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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15552

AI-Powered Document Question Answering System Using Agentic RAG and Local Language Models

Varshitha D[1], Vidya S[2]

Department of MCA, BIT, K.R. Road, V.V. Puram, Bangalore, India[1]

Assistant Professor, Department of MCA, BIT, K.R. Road, V.V. Puram, Bangalore, India[2]

Abstract: The rapid growth of digital information and unstructured data has transformed the way users search, analyze, and interact with documents. However, traditional search systems often lack contextual understanding, intelligent reasoning, and the ability to generate precise answers from multiple sources. This paper presents DocAgent, an AI-powered Agentic Retrieval-Augmented Generation (RAG) system designed to provide accurate and context-aware responses from user-provided documents and web content. The proposed system integrates document ingestion, semantic search, and local large language model processing to enable efficient knowledge retrieval. Documents and URLs are processed into vector embeddings and stored in a vector database, allowing the system to perform similarity-based retrieval. An agent-based workflow dynamically decides whether to retrieve relevant context, refine the query, or directly generate responses, improving accuracy and reducing irrelevant outputs. The system utilizes a locally hosted language model through Ollama, ensuring privacy, cost efficiency, and reduced dependency on external APIs. The application is evaluated using real-time query scenarios, demonstrating improved response relevance, reduced hallucinations, and enhanced user interaction compared to traditional document search systems.

Keywords: Artificial Intelligence, Retrieval-Augmented Generation, Agentic AI, Document Question Answering, Ollama, Vector Database, Semantic Search, Web-Based System

I. INTRODUCTION

The rapid growth of digital documents and online information has created significant challenges in efficiently retrieving and understanding relevant data. Large volumes of unstructured content such as research papers, reports, and web articles are generated daily, making it difficult for users to extract meaningful insights quickly. Traditional search systems rely heavily on keyword matching and often fail to capture the context and intent of user queries, resulting in incomplete or irrelevant information. As a result, users are required to manually analyze documents, which is time-consuming and inefficient.

Despite advancements in artificial intelligence, many existing systems lack intelligent decision-making, multi-source data integration, and context-aware response generation. Standalone language models may produce responses without grounding in actual document content, leading to inaccuracies and reduced reliability. Additionally, the absence of semantic search and adaptive query processing mechanisms limits the effectiveness of current solutions.

1.1 Project Description

This project focuses on the design and implementation of DocAgent, an Agentic Retrieval-Augmented Generation (RAG) application that enables efficient interaction with documents and web-based content. The system allows users to upload documents or provide URLs, which are processed, segmented into smaller chunks, and converted into vector embeddings. These embeddings are stored in a vector database, enabling fast and accurate semantic retrieval of relevant information.

1.2 Motivation

The increasing volume of digital information has made it challenging for users to efficiently extract relevant knowledge from documents. Traditional search methods are often slow and ineffective, as they do not understand the intent behind user queries or provide context-based answers. This leads to increased effort, time consumption, and reduced productivity, especially in academic and professional environments.

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15552

II. RELATED WORK

Paper [1] presents a retrieval-augmented generation model that integrates document retrieval with large language models to provide accurate and context-aware responses, demonstrating improved performance in knowledge-intensive question answering tasks.
Paper [2] proposes a semantic search system using vector embeddings to retrieve relevant information from large document collections, highlighting the effectiveness of similarity-based retrieval over traditional keyword-based approaches.
Paper [3] discusses the use of agent-based workflows in intelligent systems, emphasizing dynamic decision-making for query processing, including retrieval, query refinement, and response generation to improve overall system adaptability.
Paper [4] reviews existing document question answering and AI-based information retrieval systems, focusing on scalability, data privacy, and the importance of grounding responses in real data to reduce hallucinations.
Paper [5] introduces a cloud-based document analysis system that integrates document ingestion, indexing, and query processing, with emphasis on efficient data handling, performance optimization, and improved user interaction.

III. METHODOLOGY

A. System Architecture

The proposed system follows a modular and layered architecture designed to support efficient document processing, semantic retrieval, and intelligent response generation. The architecture is divided into three main layers: presentation layer, application layer, and data layer.The presentation layer consists of a web-based interface built using Streamlit, allowing users to upload documents, provide URLs, and interact through natural language queries. The application layer handles core functionalities such as document ingestion, text processing, embedding generation, and agent-based query handling. The data layer manages vector storage using Qdrant, enabling efficient similarity-based retrieval.
The system architecture integrates the following core components:
• User Interface Module
• Document Loader and Text Processing Module
• Embedding Generation Module
• Vector Database (Qdrant)
• Agent Controller and LLM Engine

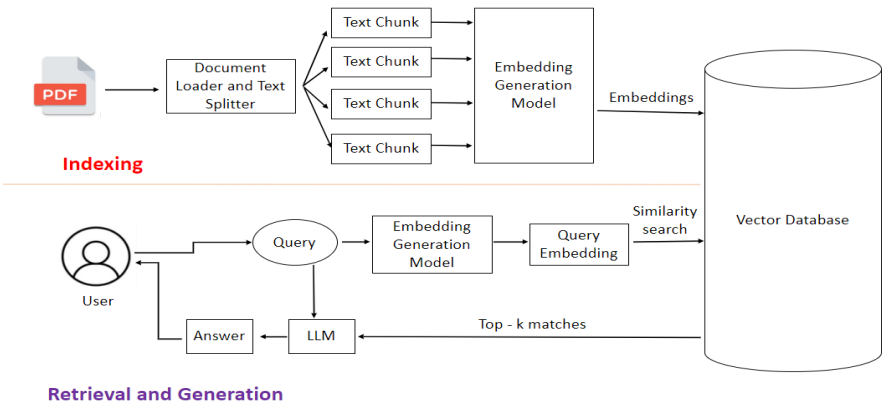

Fig. 1. Flowchart of methodology

B. DOCUMENT LOADER AND TEXT PROCESSING MODULE

The system allows users to upload multiple document formats such as PDF, Word, and text files, as well as provide web URLs. The Document Loader extracts raw content from these sources. The extracted content is then processed using a Text Splitter, which divides large documents into smaller chunks. This step ensures efficient handling of large data and improves retrieval accuracy. The processed chunks are then passed to the embedding generation stage.

C. EMBEDDING GENERATION MODULE

Each text chunk is converted into vector embeddings using a locally hosted embedding model powered by Ollama. These embeddings represent the semantic meaning of the content. The generated embeddings are stored in a vector database (Qdrant), enabling fast and efficient similarity-based search. This step forms the indexing phase of the system, as shown in the upper part of the diagram.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15552

D. RETRIEVAL AND GENERATION MODULE

When a user submits a query, the system converts the query into an embedding using the same embedding model. The query embedding is then compared with stored document embeddings using similarity search techniques. The system retrieves the top-k most relevant document chunks from the vector database. These retrieved chunks are passed to the LLM (via Ollama), which generates a context-aware response.

E. IMPLEMENTATION FLOW

1. User uploads documents or provides URLs.
2. System loads and splits documents into text chunks.
3. Embeddings are generated and stored in the vector database.
4. User submits a query through the interface.
5. Query is converted into embedding and matched with stored vectors.
6. Relevant document chunks are retrieved.
7. LLM generates a response based on retrieved context.
8. Final answer is displayed to the user.

F. HARDWARE AND SOFTWARE COMPONENTS

Hardware Components:

A standard system with Intel Core i5 or equivalent processor, 8 GB RAM, stable internet connection, and a web-enabled device is sufficient for running the application.

Software Components:

The system is implemented using Streamlit for frontend development, Python for backend processing, Qdrant for vector storage, and Ollama for local LLM and embedding generation. Additional libraries for document processing and text handling are used to support multi-source data ingestion.

IV. SYSTEM IMPLEMENTATION AND EVALUATION

A. System Architecture and Deployment

The proposed system follows a modular client–server architecture integrated with a local LLM runtime. The frontend provides an interactive web interface built using Streamlit, allowing users to upload documents, provide URLs, and perform queries.The backend manages document processing, embedding generation, retrieval operations, and response generation. The system uses Qdrant as the vector database and Ollama for local embedding and LLM execution, ensuring efficient and cost-effective deployment without dependency on external APIs.

B. Document Processing Implementation

The document processing module enables users to upload files such as PDF, Word, and text documents, as well as provide web URLs. The system extracts raw content and processes it using a text splitter, dividing large content into smaller chunks.These chunks are then converted into vector embeddings using a local embedding model.

C. Retrieval and Query Processing Module

The retrieval module handles user queries by converting them into vector embeddings. The system performs similarity search in the vector database to identify relevant document chunks.The top-k relevant results are retrieved and passed to the LLM, which generates context-aware responses.

D. User Interaction and Interface Implementation

The user interface provides options to upload documents, add URLs, and initiate queries. It also supports chat-based interaction, allowing users to ask multiple questions within the same session.Additional features such as chat history management, export options, and document indexing status enhance usability and provide a smooth user experience.

E. Performance Evaluation Metrics

The system performance is evaluated using the following metrics:

- Retrieval Accuracy – Relevance of retrieved document chunks to the query
- Response Generation Time – Time taken to generate answers
- Query Processing Efficiency – Speed of embedding and similarity search
- System Usability – Ease of interaction and user experience

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15552

F. Experimental Setup and Results Analysis

The system was tested using multiple documents and web URLs across different domains. Various queries were executed to evaluate retrieval accuracy and response quality.The results show that the system effectively retrieves relevant information and generates accurate, context-aware responses.

G. Results and Observations

1.Initial Configuration Screen
2.Main DAshboard Interface
3.Indexing Status Display

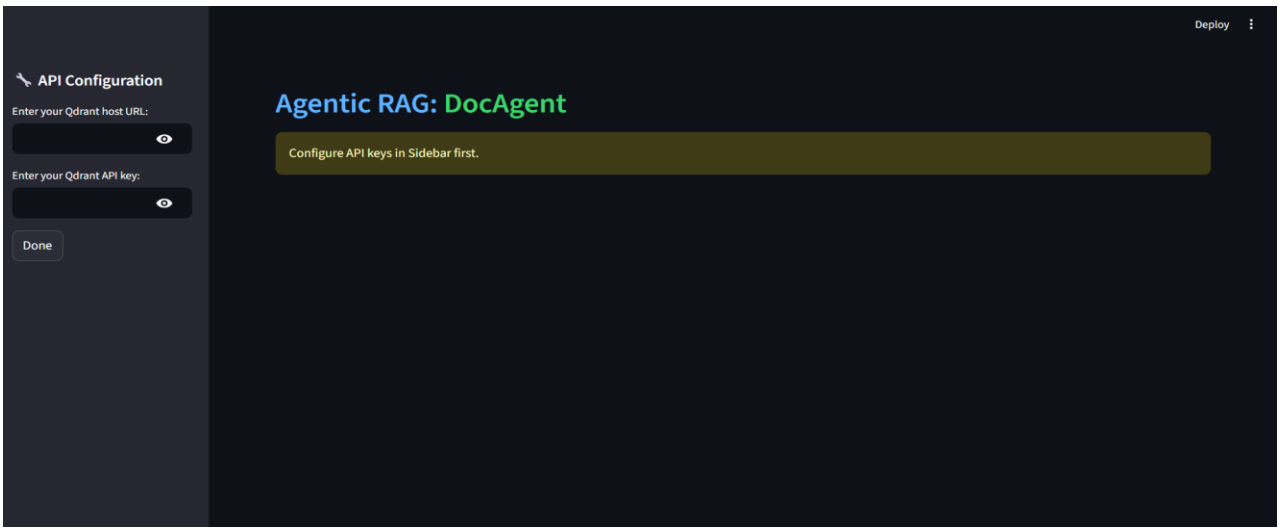

Fig. 2. Initial Configuration Screen

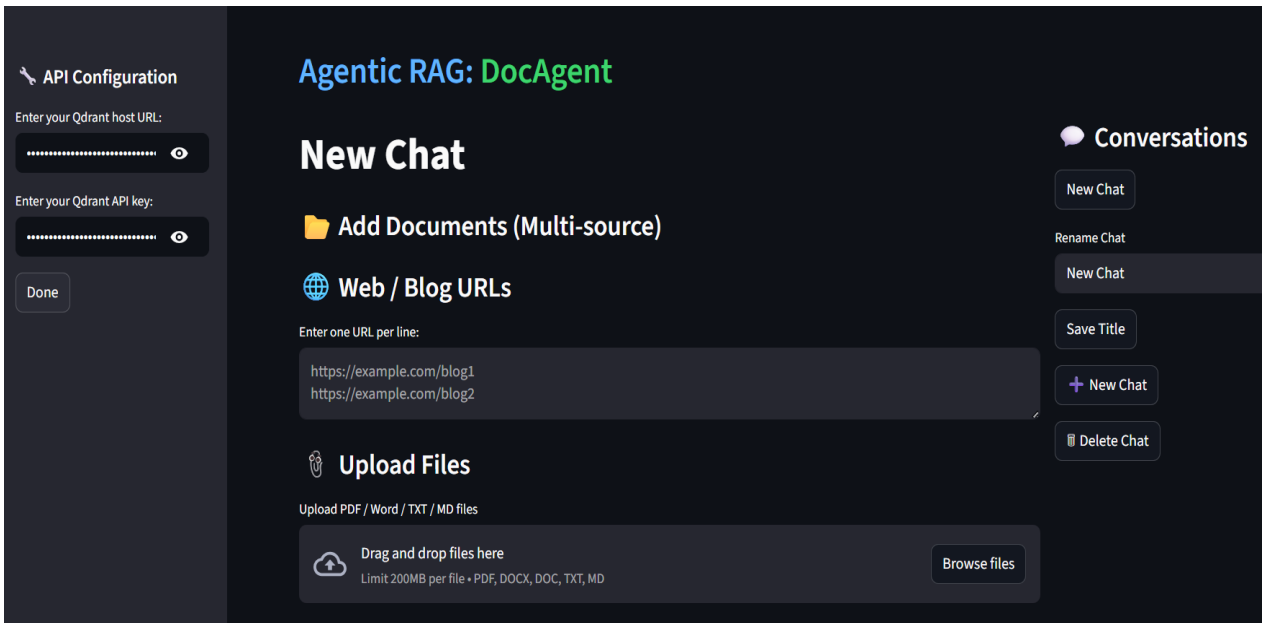

Fig. 3. Main Dashboard Interface

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15552

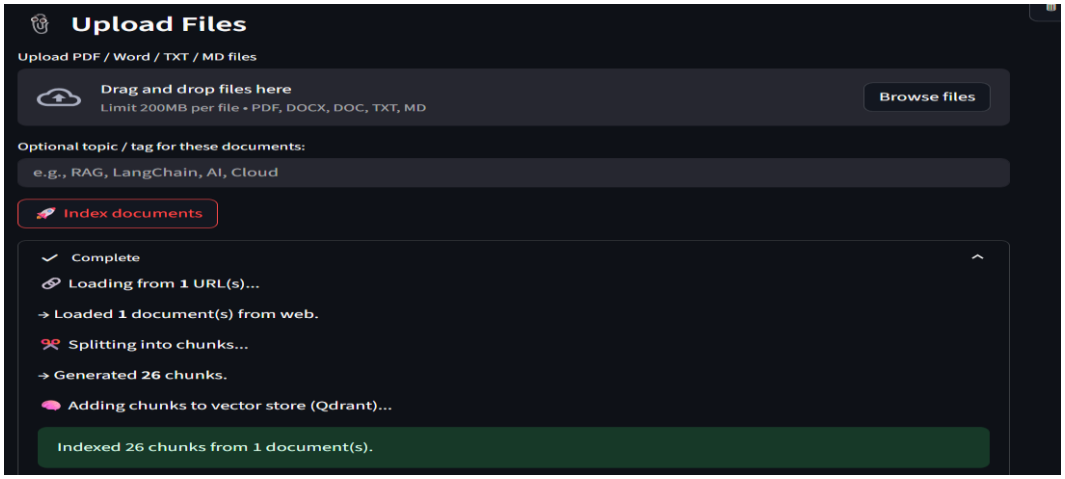

Fig. 4. Indexing Status Display

V. RESULTS AND DISCUSSION

The experimental evaluation of the DocAgent system demonstrates its effectiveness in handling document-based question answering using an agentic retrieval-augmented generation approach. The system was tested using multiple document types, including PDF files, text documents, and web URLs, along with a variety of natural language queries to assess its performance under different scenarios.

The document ingestion module successfully processed uploaded files and URLs by converting them into structured text and dividing them into smaller chunks. These chunks were then transformed into vector embeddings and stored in the Qdrant vector database. The indexing process showed consistent performance, with minimal delay even when handling larger documents. The system accurately reported the number of chunks generated, ensuring transparency in data processing.

VI. CONCLUSION

The DocAgent – Agentic RAG Application successfully demonstrates the design and development of an intelligent information retrieval system that enhances the way users interact with documents and web-based content. The system addresses the limitations of traditional keyword-based search methods by introducing a context- aware, retrieval-augmented generation approach that delivers accurate and meaningful responses to user queries.

Throughout the project, modern technologies were effectively utilized to build a modular and scalable application. The integration of a vector database enables efficient semantic retrieval, while the use of a locally hosted language model ensures secure and reliable response generation without dependency on external cloud services. The agentic workflow allows the system to retrieve relevant information dynamically and generate responses grounded in actual data sources.

VII. FUTURE WORK

The current version of the DocAgent – Agentic RAG Application successfully implements core functionalities such as document ingestion, URL processing, semantic retrieval, and response generation. While the system meets its primary objectives, there is significant scope for enhancement to improve functionality, scalability, and user experience.
One potential enhancement is the integration of advanced document analytics. Future versions of the system can include features such as document summarization, keyword extraction, and topic classification.

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,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2]. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai and J. Sun, “Retrieval-Augmented Generation for Large Language Models: A Survey,” *arXiv preprint arXiv:2312.10997*, 2023.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15552

[3]. M. Klesel and H. Wittmann, "Retrieval-Augmented Generation (RAG)," *Business & Information Systems Engineering*, vol. 67, no. 2, pp. 1–8, 2025.
[4]. P. Zhao, H. Zhang, J. Yu and L. Chen, "Retrieval-Augmented Generation for AI-Generated Content: A Survey," *International Journal of Artificial Intelligence*, 2026.
[5]. A. Brown, M. Roman and B. Devereux, "A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges," *arXiv preprint*, 2025.
[6]. Z. Li, "Retrieval-Augmented Generation for Educational Applications," *Computers and Education: Artificial Intelligence*, vol. 6, 2025.
[7]. A. Fink, "Retrieval-Augmented Generation with Large Language Models," *Radiology: Artificial Intelligence*, vol. 7, no. 1, 2025.
[8]. M. R. Hajar, "A Systematic Review of Retrieval-Augmented Generation," *Journal of Applied Computing and Software Technology*, vol. 5, no. 1, 2025.
[9]. T. E. Kim, "On the Impact of Fair Ranking in Retrieval-Augmented Generation," in *Proceedings of the ACM Conference on Information Retrieval*, 2025.
[10]. P. Jiang, "A Comparative Study on Retrieval-Augmented Generation and Chain-of-Thought Reasoning," *Engineering Applications of Artificial Intelligence*, 2025.
[11]. M. Klesel, H. Wittmann and F. Müller, "A Retrieval-Augmented Generation Framework for Organizational Knowledge Systems," *Springer Journal of Information Systems*, 2025.
[12]. E. Karakurt, "Retrieval-Augmented Generation and Large Language Models: A Review," *Preprints*, 2025.
[13]. M. Klesel and H. Wittmann, "Applications and Challenges of Retrieval-Augmented Generation," *ResearchGate Publication*, 2025.
[14]. H. Yu, A. Gan and S. Li, "Evaluation of Retrieval-Augmented Generation Systems," *Proceedings of Machine Learning Research*, 2024.
[15]. Y. Hu, Z. Lei and Z. Zhang, "GRAG: Graph Retrieval-Augmented Generation," *arXiv preprint arXiv:2405.16506*, 2024.
[16]. A. Y. Aytar, K. Kilic and K. Kaya, "A Retrieval-Augmented Generation Framework for Academic Literature Navigation," *arXiv preprint*, 2024.
[17]. M. Klesel and H. Wittmann, "Enhancing Information Retrieval using Retrieval-Augmented Generation," *ResearchGate Publication*, 2025.

