Journal of Science, Computing and Engineering Research (JSCER)
Volume-9, Issue-3, March 2026.
https://jscer.org/doi/10.46379/jscer.2024.0709

INTELLIGENT PDF QUESTION ANSWERING SYSTEM USING RAG AND LARGE LANGUAGE MODELS

[1]J.R.Arun Kumar,[2]Nitesh Tiwari,[3]Suyash Pradhan,[4]Neeraj Sharma,[5]Divyanshu Airan,[6]Vinay Saini

[1]Professor, Department of AI&DS, Modern Institute of Technology and Research Centre, Rajasthan, India.
[2,3,4,5,6]UG Student, Department of AI&DS, Modern Instituste of Technology and Research Centre, Rajasthan, India

Article Information
Received: 26 March 2026
Revised : 26 March 2026
Accepted: 27 March 2026
Published: 30 March 2026

Corresponding Author:
Nitesh Tiwari

Abstract— The rapid growth of digital documents has increased the demand for intelligent systems capable of retrieving accurate information from unstructured data. Traditional search-based systems extract keyword-matching results, often lacking contextual understanding and semantic relevance. To address this challenge, this project proposes an Intelligent PDF Question Answering System leveraging Retrieval-Augmented Generation (RAG) integrated with Large Language Models (LLMs) to provide precise, context-aware answers from PDF documents.

The system processes uploaded PDFs through an end-to-end pipeline consisting of document parsing, text chunking, embedding generation, vector storage, and retrieval-based response generation. Extracted text is converted into vector embeddings using transformer-based encoders and stored in a vector database such as FAISS or Pinecone. When a query is provided, the retriever fetches semantically relevant chunks, which are then passed to a generative LLM (e.g., GPT, LLaMA, or Mistral) to produce coherent, human-like answers grounded in the retrieved context. This hybrid approach reduces hallucinations and improves factual accuracy compared to pure generative models. The system features a user-friendly interface enabling document uploads, conversational query interaction, and citation-based output to highlight the exact source text.

Keywords: *Artificial Intelligence, Google Gemini, Retrival-Augmented Generation (RAG), MySQL, Natural Language Processing, Vector Database, Large Language Model (LLM)*

Citation: J.R.Arun Kumar, Nitesh Tiwari, Suyash Pradhan, Neeraj Sharma, Divyanshu Airan, Vinay Saini, "INTELLIGENT PDF QUESTION ANSWERING SYSTEM USING RAG AND LARGE LANGUAGE MODELS", Journal of Science, Computing and Engineering Research, 8(05), May 2026.

I. INTRODUCTION

An Intelligent PDF Question Answering System using Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) is an advanced artificial intelligence-based solution designed to extract information from PDF documents and provide accurate, meaningful answers to user queries in natural language. As the volume of digital documents grows across academic, corporate, and government sectors, retrieving relevant information from lengthy PDFs remains a major challenge.

Traditional keyword search only locates matching words without understanding context or providing summarized insights, often forcing users to read large amounts of text manually. To overcome these limitations, this system combines the semantic reasoning abilities of large language models with the precision of retrieval mechanisms that fetch relevant portions of the document before generating a response.

II. PROBLEM STATEMENT

The exponential growth of digital documents, particularly PDFs, has made accessing information easier but extracting meaningful insights remains challenging. Traditional search methods rely on keyword matching and fail to understand context, relationships, or deep semantics, leading to inefficiency and missed information.

There is a lack of an integrated, user-friendly system that enables users to interact with PDF documents conversationally while ensuring accurate, context-aware, and citation-backed responses. Therefore, there is a need to develop an intelligent PDF Question Answering system using Retrieval-Augmented Generation (RAG) that combines semantic retrieval with generative AI to provide precise, reliable, and explainable information extraction from documents. Ultimately, there is an urgent need to build a robust, scalable, AI-driven PDF question answering system that converts passive digital texts into dynamic, verifiable, and intelligent sources of knowledge

III. PROPOSED METHOD

The proposed system aims to deliver accurate, personalized, and context-aware academic assistance to students by integrating document retrieval mechanisms with a generative AI model. It leverages teacher-uploaded study materials as the primary knowledge base and enables students to receive precise answers, explanations, and customized study plans based on their learning needs. The system follows a Retrieval-Augmented Generation (RAG) framework to ensure that responses are grounded in authentic academic content while maintaining conversational intelligence.

A. Knowledge Source

The knowledge base consists of teacher-uploaded academic resources such as PDFs, lecture notes, and study materials. These documents are stored in a local server directory, while their associated metadata—including title, subject, file path, and uploader information—is maintained in a Vector database. This structured storage ensures efficient document management, indexing, and retrieval.

B. Data Preprocessing

Once documents are uploaded, the system extracts textual content using a PDF parsing library. The extracted text undergoes preprocessing steps such as cleaning, normalization, and removal of noise. The refined content is then divided into smaller, meaningful chunks to enhance retrieval efficiency. This step ensures that the system provides relevant and contextually precise information during query processing.

C. Embedding Generation

Each processed text chunk is converted into a numerical vector representation using a pre-trained embedding model such as SentenceTransformers. These embeddings capture the semantic meaning and contextual relationships within the content. By transforming text into vector space, the system enables effective comparison between student queries and stored academic material.

D. Semantic Search

When a student submits a question, the query is converted into an embedding using the same embedding model. The system then performs similarity search in the vector database to retrieve the most relevant text chunks. This ensures that the answer is grounded in appropriate and context-related study material.

E. Large Language Model (LLM) Integration

The retrieved content is combined with the student query and passed to the Google Gemini model through a Retrieval-Augmented Generation approach. This allows the model to produce accurate and context-aware responses while reducing hallucination, since the generated answer is based on verified information from uploaded documents.

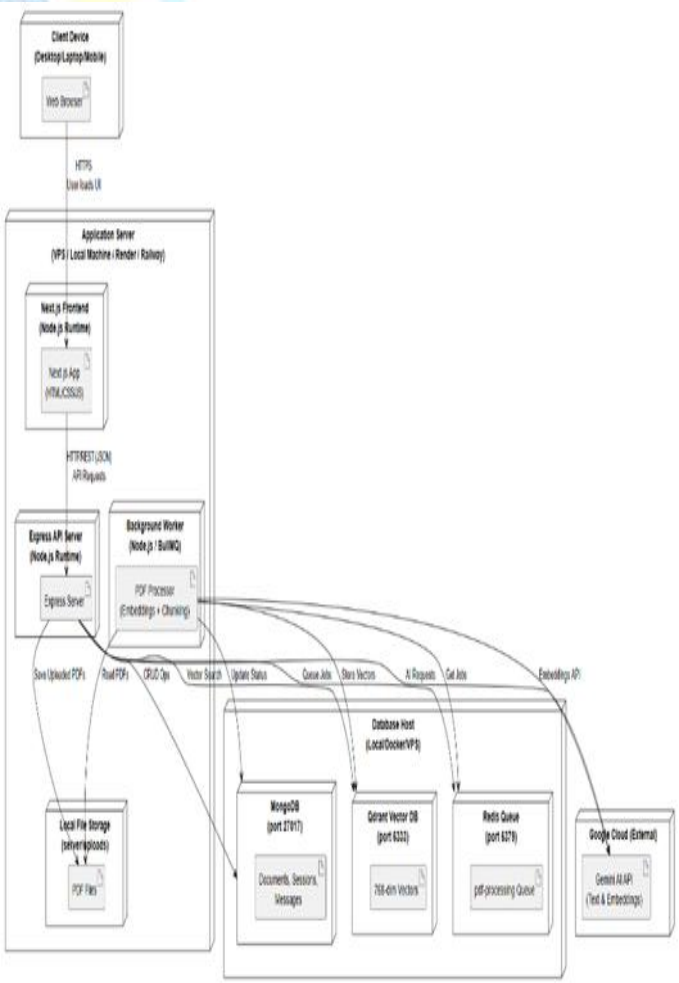
F. Response Generation

The system tracks and analyzes student interactions, queries, and performance patterns to identify strengths and areas for improvement. Based on this analysis, it generates personalized feedback and structured study plans. These plans may include daily learning goals, revision strategies, and topic recommendations, enabling students to follow a more effective and adaptive learning path.

G. System Architecture and Deployment

The Deployment diagram maps system components to the physical and runtime infrastructure of the Intelligent PDF Question Answering System. It may include frontend clients running on web browsers, backend services deployed on cloud or containerized environments (e.g., Docker or Kubernetes), a vector database hosted on a managed cloud instance, an LLM served through an API or on a dedicated inference server, and storage layers for PDFs and extracted text. This diagram supports planning for scalability, resource allocation, performance optimization, and fault-tolerant deployment, especially when transitioning from development to production environments.

Fig. 1. Deployment Diagram



IV. TECH STACK

A. Frontend Technologies

The frontend of the proposed system is developed using **React.js**, which enables the creation of a responsive, interactive, and user-friendly interface for both students and teachers. To enhance the visual appearance and improve design flexibility, **Tailwind CSS** is used for styling. The frontend is responsible for handling chat-based interaction, displaying learning dashboards, showing study plans, and managing document uploads. For seamless communication with backend services, **Axios** is used to make HTTP requests and fetch data efficiently.

B. Backend Technologies

The backend of the system is built using **FastAPI**, a modern Python framework that supports high-performance API development. It manages core functionalities such as user authentication, document upload handling, question processing, interaction storage, and study plan generation. The backend also integrates the retrieval-augmented generation pipeline and connects all major modules of the system.

C. Artificial Intelligence and Natural Language Processing

The intelligent capabilities of the system are powered by **Google Gemini**, which acts as the large language model for answering student queries, generating feedback, and creating personalized study plans. To improve the accuracy of answers, the system uses **Retrieval-Augmented Generation (RAG)**, where relevant academic content is first retrieved and then passed to the language model. For semantic understanding and embedding generation, **SentenceTransformers** is used. These embeddings help in finding the most relevant document chunks from the knowledge base.

D. Database Technologies

The system uses **MySQL** as the primary relational database for storing structured data such as user information, document metadata, interaction history, summaries, and study plans. PDF files themselves are not stored directly in the database; instead, they are saved in a local server directory, and only their file paths are stored in MySQL. This approach keeps the database lightweight and improves performance

E. Vector Database and Retrieval

To support semantic search and context retrieval, the project uses a **vector database** such as **FAISS** or **Milvus**. After PDF text is converted into embeddings, these embeddings are stored in the vector database. When a student asks a question, the system performs similarity search to retrieve the most relevant content chunks. This makes the generated response more accurate, context-aware, and aligned with the uploaded study material.

F. PDF Processing and Text Extraction

Since the project relies on teacher-uploaded academic documents, **PyMuPDF** is used to extract text from PDF files efficiently. After extraction, the text is divided into smaller chunks for better retrieval and processing. This step forms the foundation of the knowledge base used by the AI assistant. Chunking and preprocessing ensure that the model receives relevant and manageable context during response generation.

G. Authentication and Security

For secure access control, the system implements **JWT-based authentication**. This ensures that only authorized users can access specific functionalities based on their role, such as student or teacher. Sensitive configuration values like database credentials, Gemini API keys, and secret keys are stored securely in environment variables using a **.env** file. This enhances application security and prevents direct exposure of confidential information in the source code.

H. Development Tools and Deployment Support

The project is developed using **Visual Studio Code** as the primary code editor. **Git** and **GitHub** are used for version control and project collaboration. For dependency management and backend testing, Python's package ecosystem is utilized. The system is designed in such a way that it can later be deployed on cloud platforms or containerized using **Docker** for better scalability and maintainability.

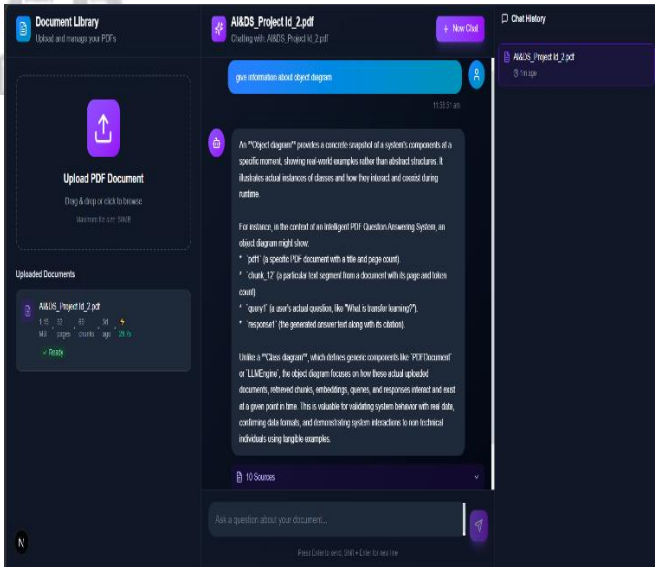


Fig. 2. Web Interface

V. RESULTS

The proposed **Intelligent PDF Question Answering System** was successfully implemented and tested on multiple academic PDF documents. The system demonstrated efficient document processing, accurate semantic retrieval, and context-aware response generation using the Retrieval-Augmented Generation (RAG) approach. After uploading PDF documents through the frontend interface, the system was able to extract and preprocess text effectively, including handling structured and semi-structured content. The chunking and embedding process enabled the system to create a semantic index of the document, allowing meaningful comparison between user queries and document content. When users submitted natural language queries, the system successfully retrieved the most relevant document chunks using similarity search and generated accurate responses using the Large Language Model. The answers were context-aware and aligned with the uploaded documents, significantly reducing irrelevant or hallucinated outputs. The system also provided supporting evidence in the form of retrieved text snippets and page references, improving transparency and trust.

VI. CONCLUSION

The The Intelligent PDF Question Answering System successfully addresses the challenges associated with extracting meaningful information from large and complex PDF documents. By integrating Retrieval-Augmented Generation (RAG) with advanced Large Language Models, the system transforms static academic materials into an interactive and intelligent knowledge platform.The system effectively processes teacher-uploaded documents, performs semantic indexing through embeddings, and retrieves contextually relevant information to generate accurate and reliable responses. Unlike traditional keyword-based search systems, it understands the meaning and intent behind user queries, providing precise, context-aware, and evidence-backed answers. The inclusion of citation-based responses further enhances transparency and trust in the system.

REFERENCES

[1]. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems.

[2]. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.

[3]. Johnson, J., Douze, M., & Jégou, H. (2017). Billion-Scale Similarity Search with GPUs. IEEE Transactions on Big Data.

[4]. Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020). REALM: Retrieval-Augmented Language Model Pre Training.

[5]. Izacard, G., & Grave, E. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. Proceedings of the European Chapter of the Association for Computational Linguistics.

[6]. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems.

[7]. React Team. (2023). React: A JavaScript Library for Building User Interfaces. Meta Open Source.