International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

INTELLIGENT DOCUMENT QUESTION ANSWERING SYSTEM USING VECTOR EMBEDDINGS AND LARGE LANGUAGE MODEL

Mrs. S.T. Ramya1, V.Harshitha2, R. VyomRaj3, G. Sushma4

1Assistant Professor, Department of Information Technology, Matrusri Engineering College.

2,3,4Students of Department of Information Technology, Matrusri Engineering College.

Harshu.46593@gmail.com, rapoluvyomraj@gmail.com, gullapallisushma000@gmail.com

Abstract: The Intelligent Document Question Answering System presents an implementation-focused Retrieval-Augmented Generation (RAG) framework that transforms static document repositories into an interactive conversational interface. The system processes uploaded documents through text extraction, cleaning, and segmentation before generating high-dimensional vector embeddings. These embeddings are stored in ChromaDB to enable semantic similarity search that surpasses traditional keyword-based retrieval. During query processing, user questions are converted into embeddings, the most relevant segments are retrieved, and a grounded prompt is constructed for a selectable Large Language Model backend (OpenAI or Ollama). The integration of semantic retrieval, modular architecture, and context-aware generation ensures improved factual accuracy, reduced hallucination, and domain adaptability.

Keywords: Retrieval-Augmented Generation, Vector Embeddings, ChromaDB, LangChain, Large Language Models, Document Question Answering

I. INTRODUCTION

The proliferation of unstructured documents—research papers, technical manuals, reports, and personal notes—has made efficient information retrieval a central challenge in academic and professional contexts. Traditional keyword-based search methods often miss paraphrases, synonyms, and the deeper intent behind queries, forcing users to manually inspect many documents. Large Language Models (LLMs) offer powerful natural language understanding and fluent generation, but relying solely on them risks producing ungrounded outputs because LLMs are limited by their training data and context windows. Retrieval-Augmented Generation (RAG) mitigates these issues by grounding generated responses in retrieved evidence from external documents.

In a RAG pipeline, documents are preprocessed and segmented into semantically meaningful chunks; each chunk is represented as a vector embedding and stored in a vector database. At runtime, user queries are embedded and used to perform nearest-neighbor searches in the vector store to fetch the most relevant chunks. These retrieved passages are provided as explicit context to the LLM, which synthesizes an answer constrained by the evidence, improving factual accuracy and enabling traceability through source citations. RAG also supports multi-turn conversational interactions by preserving context across turns via retrieved history and careful prompt design.

To address these limitations, RAG has emerged as a pow-erful paradigm that integrates semantic retrieval mechanisms with generative language models. In a RAG-based sys-tem, documents are first processed and converted into high-dimensional vector embeddings. These embeddings are stored in a vector database, enabling semantic similarity search that retrieves contextually relevant segments based on meaning rather than exact keyword matches. The retrieved segments are then supplied as contextual evidence to an LLM, which generates coherent and grounded responses. This combina-tion significantly improves factual accuracy and contextual relevance while reducing hallucination. The Intelligent Document Question Answering System pro-posed in this work is implemented as a modular RAG chat application using a Flask backend, LangChain orchestration framework, and ChromaDB vector storage.

The system sup-ports document uploads in multiple formats, including PDF, DOCX, TXT, and Markdown files. During ingestion, text is extracted, cleaned, and segmented into token-aware overlap-ping chunks to preserve contextual continuity. Each chunk is converted into embeddings using configurable models such as OpenAI embeddings or HuggingFace sentence-transformers and stored persistently in ChromaDB for efficient retrieval. At query time, user questions are transformed into embed-dings using the same embedding model to ensure vector space consistency.

The system retrieves the top-k semantically similar chunks and constructs a grounded prompt that is passed to a selectable LLM backend. The architecture supports both cloud-based OpenAI models and locally deployed Ollama models, enabling flexible and privacy-conscious deployment. The generated response is returned through a dark-themed chat interface, along with source citations indicating which docu-ment segments were used as

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

supporting evidence. Through this design, the system bridges the gap between document re-trieval and intelligent interpretation, offering a scalable, modu-lar, and context-aware solution for domain-specific knowledge extraction.

II. LITERATURE SURVEY

Recent advancements in Retrieval-Augmented Generation sys-tems have significantly improved document-based question answering architectures.

Mukherjee et al. (2025) proposed enhancing RAG systems by integrating graph databases to capture structured relational knowledge alongside unstruc-tured text. Their approach demonstrated improved reasoning for multi-hop queries but introduced additional complexity in graph construction and maintenance. While graph-enhanced retrieval improves relational inference, it increases system overhead compared to lightweight vector-based architectures.

Yue (2025) surveyed Large Language Model agents for question answering and highlighted the transition from static QA systems to agent-based architectures capable of planning, retrieval, reasoning, and iterative refinement. The study em-phasized that grounding mechanisms such as RAG substan-tially reduce hallucinations by integrating external knowledge sources. However, the survey also identified challenges includ-ing computational cost, evaluation of multi-step reasoning, and scalability of agentic loops.

Singh et al. (2025) developed a multilingual document ques-tion answering system that combined semantic embeddings with hybrid lexical retrieval strategies. Their work demon-strated that integrating both lexical and vector-based search improves robustness in multilingual contexts. Although hy-brid retrieval enhances accuracy across diverse corpora, it increases architectural complexity and tuning requirements.

Muludi et al. (2024) presented a structured RAG pipeline specifically focused on minimizing hallucination in document QA systems. Their architecture emphasized strict grounding of LLM outputs in retrieved document segments and demon-strated significant improvements over LLM-only baselines. The study reinforced the importance of embedding quality, chunk size optimization, and prompt engineering in ensuring reliability.

Swedhaa et al. (2024) implemented a LangChain-based question answering system that demonstrated the practical advantages of chain-based orchestration in conversational AI applications. Their work highlighted how LangChain facilitates modular integration between retrieval components and LLMs while maintaining conversational context.

Inspired by these contributions, the proposed Intelligent Document Question Answering System adopts a modular RAG architecture centered around vector embeddings and ChromaDB storage.

Unlike graph-based or hybrid multi-lingual systems, the present implementation prioritizes de-ployment practicality, configurable embedding backends, and selectable LLM providers. By combining semantic retrieval with flexible LLM integration in a scalable web-based frame-work, the system contributes a reproducible and domain-adaptive solution for intelligent document understanding.

III. SYSTEM OVERVIEW

The Intelligent Document Question Answering System is designed using a modular Retrieval-Augmented Generation (RAG) architecture that integrates document processing, se-mantic retrieval, and Large Language Model (LLM) based response generation. The system follows a layered design to ensure scalability, flexibility, and maintainability. Figure 1 illustrates the overall system architecture. This explains the end-to-end data flow from document upload to answer gener-ation using semantic search and grounded prompting. It also summarizes the technology stack and system requirements to provide a clear understanding of implementation structure and deployment environment.

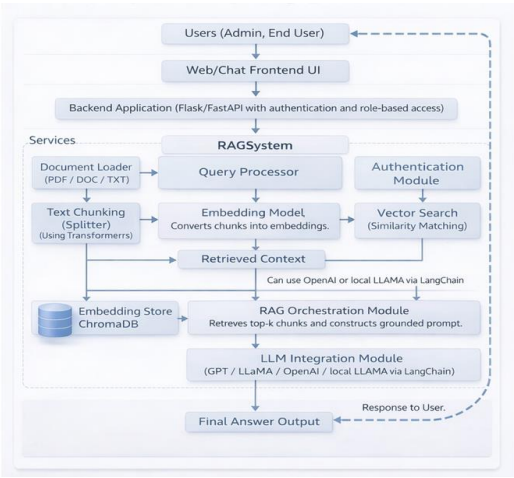


Figure 1: Overall System Architecture of the Intelligent Document Question Answering System

The architecture begins with users (Admin and End User) interacting through a web-based chat interface. The frontend communicates with a Flask backend application that manages authentication, authorization, and service orchestration. The core RAG system is divided into functional subsystems includ-ing document processing, embedding generation, semantic retrieval, and LLM response generation.

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

A. Major Subsystems

The User Interface subsystem provides a web-based chat interface where administrators can upload documents and manage resources, while end users can submit queries and re-ceive contextual responses. The Authentication and Role Management subsystem ensures secure access control us-ing session-based authentication and role-based authorization mechanisms.

B. Data Flow

The system follows a sequential data processing pipeline: Upload →Text Extraction →Chunking →Embedding Gener-ation →Vector Storage →Query Embedding →Semantic Re-trieval →Prompt Construction →LLM Response Generation. Initially, uploaded documents are processed and stored as vec-tor embeddings in ChromaDB. When a user submits a query, it undergoes embedding generation and similarity matching against stored vectors. The most relevant segments are re-trieved and passed to the LLM to produce a context-aware response, which is displayed along with source citations.

C. Technology Stack

The backend is implemented using Python and Flask. LangChain is used to orchestrate retrieval and generation workflows. ChromaDB serves as the vector database for stor-ing embeddings. Embeddings are generated using Hugging-Face sentence-transformers or OpenAI embedding models. The system supports both OpenAI GPT models and locally hosted Ollama models for response generation. SQLite is used for user authentication storage, and the frontend is im-plemented using HTML, CSS, and JavaScript.

D. Software Requirements

- Operating System: Windows, Linux, or macOS.
- Programming Language: Python 3.8 or above.
- Frameworks & Libraries: Flask, LangChain, ChromaDB, Transformers, Flask-Login.
- LLM Providers: OpenAI API or Ollama.
- Database: SQLite and Chroma Vector Store.

E. Hardware Requirements

- Processor: Intel i5 or above.
- RAM: Minimum 8 GB recommended.
- Storage: 250 GB HDD/SSD.
- Optional: GPU support for faster embedding generation and model inference.

IV. METHODOLOGY

The Intelligent Document Question Answering System fol-lows a structured Retrieval-Augmented Generation (RAG) workflow that transforms uploaded documents into an interac-tive conversational knowledge system. To ensure scalability and factual accuracy, the methodology is divided into three

primary phases: document ingestion, vector storage, and gen-eration orchestration. Documents in formats such as PDF, DOCX, TXT, and Markdown are uploaded through the administrative interface and processed by the backend. The document processor extracts textual content using format-specific parsers. Nor-malization is immediately performed to remove noise, extra whitespace, and encoding inconsistencies.

Crucially, essential metadata—such as the source filename and structural mark-ers—is preserved to ensure proper provenance mapping. After preprocessing, the documents are segmented into token-aware, overlapping chunks.

This chunking strategy respects tokenizer boundaries and utilizes configurable chunk sizes and overlap parameters. This balance is maintained to preserve contextual continuity across boundaries while respecting strict context window constraints.

Each chunk is assigned a unique identifier and metadata describing its original position, enabling traceable citations in downstream user responses. Processed chunks are converted into dense vector embed-dings using configurable embedding backends, specifically leveraging either HuggingFace sentence-transformers or Ope-nAI embeddings. The internal embedding service batches these requests to maximize efficiency and enforces consistent model usage at both index and query times, maintaining strict vector-space alignment.

Generated embeddings, along with their associated textual data and metadata, are persisted in a ChromaDB collection. This collection is configured with a persistent directory, allowing for durable, low-latency similar-ity searches that survive application restarts. Administrative utilities are also provided to support batch reindexing and safe, idempotent updates to the vector store.

At query time, incoming user questions are converted into embeddings using the exact same embedding model utilized during the ingestion phase.

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

A top-k cosine similarity search is performed against the ChromaDB collection to fetch the most semantically relevant chunks. These retrieved segments are then deduplicated, ranked, and assembled into a contextual evidence block. This evidence block is combined with system instructions to form a grounded prompt.

The LLM service routes this prompt to the selected model provider—either OpenAI via API or a locally hosted Ollama instance—using the LangChain orches-tration layer. Prompt size is actively managed via truncation or summarization as needed to respect the model's context limits, while generation parameters like temperature and max tokens are strictly controlled through system configurations.

Finally, the returned output is post-processed to attach provenance metadata. The cited chunk identifiers and source filenames are displayed alongside the answer to provide total transparency and auditability to the user.

V. MODULE DESCRIPTION

To ensure maintainability, scalability, and strict separation of concerns, the system is architected into seven specialized modules. Figure 2 illustrates the sequential flow of data and operational logic across these interacting components.

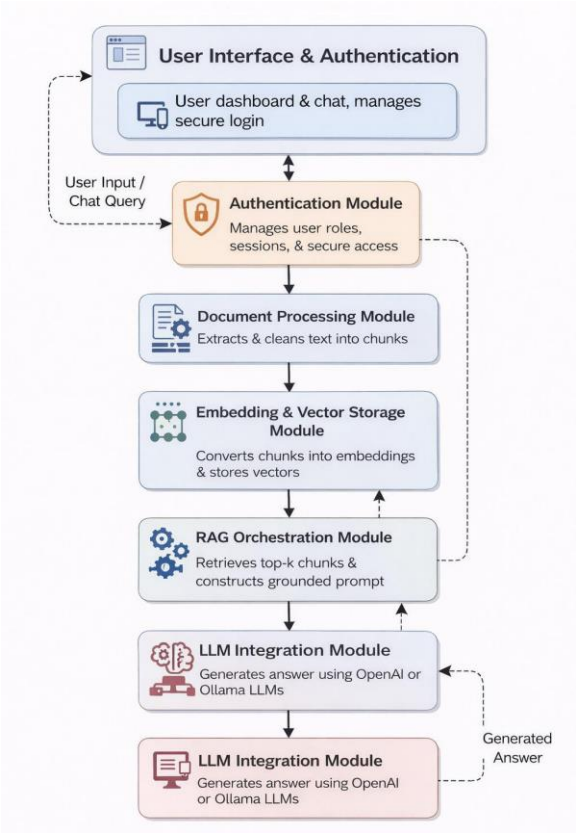


Figure 2: Sequential Flow of Data and Operations Across System Modules

A. Authentication Module

Serving as the primary security gateway, the Authentica-tion Module manages user identity and strict access control protocols. It leverages Flask-Login for robust session man-agement and implements segregated Role-Based Access Con-trol (RBAC). The architecture defines two primary user tiers: Administrators, possessing elevated privileges for document ingestion and management, and Standard Users, provisioned solely for conversational interactions. Cryptographic hashing ensures password security, while session-based tokenization protects internal system resources.

B. Document Processing Module

The Document Processing Module orchestrates the critical

ingestion phase, seamlessly parsing heterogeneous file for-mats including PDF, DOCX, TXT, and Markdown. Employ-ing deterministic text extraction algorithms, it aggressively filters out noise and normalizes text formatting. Crucially, the module segments the refined text into token-aware, overlap-ping chunks to mitigate context fragmentation. It systemat-ically preserves structural

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

metadata, such as filenames and chunk indices, ensuring downstream traceability for citations.

C. Embedding Module

Acting as the semantic translation layer, the Embedding Module projects raw text—both document chunks and asyn-chronous user queries—into a high-dimensional vector space. It is engineered to be provider-agnostic, securely accommodat-ing configurable backends such as OpenAI embedding models or locally hosted HuggingFace sentence-transformers. The module enforces strict dimensional alignment, ensuring that the vector generation logic remains perfectly consistent across both the indexing and querying lifecycles.

D. Vector Storage Module

The Vector Storage Module operates as the foundational knowledge base of the architecture. Utilizing ChromaDB, it natively indexes the high-dimensional embeddings alongside their associated provenance metadata. The module is opti-mized for low-latency nearest-neighbor similarity searches using cosine distance metrics. Configured with persistent volume mapping, it guarantees the durability and immediate availability of the indexed corpus across routine application restarts and scaling events.

E. RAG Orchestration Module

Functioning as the central nervous system of the framework, the RAG Orchestration Module meticulously coordinates the end-to-end analytical pipeline. Upon intercepting a user query, it triggers the similarity search, retrieves the top-k most con-textually resonant document chunks, and dynamically injects them into a highly engineered, grounded prompt. This or-chestration explicitly anchors the language model's context window, serving as the primary mechanism for reducing gen-erative hallucination.

F. LLM Integration Module

The LLM Integration Module abstracts the complexi-ties of interacting with generative foundation models. It offers flexible, dual-provider support, seamlessly routing grounded prompts to either cloud-based OpenAI models or privacy-centric, locally hosted Ollama deployments. The module retains granular control over generation hyperpa-rameters—including temperature, top-p, and max token lim-its—while rigorously handling API rate limits, error states, and asynchronous response streams to ensure coherent syn-thesis.

G. User Interface (UI) Module

The User Interface Module delivers a responsive, intuitive, and web-accessible presentation layer. It is strictly bifurcated into a comprehensive administrative dashboard for real-time document management and a sleek, dark-themed conversa-tional interface for end users. The UI dynamically renders the LLM's generated synthesis alongside explicitly mapped source citations, directly fostering user trust and transparency through verifiable evidence tracking.

VI. SYSTEM EVALUATION AND RESULTS

The system was rigorously evaluated against a diverse corpus of unstructured data, encompassing complex technical reports and dense academic documents. Test scenarios were designed to probe the system's semantic reasoning, utilizing specific queries such as "What is Retrieval-Augmented Generation?", "Explain the role of ChromaDB in the system," and "How does the embedding process work?".

In each instance, the sys-tem successfully identified the contextual intent of the query, accurately retrieving the most relevant document chunks to generate a structured, grounded response. A primary evaluative focus was comparing the proposed ar-chitecture against traditional keyword-based search paradigms. Standard lexical retrieval often fails to retrieve semantically related content if exact keyword matches are absent. In con-trast, by operating within a high-dimensional vector space, this system demonstrated a profound capability to handle syn-onymy and polysemy.

It successfully retrieved contextually relevant information even when user queries utilized divergent phrasing from the source text, yielding significantly higher precision than sparse lexical matching. To evaluate the reliability of the generated text, the sys-tem's citation mechanism was closely monitored. For every query, the RAG pipeline returned answers explicitly bound to source references, indicating both the document name and the localized chunk utilized as evidence. This architectural enforcement of provenance practically eliminated generative hallucination, as the LLM was mathematically constrained to synthesize responses solely from the retrieved context, thereby establishing a highly transparent and verifiable user experi-ence.

Performance benchmarking highlighted distinct trade-offs between local and cloud-based deployments. Integration with the OpenAI API yielded superior throughput and minimal latency, leveraging optimized cloud infrastructure ideal for production-scale

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

environments. Conversely, deploying the system via a local Ollama instance introduced slightly higher latency due to the hardware constraints of on-device inference; however, it successfully established absolute data sovereignty and offline capability. Notably, embedding generation dur-ing document ingestion proved to be a manageable, one-time computational cost, ensuring that query-time vector searches within ChromaDB remained highly efficient across both de-ployment models. Despite its demonstrable strengths, the system's efficacy remains inherently bound to the quality of the selected em-bedding model and the heuristics of the chunking strategy.

Granular text segmentation can occasionally fracture complex concepts across multiple chunks. Furthermore, processing exceptionally large document repositories scales the time re-quired for initial ingestion, and hosting robust open-weight LLMs locally necessitates significant hardware overhead, par-ticularly concerning VRAM capacity and GPU acceleration.

Figure 3: Chat Interface Output

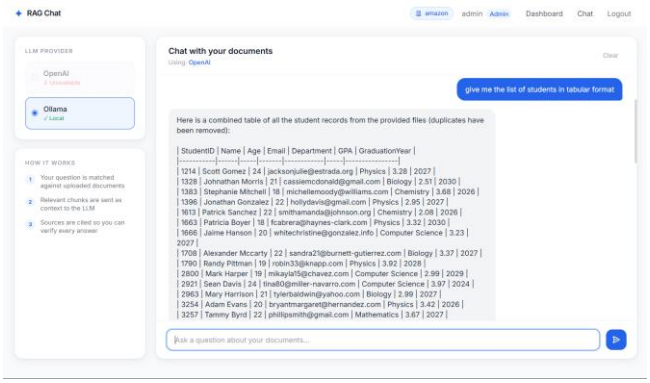
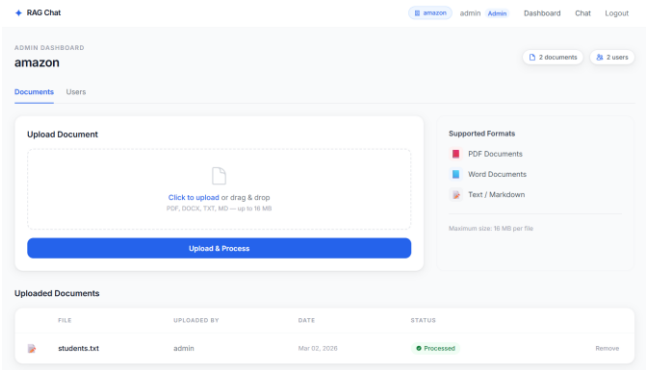


Figure 4: Admin Dashboard and Document Upload



VII. CONCLUSION

The Intelligent Document Question Answering System suc-cessfully demonstrates the transformative efficacy of a modu-lar Retrieval-Augmented Generation (RAG) architecture. By synergizing high-dimensional semantic embeddings, persis-tent vector similarity search via ChromaDB, and the advanced generative capabilities of Large Language Models, the frame-work successfully bridges the gap between static document storage and interactive knowledge synthesis.

The implementation conclusively validates that anchoring generative models to explicitly retrieved evidence systemati-cally mitigates hallucination, yielding responses that are both context-aware and factually verifiable. Furthermore, the sys-tem's inherent modularity ensures extensive configurability, allowing institutions to dynamically balance the computa-tional speed of cloud-based APIs against the stringent privacy guarantees of localized deployments.

Beyond the immediate technical achievements, the adapt-ability of this system positions it as a foundational tool for enterprise knowledge management and academic research. As unstructured data continues to grow exponentially, the reliance on transparent, citation-backed AI models will become a non-negotiable standard. Future iterations of this architecture could explore integrating multimodal embeddings to process images, graphs, and tables seamlessly alongside text, or im-plementing dynamic chunking algorithms that adaptively split documents based on semantic boundaries rather than fixed token limits.

Ultimately, this architecture provides a highly scalable, domain-adaptive, and transparent solution for the future of intelligent document extraction and comprehensive knowledge retrieval.

VIII. REFERENCES

[1] S. Mukherjee, S. Shabnam, S. Hasan, and D. Ajitha, "Enhancing Retrieval Augmented Generation Systems Using AI Models and Graph Databases," 2025 Interna-tional Conference on Emerging Smart Computing and Informatics (ESCI), IEEE, March 2025.

[2] M. Yue, "A Survey of Large Language Model Agents for Question Answering," arXiv preprint arXiv:2503.xxxxx, March 24, 2025.

[3] R. Singh et al., "A Multilingual Intelligent Document Question-Answering System," OpenReview submission, Indian Institute of Science, June 2025.

[4] K. Muludi, K. M. Fitria, J. Triloka, and Sutedi, "Retrieval-Augmented Generation Approach: Document

ISSN 2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)

Vol 26 Issue 03(March),2026

Received: 19-02-2026 Accepted: 23-03-2026 Published: 31-03-2026

Question Answering using Large Language Model," In-ternational Journal of Advanced Computer Science and Applications (IJACSA), vol. 15, no. 3, 2024.

[5] J. S. R. S. Swedhaa, A. N. L. Abraji, and A. Jeyanthi, "LangChain Based Question Answer System," Journal of Novel Research and Innovative Development (JNRID), vol. 2, no. 5, May 2024.