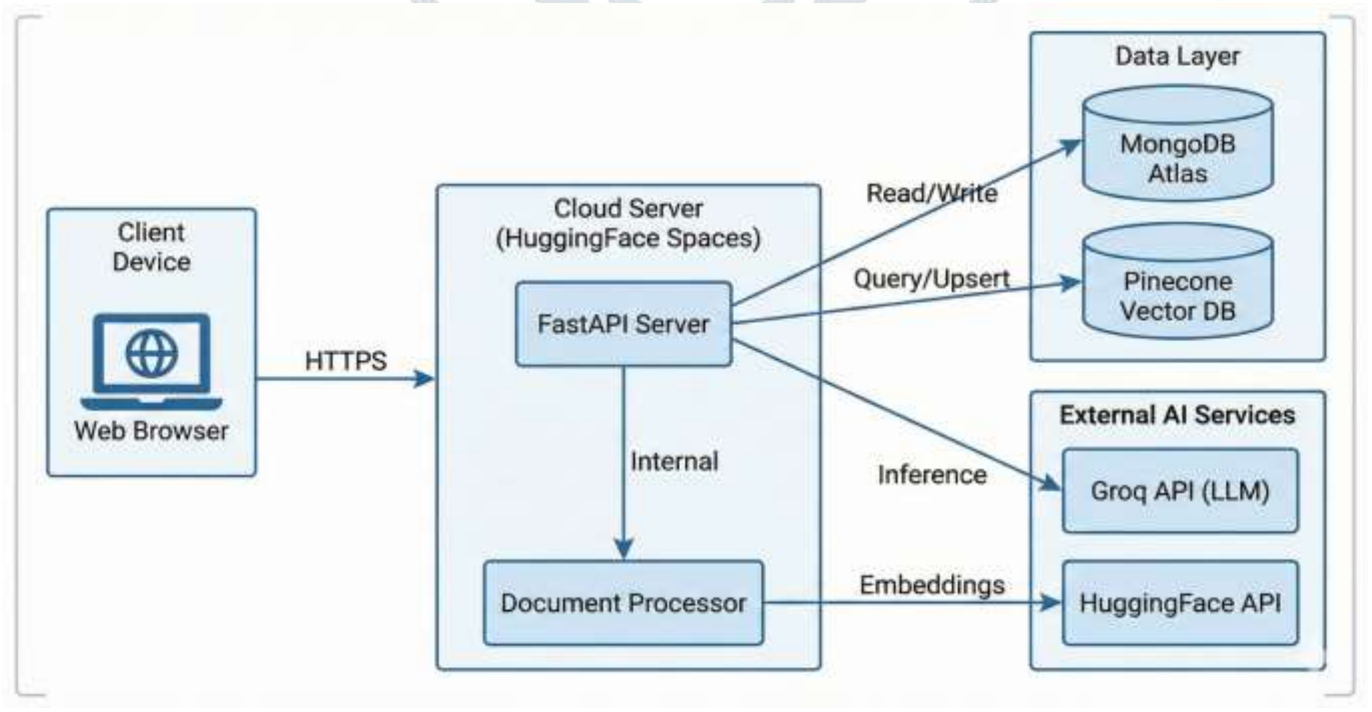


Figure 6 Deployment Diagram

IV. Results and Discussion

The platform is designed to improve learning outcomes and user engagement compared to traditional study methods. By combining retrieval-augmented generation with active recall through quizzes, mock tests, and virtual interviews, the system targets measurable improvements in post-test scores and long-term retention. Formal controlled studies are planned to quantify these improvements as the user base grows.

The RAG-based question answering is expected to achieve substantially higher accuracy when responses are grounded in retrieved source documents compared to generic language model responses without retrieval augmentation. By conditioning generation on user-specific document content, the system is designed to reduce hallucination and improve factual reliability.

Automatically generated quizzes are designed to achieve pedagogical quality ratings comparable to instructor-created assessments. The multi-stage generation pipeline with quality filtering targets high scores for question clarity, appropriate difficulty, and distractor plausibility. Formal blind-rating evaluations by experienced educators are planned to validate that automated generation approaches human-level quality.

The system underwent comprehensive testing across multiple dimensions. Functional testing verified all core features: document upload and processing, chat query and response generation, quiz creation and taking, note saving and retrieval, and user authentication. Integration testing validated the interaction between system components, ensuring proper communication between frontend and backend, correct database operations, and successful vector database queries.

Performance testing evaluated system behavior under load, measuring response times for various operations, concurrent user capacity, document processing throughput, and query response times. The system is designed to maintain responsive query times for chat interactions and efficient document processing for typical textbook chapters. User acceptance testing with students from various academic backgrounds provided qualitative feedback, with participants rating the platform highly for ease of use, accuracy of AI responses, and usefulness of quiz generation.

V. Conclusion

PRISM presents an integrated AI-powered learning platform that addresses fundamental limitations in current educational technology by combining retrieval-augmented generation with multiple active learning modalities. The system's architecture—comprising a React frontend, FastAPI backend, MongoDB for structured storage, and Pinecone for vector-based semantic retrieval—provides a scalable, reliable foundation for personalized educational support.

By grounding all AI responses in user-uploaded document content, the platform substantially reduces hallucination compared to unaugmented language models, making it suitable for serious academic use. The eight integrated learning modes—chat Q&A, automated quiz generation, AI-scored mock tests, virtual interview sessions, Doomscroll flashcards, per-page PDF analysis, in-browser PDF annotation, and rich note-taking—provide a comprehensive toolkit that covers the full spectrum of evidence-based study techniques within a single interface.

The platform's design principles of document-conditioned generation, semantic chunking with overlap preservation, multi-stage quiz quality filtering, and privacy-preserving document isolation collectively address the key challenges of accuracy, relevance, quality, and security in AI-powered educational systems. Future work includes formal controlled studies to quantify learning outcome improvements, blind educator evaluations of quiz quality, and expansion of supported content modalities.

References

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), 2020.
[2] A. C. Graesser, N. K. Person, and J. P. Magliano, "Collaborative dialogue patterns in naturalistic one-to-one tutoring," Applied Cognitive Psychology, vol. 9, no. 6, 1995.
[3] C. P. Rosé et al., "Interactivity by Design Makes a Difference in Computer Supported Collaborative Learning," International Journal of Computer-Supported Collaborative Learning, 2008.
[4] J. Johnson and A. Lella, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," EMNLP, 2019.
[5] E. Malmi et al., "Automatic Generation of Short Answer Feedback," ACL, 2018.
[6] A. Radford et al., "Language Models are Unsupervised Multitask Learners," OpenAI Blog, 2019.
[7] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, 2009.