CONFERENCE PAPER
World Conference on Science, Innovation and Human Progress 2026

Accepted: July 6, 2026 | **Published:** July 28, 2026 | **Language:** English

FUTURITY PUBLISHING

Section of the conference: *Computer science, computing and automation*
Sekcja konferencji*: Informatyka, techniki komputerowe i automatyka*

AskYourDocs: A Retrieval-Augmented Generation (RAG) Powered Document Assistant for Intelligent Information Retrieval

Dr. V. Neelima[1], Panuganti Varshini[2], Shaik Zohaib Ur Rahman[2], Purella Sravya[2], Mohammed Ansaar Ahmed[2]

[1]Professor and Head of Department, Department of CSE (AI&ML), Jyothishmathi Institute of Technology and Science, Karimnagar, Telangana, India
[2]UG Students, Department of CSE (AI&ML), Jyothishmathi Institute of Technology and Science, Karimnagar, Telangana, India

✉ **Corresponding author:** Dr. V. Neelima, Department of CSE (AI&ML), Jyothishmathi Institute of Technology and Science, Karimnagar, Telangana, India

ABSTRACT: The rapid growth of electronic documents in academic, research, legal and enterprise environments has made effective information retrieval progressively more difficult. Conventional keyword-based search approaches are unable to understand the semantic meaning and the contextual intent of the user query, leading to incomplete or irrelevant results. In this paper, we introduce AskYourDocs, a RAG-based intelligent document assistant that enables users to query documents in natural language. The system accepts multiple document formats such as PDF, DOCX, and TXT. It processes the documents by cleaning the text and splitting it into smaller chunks. These chunks are transformed into semantic embeddings using Sentence-Transformer models and stored in a vector database, FAISS. When a user query comes in, the system performs a vector similarity search to retrieve the most relevant document segments and uses them as grounding context for the Google Gemini Large Language Model (LLM) to generate accurate, document-grounded responses. Voice input and text-to-speech output are also integrated into the system for better accessibility. The system is implemented through an interactive interface built with Streamlit. AskYourDocs combines semantic retrieval and generative AI to reduce hallucinated responses and improve the reliability of document-based question answering, providing a practical solution for education, research, legal, healthcare and enterprise use cases.

KEYWORDS: Retrieval-Augmented Generation, Semantic Search, Vector Database, FAISS, Sentence Transformers, Large Language Model, Natural Language Processing, Document Assistant, Google Gemini, Streamlit.

How to cite: Neelima, V., Varshini, P., Rahman, S. Z. U., Sravya, P., & Ahmed, M. A.(2026). AskYourDocs: A Retrieval-Augmented Generation (RAG) Powered Document Assistant for Intelligent Information Retrieval. *World Conference on Science, Innovation and Human Progress*. Futurity Research Publishing. https://doi.org/10.5281/zenodo.21672590

Introduction

The rapid digital transformation of academic, professional and organizational content has resulted in large volumes of unstructured documents such as reports, manuals and research papers. The process of extracting specific information from these documents is a time-consuming manual process and traditional keyword-based search systems often do not understand the underlying context and semantic meaning of a user's query resulting in inaccurate or incomplete retrieval results.

To address these issues, in this work, we propose AskYourDocs, an AI-powered document assistant based on the Retrieval-Augmented Generation (RAG) paradigm. RAG is a hybrid of semantic retrieval and generative language modelling, which enables a system to ground its responses in the real content of the uploaded documents, rather than the parametric knowledge of the underlying language model. This grounding mechanism removes the possibility of hallucinated or fabricated answers, a known limitation of stand-alone large language models.

AskYourDocs allows users to upload documents in PDF, DOCX, or TXT format and have a chat with them. The system extracts and sanitizes the textual content, splits it into manageable chunks, converts these chunks into semantic vector embeddings, and stores these embeddings in a vector database for efficient similarity-based retrieval. When the system receives a natural language query, it retrieves the most contextually relevant chunks and sends them to a large language model to generate a coherent, context-aware answer. The platform also includes voice interaction with speech-to-text input and text-to-speech output, making it more accessible to a wider range of users.

The main motivations for this work are (i) reducing the manual effort needed to search large document collections, (ii) improving the contextual accuracy of retrieved information beyond simple keyword matching, (iii) enabling querying across documents from a single interface, and (iv) providing an accessible, voice-enabled interaction model suitable for educational, research, legal, healthcare and enterprise environments.

A. Contributions of This Work

The following list summarises the main contributions of this paper, which are a direct consequence of the design and implementation of the AskYourDocs system:

- A modular Retrieval-Augmented Generation pipeline that combines multi-format document ingestion, Sentence-Transformer embeddings, FAISS-based vector retrieval, and Google Gemini LLM generation to produce document-grounded answers.
- A semantic retrieval mechanism that reduces dependence on exact keyword matching by identifying document segments based on contextual similarity rather than literal word overlap.
- Ability to query multiple documents at once through a single conversation interface without having to cross-reference information manually across multiple files.
- Add support for optional voice input and text to speech output and add support for multiple languages (English, Hindi, Telugu) to improve accessibility.
- A lightweight Streamlit based user interface needing no deep technical expertise, plus auxiliary utilities like jumbled-text correction, symbol/character table generation and chat export.

Literature Review

The combination of Large Language Models (LLMs) and Retrieval-Augmented Generation has had a significant impact on recent research into intelligent document question-answering systems. In this section, we review related work that inspired the design of AskYourDocs.

Singh et al. (2025) propose an agentic RAG framework with autonomous reasoning agents in the retrieval pipeline for dynamic retrieval and reasoning steps to address hallucination and improve factual consistency of LLM outputs. This notion of retrieval augmented with controlled reasoning inspired AskYourDocs' design of restricting LLM responses to the context of the retrieved documents.

Shan (2024) proposed OpenRAG, a modular open-source RAG architecture for customized learning applications, which is scalable and integrates with vector databases flexibly. The layered architecture used for AskYourDocs reflects the modular design philosophy of separating document processing, embedding generation, retrieval, and generation into separate components.

Heydari et al. (2024) proposed a Context Awareness Gate (CAG) to filter out irrelevant retrieved content before sending it to the language model for reduced hallucination and improved factual reliability. This principle of controlled, filtered retrieval is consistent with the context selection used in AskYourDocs, which is based on similarity search.

Mengmeng et al. (2024) presented a multi-stage semantic filtering mechanism to refine search results prior to generation, which enhanced the retrieval precision. This underscores the significance of an optimized vector similarity search stage, which constitutes a core part of the AskYourDocs pipeline.

Gijre et al. (2024) examined the joint use of knowledge graphs and retrieval-augmented generation to enhance interpretability and contextual matching in domain-specific question answering. Although AskYourDocs applies vector retrieval instead of knowledge graphs, this research creates avenues for enhancement.

The research conducted by Putri et al. (2024) assessed strategies for streamlining the generation of embedded data in RAG-based systems in order to lower computational burden whilst maintaining the efficacy level, as indicated by the implementation of a slim Sentence-Transformer embedding engine in AskYourDocs, making it usable even in real-time interactions.

Tural et al. (2024) wrote on various design approaches for ensuring integration of various retrieval components with LLMs. These include embedding matching, query rewriting, and document re-ranking. They are seen in AskYourDocs, where query-embedding, vector-similarity-search, and context-based generation techniques are used.

Other key contributions in this field include Lewis et al. (2020), who first described a combination of a neural retriever and a sequence-to-sequence generator in the retrieval-augmented generation framework; Reimers and Gurevych (2019), whose Sentence-BERT architecture is the basis of applied models for obtaining sentence embeddings, including for the all-MiniLM-L6-v2 model from AskYourDocs; as well as Johnson et al. (2021), whose work on a method for efficient similarity search laid the computational foundation for the FAISS library applied in the system for vector storage and retrieval.

To sum up the articles analyzed so far, RAG technology plays a positive role in enhancing the accuracy of

information provided by documents when answering user queries. Nevertheless, the majority of implementations of this technology are limited to one specific area of application, do not support different document formats, and do not even prioritize the ease of use of their service. AskYourDocs builds on these contributions by combining semantic embedding generation, vector database retrieval, and LLM-based generation within a multi-format, voice-enabled, user-accessible system.

Existing System and Problem Statement

A. Existing System

Conventional document management and retrieval systems rely predominantly on manual reading or keyword-based search. Standard PDF readers and basic search tools match exact words rather than the semantic intent behind a query, and cannot generate conversational, context-aware responses. When multiple documents must be searched, users are required to manually cross-reference content, which increases workload and reduces productivity. Furthermore, standalone LLMs used without a retrieval mechanism may generate responses not grounded in actual document content, resulting in hallucinated or misleading outputs.

B. Limitations of Existing Systems

- Time-consuming manual document analysis.
- Keyword-based search lacks semantic understanding.
- Difficulty retrieving information across multiple documents simultaneously.
- Absence of context-aware or conversational response generation.
- Increased likelihood of hallucinated answers in standalone AI systems without document grounding.
- Inefficient handling of large volumes of unstructured data.

C. Problem Statement

There is a need to design and implement an intelligent document assistant using Retrieval-Augmented Generation that allows users to query documents in natural language, retrieves relevant sections through semantic similarity search, generates accurate and document-grounded responses, supports multiple document formats, and provides scalable, user-friendly interaction suitable for academic and professional use.

Proposed System and System Architecture

A. Proposed System

AskYourDocs addresses the above limitations through a RAG-based pipeline that integrates document preprocessing, semantic embedding, vector-based retrieval, and LLM-based response generation. Uploaded documents (PDF, DOCX, TXT) are parsed to extract raw text, which is cleaned and divided into overlapping chunks using a recursive character-based text-splitting strategy. Each chunk is transformed into a semantic vector embedding using a Sentence-Transformer model (all-MiniLM-L6-v2) and stored in a FAISS vector database.

When a user submits a natural language query, the same embedding model converts the query into a vector representation. The system performs a similarity search against the stored embeddings to retrieve the top-k most relevant chunks. These retrieved chunks, together with the original query, are passed as contextual input

to the Google Gemini LLM, which generates a response grounded in the retrieved document content. The final response is displayed through a Streamlit-based chat interface and can optionally be converted to speech using a text-to-speech module.

B. System Architecture

The system architecture consists of the following functional modules:

- Document Upload Module – Accepts PDF, DOCX, and TXT files through the Streamlit interface.
- Text Extraction and Preprocessing Module – Extracts and cleans raw text using format-specific loaders (e.g., PyPDFLoader for PDF, docx2txt for DOCX).
- Text Chunking Module – This module divides the extracted text into overlaps by using a recursive character text splitter that allows for contextual continuity in the text chunks.
- Embedding Generation Module – This module turns text chunks into high-dimensional semantic vectors with a Sentence-Transformer embedding model.
- Vector Database Module – This module saves and organizes the embeddings in such a way that makes it easy to find the similarities between them with FAISS.
- Module of Query Processing and Retrieval – Converts user queries into an embedding before carrying out a similarity search to find relevant portions.
- Response Generation Module – Provides the Google Gemini LLM with collected information along with the user query so it can produce a context-aware, document-grounded answer.
- Voice Interaction Module – Facilitates speech query input and text-to-speech output for the generated answers.

Fig. 1.

Architecture of AskYourDocs system.

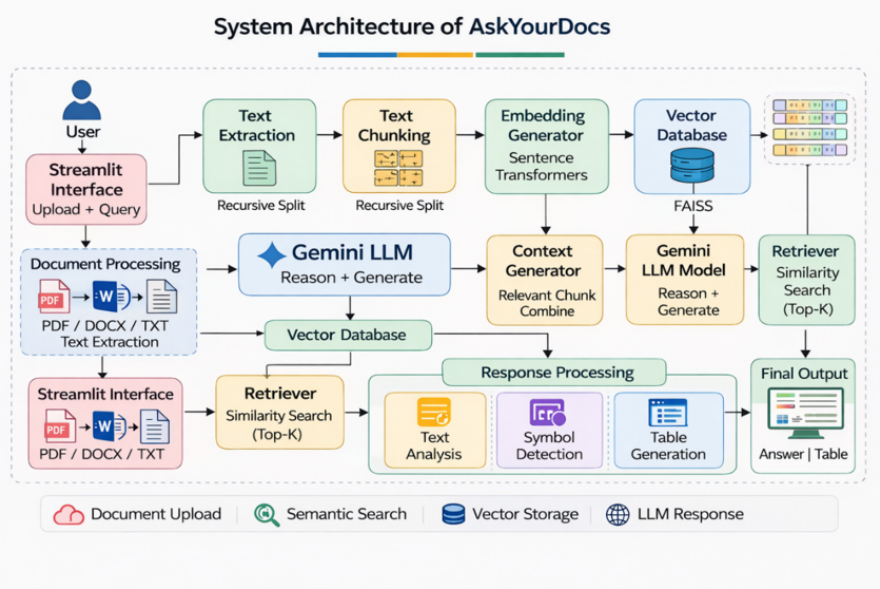
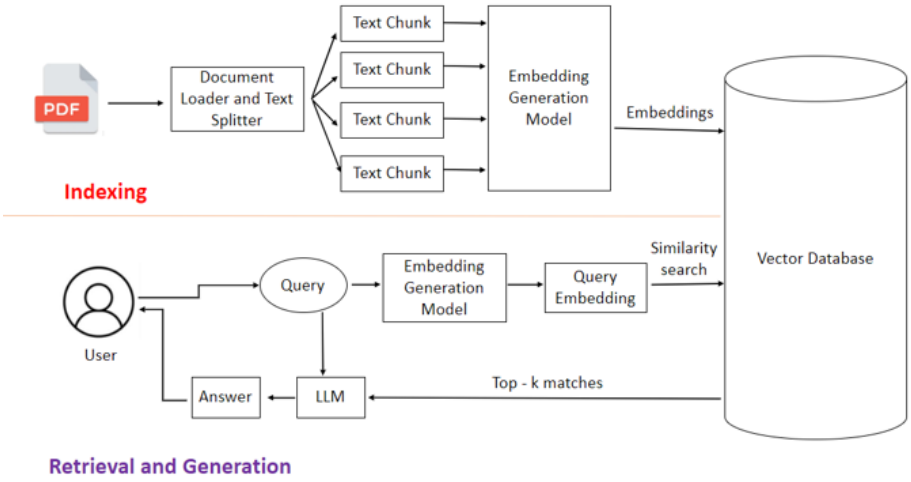


Fig. 2.

Data flow diagram for indexing and generation retrieval phases.



The modular design allows for the decoupling of the retrieval and generation functions, thereby allowing the system to anchor the results of the large language model in reality based on the verifiability of the documents.

C. Mathematical Formulation

AskYourDocs' retrieval-augmented generation (RAG) algorithm can be described in the following manner. Let there be a document split into chunks c1, c2,...cN. Every chunk undergoes the transformation from a sentence to a d-dimensional embedding vector using the Sentence-Transformer embedding function f:

$$e_i = f(c_i), \quad i = 1, 2, \ldots, N \tag{1}$$

Likewise, the user query q is mapped to the same space of embeddings:

$$e_q = f(q) \tag{2}$$

The relevance between the query embedding and each chunk embedding is measured using cosine similarity, which quantifies the angular closeness between two vectors irrespective of their magnitude:

$$\mathrm{sim}(e_q, e_i) = (e_q \cdot e_i) / (\|e_q\| \, \|e_i\|) \tag{3}$$

where $e_q \cdot e_i$ denotes the dot product of the query and chunk embeddings, and $\|\cdot\|$ denotes the Euclidean norm. A similarity value closer to 1 indicates greater semantic closeness between the query and the chunk.

The FAISS vector database ranks all indexed chunks by similarity and returns the top-k most relevant segments:

$$C_{top\text{-}k} = \arg\,\text{top-}k \{ \mathrm{sim}(e_q, e_i) \mid i = 1, \ldots, N \} \tag{4}$$

The last step involves concatenation of the obtained top-k chunks, $C_{top\text{-}k}$, together with the query, q, to be used as the grounding context for the Gemini language model to obtain the answer, A.

$$A = \mathrm{LLM}(q, C_{top\text{-}k}) \tag{5}$$

By limiting the generation process to be conditioned on Ctop-k instead of conditioning on parametric knowledge only, Equation (5) formulates the grounding process, which avoids hallucinations, according to the principle of retrieval-augmented generation mentioned in the Literature Review section.

System Implementation

A. Hardware and Software Environment

The system has been developed and tested using a standard laptop/desktop computing environment that requires at least 8 GB of RAM (16 GB recommended), a microphone as an input device, loudspeakers or headphones as the output device, and an Internet connection to connect with the cloud-based Gemini API. The software stack includes the following: Python (version 3.8 and above), Streamlit as a web interface, LangChain for RAG process orchestration, FAISS for vector storage, Sentence-Transformers for generating embeddings, Google Generative AI SDK for large language models, and additional libraries like PyPDF, docx2txt, python-dotenv, gTTS, and SpeechRecognition.

B. Pipeline Algorithm

Algorithm 1 summarizes the complete AskYourDocs pipeline. It covers document ingestion and response generation, as shown in the RAGEngine module.

Algorithm 1

Pseudocode of the AskYourDocs retrieval-augmented generation pipeline.

Algorithm 1: AskYourDocs RAG Pipeline

```
Input: Document set D = {d1, ..., dm}; user query q
Output: Generated answer A
for each document d in D do
  text ← ExtractText(d) // PyPDFLoader / docx2txt / plain read
  chunks ← RecursiveCharacterTextSplitter(text, size=1000, overlap=200)
  for each chunk c in chunks do
    ec ← SentenceTransformerEmbed(c)
    FAISS_Index.add(ec, c)
  end for
end for
eq ← SentenceTransformerEmbed(q)
Ctop-k ← FAISS_Index.similaritySearch(eq, k)
context ← Concatenate(Ctop-k)
A ← GeminiLLM.generate(q, context) // RetrievalQA chain
Display(A) on Streamlit chat interface
if voice_output enabled then
  SpeakText(A) // gTTS text-to-speech
end if
return A
```

C. Implementation Workflow

The implementation is done following a systematic workflow that involves:

1. Environment Setup: Setup of Python along with the required libraries and configuration of the code editor.

2. API Configuration: Creation of an API key for Google Gemini using Google AI Studio and saving it through environment variables (.env file) that avoids the exposure of credentials in the source code. The langchain-google-genai library makes use of the connection between the RAG pipeline and the Gemini model.

3. Project Structuring: The structure of the project is based on modularity, where the application file (app.py) is separated from the files for loading the documents, generating embeddings, retrieving the results and working with the voice.

4. Documents Upload and Text Extraction: Documents are uploaded by users in the Streamlit interface and loaders specific to their formats get the text part of the document for further operations.

5. Text Chunking: The RecursiveCharacterTextSplitter splits the extracted text into equal overlapping chunks to maintain the context.

6. Embeddings Creation: The Sentence-Transformer model processes each chunk to create its vector representation.

7. Embeddings Storage: Created embeddings are saved in the FAISS vector store.

8. Query Input: The users ask questions via text or voice input and if there is voice input, it is converted to text by the SpeechRecognition library.

9. Similarity Search: Cosine similarity is used for finding k-nearest chunks based on the similarity of embedding.

10. Response Generation: The found chunks together with the query of the user are used as the context by the Gemini LLM to generate the answer.

11. Answer Presentation and Voice Output: The answer is shown in the chat interface and if it is needed, it can be converted to voice by gTTS.

Fig. 3.

Streamlit-based document upload and chatting interface.

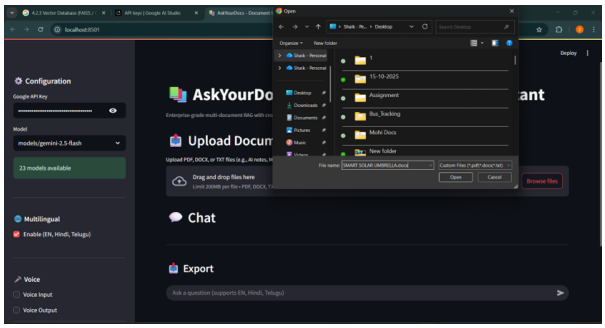


Fig. 4.

Use-case diagram of AskYourDocs.

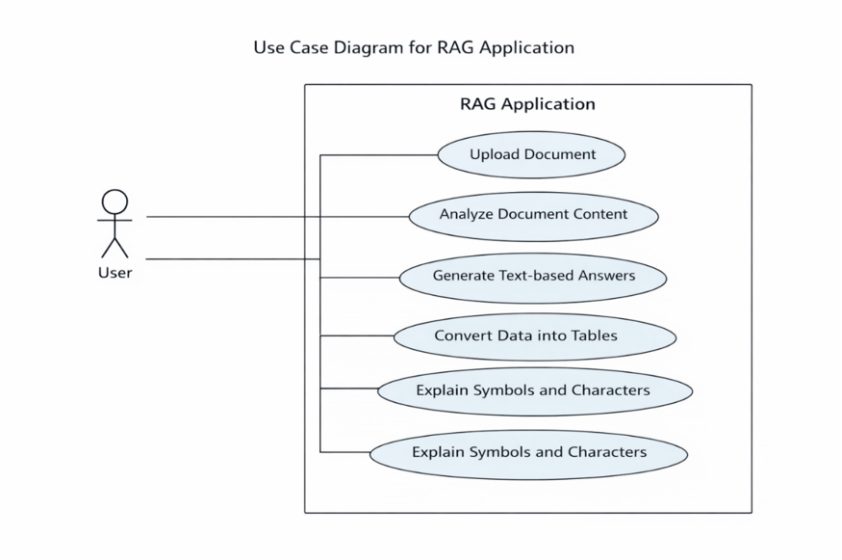
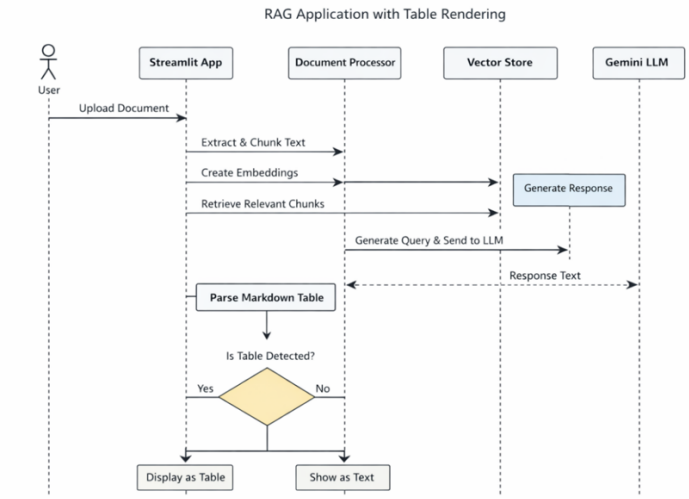


Fig. 5.

Query process and response generation sequence diagram.



D. Core Modules

The architecture of the system consists of distinct modules, which include:

(1) RAGEngine – is a class responsible for document chunking, embeddings generation, vector stores generation, and executing queries via a RetrievalQA pipeline;

(2) document_loader – is a module that extracts text in different formats;

(3) embeddings – is a module that holds HuggingFace Sentence-Transformer model; and

(4) voice_module – is a module responsible for playing audio with texts converted to speech via gTTS.

Furthermore, there are additional modules, such as correction of disordered/corrupted text, generation of symbol/character table based on the raw data, multilingual detection, and exporting the chat log.

Research Results and Discussion

The implemented AskYourDocs solution was evaluated based on observed performance rather than using benchmarking with metrics such as accuracy, precision, recall, and latency, since there were no measurements performed during the development process of this tool. Thus, the following analysis will be focused only on qualitative evaluation of the system's performance and usability.

A. Streamlit Interface

The interface offered an efficient and unified environment where one could set up their Google API key, choose from the Gemini model, upload files, and communicate through the chat box. Multilingual capabilities of the interface were shown by allowing languages like English, Hindi, and Telugu, along with the option to use voice inputs/outputs, without requiring any changes in the backend pipeline itself.

B. Document Upload and Processing

This system was right in accepting PDF, DOCX, and TXT files since they were analyzed through text extraction and chunking and embedding was done without the need for any human effort. This clearly shows that the multi-format ingestion process worked well as expected.

C. Semantic Search and Retrieval Quality

Since both the query and document fragments are mapped to the same vector space, the retrieval can be done based on semantic similarity as opposed to lexical match. This is the main benefit of the RAG framework when compared to keyword search since the system is able to find relevant fragments regardless of whether the wording used by the user is different from the one used in the document.

D. Response Relevance and Hallucination Reduction

With the inclusion of only the extracted and ranked similar documents as the context for the Gemini model, the output was grounded on the source texts instead of being produced using the model's parametric knowledge alone. The approach is consistent with the aims of reducing hallucinations identified in the referenced literature (Singh et al., 2025; Heydari et al., 2024). This is also seen from the ability of the system to provide responses that could be linked back to the documents uploaded.

E. Voice Interaction and Accessibility

Voice inputs and speech outputs served as an accessibility improvement, allowing the users to make their queries vocally and get answers in speech form. This addition improved the usability of the system for those users who prefer to work with auditory feedback.

F. Additional Functional Observations

Apart from these functions, there are some additional auxiliary abilities possessed by this system, such as the ability to fix scrambled or distorted texts, create tables for symbols, numbers, and letters out of disorganized

data, and exporting chat conversations.

Fig. 6

Sample of context-based text response generated using the uploaded document.

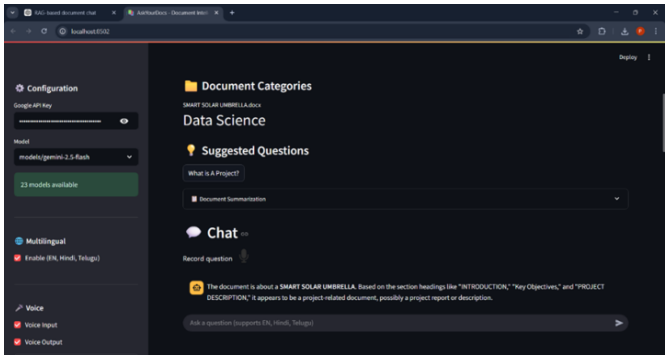
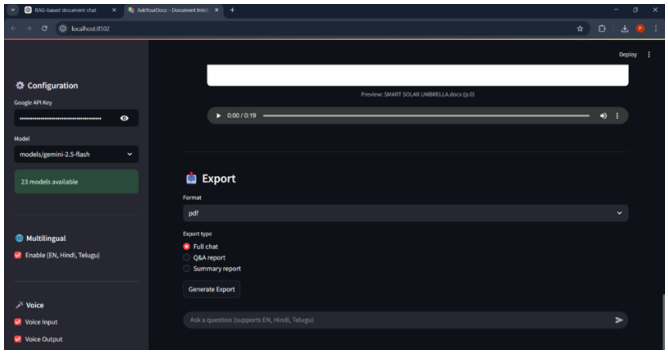


Fig. 7

Sample text-to-speech playback interface for the response generated.



G. Limitations

- No quantitative evaluation (accuracy, precision, recall, F1-score, or latency) was performed; all observations reported are qualitative and based on functional testing during development.
- The mechanism requires a consistent internet connection and the availability of an external Google Gemini API to generate responses, hence making it incompatible for off-line usage.
- The process of dividing information into chunks using overlap can sometimes divide contextually-related data into different chunks, thus reducing the effectiveness of the retrieval.
- The responses generated depend entirely on the language processing and understanding ability of the Google Gemini LLM, which is not under the control of the retrieval framework.
- The current version of the retrieval mechanism uses similarity search based solely on vector space and does not incorporate structured knowledge representation such as knowledge graphs which could further improve contextual precision.
- On the whole, it can be concluded that the RAG-driven approach to AskYourDocs design is an actual improvement of keyword-based search by providing for semantic interpretation of the user requests,

minimizing dependency on non-grounded knowledge of the LLM, and accommodating accessible multi-modal interaction. Mentioned limitations appear to be truly restricting and determine the path for further research.

H. Qualitative Comparison of Retrieval Approaches

The following table highlights a comparative study that is consistent with the above-mentioned observations regarding AskYourDocs, keyword search approach, and a single large language model without using any retrieval system. Note that the table indicates only descriptive characteristics.

Table 1

Comparison of document information retrieval approaches.

Capability	**Traditional Keyword Search**	**Standalone LLM (No Retrieval)**	**AskYourDocs (RAG-Based)**
Understands query semantics/context	No – exact word matching only	Yes, but based only on parametric knowledge	Yes – semantic embedding similarity
Responses grounded in actual uploaded document content	Partial – limited to literal matches	No – not grounded in any external document	Yes – restricted to retrieved chunks
Risk of hallucinated/fabricated answers	Low (no generation occurs)	High	Reduced, via retrieval grounding
Cross-document querying from one interface	Manual cross-referencing required	Not applicable	Yes – single conversational interface
Multi-format ingestion (PDF / DOCX / TXT)	Format-specific tools needed separately	Not applicable	Yes – unified pipeline
Voice input/text-to-speech accessibility	No	No (typically)	Yes – optional voice module
Requires manual effort to locate information	Yes – time-consuming manual reading	N/A	Reduced – automated retrieval

Source: author(s) own development.

Conclusion

The current paper outlines the implementation of a novel document assistant named AskYourDocs, which is a Retrieval-Augmented Generation (RAG) based tool aimed at improving information extraction efficiency and accuracy. In this regard, a combination of multiple steps, including multi-document format ingestion, generating semantic embeddings with the help of Sentence-Transformers, conducting vector similarity search using FAISS, as well as generating the response with the help of Google Gemini LLMs, can be found in the Streamlit framework. The use of RAG as a basis for the grounding procedure reduces the likelihood of getting hallucinated answers since the generation process is confined to contextually relevant documents retrieved. Also, the adoption of voice commands and speech-to-text functionality improves the accessibility of the model.

Some of the future possibilities that could be considered with regards to such a system would be the use of more advanced language models to improve the reasoning abilities of the software; the ability to handle more

file types, such as scanned documents and spreadsheets using OCR methods; better multilingual processing and translation; advanced speech recognition so that the system can accept voice commands; real-time collaboration among users for document analysis; and distributed vector storage for handling massive document corpora. With advances in artificial intelligence technology in knowledge management, there is definite scope for such systems to make knowledge bases from documents an interactive reality.

References

Gijre, S., Agrawal, R., Laddha, P., & Keswani, G. (2024). Enhancing question answering with knowledge graph retrieval and generation using LLMs. In *Proceedings of the 2024 International Conference on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICAIQSA64000.2024.10882212

Heydari, M. H., Hemmat, A., Naman, E., & Fatemi, A. (2024). Context awareness gate for retrieval-augmented generation. In *Proceedings of the 2024 15th International Conference on Information and Knowledge Technology (IKT)* (pp. 260–264). IEEE. https://doi.org/10.1109/IKT65497.2024.10892659

Johnson, J., Douze, M., & Jégou, H. (2021). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)* (pp. 9459–9474). Curran Associates.

Mengmeng, S., Zhibin, L., Qingwei, W., Man, H., & Feiyang, X. (2024). An effective retrieval method to improve RAG performance. In *Proceedings of the 2024 7th International Conference on Data Science and Information Technology (DSIT)* (pp. 1–5). IEEE. https://doi.org/10.1109/DSIT61374.2024.10881380

Putri, M. R., [+ additional authors]. (2024). *Simplification of embedding process in RAG for optimizing QA chatbot*.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Shan, R. (2024). OpenRAG: Open-source retrieval-augmented generation architecture for personalized learning. In *Proceedings of the 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)* (pp. 212–216). IEEE. https://doi.org/10.1109/ICAIRC64177.2024.10900069

Singh, A., Gupta, R., & Kumar, P. (2025). *Agentic retrieval-augmented generation: Enabling autonomous reasoning in RAG systems*. arXiv. https://doi.org/10.48550/arXiv.2501.09136

Tural, B., [+ additional authors]. (2024). *Retrieval-augmented generation (RAG) and LLM integration*.