iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83401

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

DOC GPT: A Retrieval-Augmented Generation Based Intelligent Document Chatbot Using Local Large Language Models

Abhishek R[1], Raviprakash DK[2]
Department of Computer Applications, Bengaluru University, Bengaluru, India
Seshadripuram Academy of Business Studies

Abstract: ***The rapid growth of Artificial Intelligence has changed how users engage with digital documents and information systems. Traditional keyword-based document search systems often do not provide meaningful and context-aware responses. This research presents DOC GPT, an intelligent Retrieval-Augmented Generation (RAG) based chatbot that allows users to interact with uploaded documents using local Large Language Models (LLMs). The system integrates FastAPI, Next.js, FAISS vector indexing, Sentence Transformers, and Ollama-powered LLMs. It creates a privacy-focused and efficient AI assistant that answers user queries from PDF documents. The system performs document ingestion, Optical Character Recognition (OCR), generates semantic embeddings, conducts vector similarity searches, reranks results, and produces context-aware responses. Unlike cloud-dependent AI systems, DOC GPT operates locally, ensuring better privacy, lower latency, and offline access. Experimental evaluation shows improved semantic retrieval accuracy and efficient response generation for academic and enterprise document interactions.***

Keywords: ***Retrieval-Augmented Generation, RAG, Large Language Models, FAISS, Ollama, Semantic Search, FastAPI, Next.js, Artificial Intelligence.***

I. INTRODUCTION

The rapid increase of digital documents in academic, enterprise, and personal settings has raised the need for intelligent systems that can efficiently understand and retrieve contextual information. Conventional document retrieval systems rely heavily on keyword matching and static search methods, which often fail to recognize the semantic relationships between user queries and document content.

Recent developments in Large Language Models (LLMs) have enabled the creation of conversational AI systems that can generate human-like responses. However, standalone LLMs face issues with hallucinations and lack specific contextual awareness. Retrieval-Augmented Generation (RAG) solves this problem by merging semantic retrieval with generative AI capabilities. This paper proposes DOC GPT, a local RAG-based intelligent document chatbot developed to deliver context-aware answers from uploaded documents. The system combines semantic embeddings, vector databases, reranking models, OCR processing, and local LLM inference using Ollama. This architecture allows for secure and effective document interaction without depending on external cloud APIs.

1) The main goals of this project are:
2) Create a conversational AI interface for document interaction.
3) Implement semantic retrieval using vector embeddings.
4) Enable local LLM inference to protect privacy.
5) Support PDF text extraction and OCR processing. • Enhance response accuracy through reranking models.

II. LITERATURE SURVEY

Document retrieval systems have improved considerably in recent years with the growth of Artificial Intelligence and Natural Language Processing technologies. Earlier systems mainly relied on keyword-based searching, where documents were retrieved using exact words or phrases entered by users. Although these methods worked for simple searches, they often failed to understand the actual meaning behind user queries. Because of this limitation, users frequently received incomplete or unrelated search results.

To overcome these issues, researchers began using machine learning and deep learning methods for semantic understanding. Transformer-based models such as BERT and GPT improved the ability of systems to process natural language more effectively. These models helped systems understand sentence meaning, contextual relationships, and similarities between different pieces of text. As a result, intelligent search systems and conversational AI applications became more accurate and practical.

Sentence embedding models further improved semantic retrieval by converting text into dense vector representations. Instead of depending only on keyword matching, vector embeddings allow systems to compare the meaning of sentences mathematically. Models such as Sentence-BERT and BGE embeddings became widely used in semantic search applications because they generate meaningful vector representations for textual information.

Vector databases and similarity search frameworks also became important parts of modern retrieval systems. Facebook AI Similarity Search (FAISS) introduced an efficient approach for storing and retrieving high-dimensional vector embeddings. FAISS supports fast nearest-neighbor search operations, making it suitable for large-scale semantic retrieval tasks. Many Retrieval-Augmented Generation systems now use vector databases to retrieve relevant contextual information before generating responses.

Recent developments in Large Language Models have also changed the field of conversational AI. Models such as GPT, Llama, and Mistral are capable of generating detailed responses that resemble human conversation. However, standalone language models may still produce inaccurate or fabricated information, commonly known as hallucinations. Retrieval-Augmented Generation (RAG) was introduced to reduce this problem by combining semantic retrieval with response generation. In RAG systems, relevant information is first retrieved from documents and then provided to the language model as contextual input. This process improves response quality and reduces irrelevant outputs. Many cloud-based AI assistants currently offer document understanding and conversational search features. However, these systems usually depend on internet connectivity and external cloud infrastructure. This can create concerns related to privacy, response delay, security, and operational cost. Organizations dealing with sensitive or confidential information may not prefer sending their documents to third-party cloud services.

The availability of local inference platforms such as Ollama and open-source Large Language Models has created new opportunities for privacy-focused AI applications. Local deployment allows users and organizations to run AI models directly on their own systems without depending on external services. This approach improves privacy and also enables offline functionality.

Optical Character Recognition (OCR) technologies have also become important in intelligent document systems. Tools such as Tesseract OCR can extract readable text from scanned or image-based PDF documents. OCR processing allows AI systems to understand documents that do not contain selectable digital text.

The proposed DOC GPT system combines these technologies into a single platform. It integrates semantic vector retrieval, OCR-based document processing, reranking mechanisms, and locally hosted language models to create an intelligent document chatbot. Unlike many traditional systems, DOC GPT focuses on privacy-oriented local deployment while maintaining efficient semantic search and context-aware response generation.

III. PROPOSED METHODOLOGY

The proposed DOC GPT system follows a Retrieval-Augmented Generation (RAG) approach to provide context-aware responses from uploaded documents. The system combines document processing, semantic retrieval, vector similarity search, and Large Language Model inference to create an intelligent document chatbot. The overall workflow includes document upload, text extraction, embedding generation, semantic retrieval, reranking, and AI response generation.

A. Document Upload and Processing

The system allows users to upload PDF documents through the frontend interface. Uploaded files are sent to the backend server, where document processing operations are performed. Text-based PDF documents are processed using pdfplumber for direct text extraction. For scanned or image-based PDFs, the system uses pdf2image and Tesseract OCR to extract readable text from document images. After extraction, the document content is divided into smaller text chunks. Chunking improves retrieval efficiency and allows the system to process large documents more effectively. Each chunk is then prepared for semantic embedding generation.

B. Semantic Embedding Generation

The extracted text chunks are converted into dense vector embeddings using Sentence Transformer models. The system uses the BAAI/bge-base-en-v1.5 embedding model for generating semantic vector representations. These embeddings capture the contextual meaning of text instead of relying only on keyword matching.

Semantic embeddings allow the system to compare user queries and document content based on meaning and contextual similarity. This improves retrieval accuracy for natural language questions.

C. Vector Database Storage

Generated embeddings are stored in a FAISS vector database. FAISS provides efficient indexing and similarity search for high-dimensional vectors. The vector database enables fast retrieval of relevant document chunks during user interaction.

Each embedding is associated with its corresponding document content and metadata. This allows the system to retrieve the original text linked to semantically similar embeddings.

D. User Query Processing

When a user submits a query, the question is converted into a semantic embedding using the same embedding model used during document processing. The generated query embedding is compared with the stored document embeddings in the FAISS vector database.

The system retrieves the most relevant document chunks based on semantic similarity scores. This retrieval process helps identify contextual information related to the user query.

E. Context Reranking

After similarity retrieval, the retrieved document chunks are passed through a reranking model to improve relevance. The reranking mechanism prioritizes the most contextually relevant information before sending it to the language model.

This step improves response quality by reducing irrelevant context and increasing retrieval precision for complex user queries.

F. AI Response Generation

The final retrieved context is combined with the user query and passed to a locally hosted Large Language Model through Ollama. Models such as Llama 3.1 are used for generating conversational responses.

The language model uses the retrieved document context to generate accurate and context-aware answers. Since the system operates locally, user documents remain private and are not sent to external cloud services.

G. Frontend and Backend Integration

The frontend of the system is developed using Next.js, React.js, and Tailwind CSS. It provides features such as document upload, conversational chat interface, authentication, conversation history, and voice input support.

The backend is implemented using FastAPI, which manages document processing, embedding generation, authentication APIs, vector retrieval operations, and communication with the Ollama inference engine.

H. Database Management

SQLite is used for storing user information, authentication data, conversation history, and document metadata. The database layer helps manage user sessions and maintain persistent conversation records within the application.

I. System Workflow

The overall workflow of the proposed DOC GPT system can be summarized as follows:

1) User uploads PDF documents.
2) Text is extracted using PDF processing and OCR tools.
3) Document text is divided into chunks.
4) Semantic embeddings are generated for each chunk.
5) Embeddings are stored in the FAISS vector database.
6) User submits a query through the chat interface.
7) Query embeddings are generated and compared with stored vectors.
8) Relevant document chunks are retrieved and reranked.
9) Retrieved context is passed to the Large Language Model.
10) The generated response is displayed to the user through the frontend interface.

IV. SYSTEM ARCHITECTURE

DOC GPT is designed using a modular architecture that combines frontend technologies, backend APIs, semantic retrieval systems, and locally hosted Large Language Models. The architecture is divided into multiple layers to ensure efficient document processing, secure communication, and accurate AI-generated responses.

The frontend layer is the first point of interaction between the user and the system. The frontend is developed using Next.js, React.js, TypeScript, Tailwind CSS, and shadcn/ui components. This layer provides features such as user authentication, document upload, conversational chat interface, conversation history, and voice input support. Communication between the frontend and backend is handled through REST API requests.

The backend layer is implemented using FastAPI. It acts as the central processing unit of the system and manages authentication, document uploads, embedding generation, vector retrieval operations, and communication with the AI inference engine. The backend processes incoming user requests and coordinates the workflow between different modules.

The document processing module extracts textual information from uploaded PDF files. Text-based PDFs are processed using pdfplumber for direct text extraction, while scanned or image-based documents are processed using pdf2image and Tesseract OCR. After extraction, the document content is divided into smaller text chunks to improve retrieval efficiency and embedding generation performance.

The semantic embedding module converts document chunks into dense vector representations using Sentence Transformer models. The system uses the BAAI/bge-base-en-v1.5 embedding model to generate contextual embeddings that capture semantic meaning rather than simple keyword relationships.

Generated embeddings are stored in the FAISS vector database. FAISS supports efficient indexing and fast similarity search operations for high-dimensional vectors. During user interaction, the vector database retrieves document chunks that are semantically related to the user query.

The query processing module converts user questions into embeddings using the same embedding model used during document processing. The generated query vectors are compared with stored vectors in the FAISS database to retrieve the most relevant contextual information. Retrieved chunks are then passed through a reranking mechanism to improve relevance and reduce unnecessary context.

The AI response generation module uses locally hosted Large Language Models through Ollama. Models such as Llama 3.1 process the retrieved context and generate context-aware conversational responses. Since the models are hosted locally, user documents and conversations remain private without dependency on external cloud services.

The database layer uses SQLite for storing user credentials, authentication data, document metadata, and conversation history. This layer supports persistent storage and session management within the application.

The overall workflow of the system can be summarized as follows:

1) User uploads a PDF document through the frontend interface.
2) Backend services process the uploaded document.
3) OCR and text extraction operations are performed.
4) Extracted content is divided into text chunks.
5) Semantic embeddings are generated for each chunk.
6) Embeddings are stored in the FAISS vector database.
7) User submits a query through the chat interface.
8) Query embeddings are generated and compared with stored vectors.
9) Relevant document chunks are retrieved and reranked.
10) Retrieved context is passed to the Ollama-hosted Large Language Model.
11) The generated response is returned to the frontend and displayed to the user.

The modular design of DOC GPT improves scalability, maintainability, and retrieval efficiency while ensuring secure and privacy-focused document interaction.

V. IMPLEMENTATION

The implementation of DOC GPT focuses on integrating document processing, semantic retrieval, and conversational AI into a single intelligent system. The application is developed using a full-stack architecture consisting of a Next.js frontend, FastAPI backend, FAISS vector database, and locally hosted Large Language Models through Ollama. The system is designed to support secure document interaction, semantic search, and context-aware response generation.

A. Frontend Implementation

The frontend of DOC GPT is developed using Next.js, React.js, TypeScript, and Tailwind CSS. The frontend interface provides a responsive and user-friendly environment for interacting with the system. It allows users to upload PDF documents, ask questions, view conversation history, and interact with the chatbot in real time.

Authentication features are implemented to support secure user login and signup operations. The frontend communicates with backend APIs using REST-based HTTP requests. User sessions and authentication tokens are managed to ensure secure access to application resources.

The chat interface is designed to support conversational interaction with uploaded documents. Users can submit queries through text input as well as voice input functionality. Voice input support is implemented using browser-based speech recognition APIs.

B. Backend Implementation

The backend of the system is implemented using FastAPI. It manages document uploads, authentication, semantic retrieval operations, embedding generation, and communication with the language model inference engine.

The backend exposes multiple REST API endpoints for:

1) User authentication
2) PDF upload handling
3) Conversation management
4) Semantic search operations
5) AI response generation

The backend processes uploaded documents asynchronously to improve performance and responsiveness. Communication between frontend and backend APIs follows JSON-based request and response handling.

C. Document Processing Module

The document processing module handles PDF extraction and preprocessing operations. Text-based PDF documents are processed using pdfplumber for direct text extraction. Scanned or image-based documents are converted into images using pdf2image and processed using Tesseract OCR. The extracted document content is cleaned and divided into smaller chunks before embedding generation. Chunking improves semantic retrieval efficiency and enables the system to process large documents effectively.

D. Semantic Embedding and Retrieval

The system uses Sentence Transformer models for semantic embedding generation. The BAAI/bge-base-en-v1.5 model converts document chunks and user queries into dense vector embeddings. Generated embeddings are stored inside the FAISS vector database. During query processing, semantic similarity search operations retrieve document chunks that are contextually related to the user query. To improve retrieval quality, the system applies reranking techniques after similarity search. The reranking mechanism prioritizes the most relevant contextual information before passing it to the language model.

E. AI Response Generation

The response generation module uses locally hosted Large Language Models through Ollama. Models such as Llama 3.1 generate conversational responses using the retrieved document context and user query. The Retrieval-Augmented Generation workflow helps reduce hallucination issues by grounding responses using document-based contextual information. Since the models operate locally, the system maintains privacy and reduces dependency on external cloud services.

F. Database Implementation

SQLite is used as the primary database for storing user information, authentication details, document metadata, and conversation history. The database layer supports persistent storage and user session management within the application. Conversation records are maintained to allow users to revisit previous interactions and continue document-based discussions.

G. Voice Input Integration

The system also includes voice input functionality for improved user interaction. Browser-based speech recognition APIs are used to convert speech into text. The recognized text is automatically inserted into the chat input field and processed like standard user queries. Error handling and browser compatibility mechanisms are implemented to improve stability across different browsers.

H. System Integration and Workflow

All system components are integrated to create a complete end-to-end document chatbot workflow. The implementation flow can be summarized as follows:

1) User uploads a document through the frontend interface.
2) Backend services process the uploaded document.
3) Text extraction and OCR operations are performed.
4) Extracted text is divided into chunks.
5) Semantic embeddings are generated and stored in the FAISS database.
6) User submits a query through the chat interface.
7) Relevant document chunks are retrieved using semantic similarity search.
8) Retrieved results are reranked for improved relevance.
9) Contextual information is passed to the Large Language Model.
10) The generated response is returned to the frontend and displayed to the user.

The implemented system successfully combines semantic retrieval, OCR processing, vector similarity search, and local AI inference into a unified intelligent document interaction platform.

VI. EXPERIMENTAL RESULTS

The DOC GPT system was evaluated using various types of PDF documents, including academic reports, scanned documents, technical articles, and text-based PDFs. The evaluation focused on semantic retrieval accuracy, response relevance, OCR effectiveness, system responsiveness, and overall user interaction performance.

A. Retrieval Accuracy

Several user queries related to uploaded documents were used to test the semantic retrieval pipeline. In most natural language searches, the FAISS vector database successfully retrieved contextually relevant document chunks. Compared to traditional keyword-based retrieval methods, the semantic search approach provided more accurate and meaningful results.

Sentence Transformer embeddings improved the system's ability to understand contextual similarity between user questions and document content. The application of the reranking mechanism further improved retrieval efficiency, especially for lengthy or semantically complex queries.

B. OCR Performance

The OCR module was evaluated using scanned and image-based PDF documents. Tesseract OCR successfully extracted readable text from most scanned files with acceptable accuracy. The extracted text was then processed for semantic embedding generation and retrieval operations.

OCR accuracy depended on factors such as image quality, font clarity, and document resolution. Clean scanned documents produced better extraction results compared to low-quality or distorted images. Although minor extraction errors were observed in some cases, the OCR module still performed effectively for general document understanding tasks.

C. Response Generation Performance

The response generation module was tested using locally hosted Large Language Models through Ollama. Models such as Llama 3.1 used retrieved document information to generate context-aware responses. The Retrieval-Augmented Generation process reduced irrelevant outputs by grounding responses using contextual document data.

The average response generation time ranged between 2 to 5 seconds depending on document size, query complexity, and hardware configuration. During repeated testing sessions, the system maintained stable conversational interaction.

D. System Responsiveness

Frontend and backend communication were evaluated during document upload, semantic retrieval, and conversational interaction. The Next.js frontend enabled responsive real-time interaction, while FastAPI efficiently handled API requests.

The system successfully managed:

1) PDF uploads
2) Multi-query interactions

3) Semantic retrieval operations
4) Voice input processing
5) Conversation history management

Testing showed that the integration between the frontend, backend, FAISS retrieval system, and Ollama inference engine operated consistently during user interaction.

E. Privacy and Local Inference Evaluation

One of the major advantages observed during testing was privacy preservation through local AI inference. Since the language models operated locally using Ollama, uploaded documents and user conversations were not transmitted to external cloud services.

The local deployment approach reduced dependency on internet connectivity and improved control over sensitive document information. Offline functionality was also maintained for document interaction and response generation.

F. Overall Performance Analysis

The experimental evaluation demonstrated that the proposed DOC GPT system effectively integrates semantic retrieval, OCR processing, vector similarity search, and local Large Language Model inference into a single intelligent platform.

The system demonstrated:

1) Improved contextual understanding
2) Faster semantic retrieval
3) Accurate document-based responses
4) Stable conversational interaction
5) Effective OCR-based text extraction
6) Privacy-focused local deployment

The integration of Retrieval-Augmented Generation significantly improved response quality compared to traditional keyword-based document retrieval systems.

VII. ADVANTAGES OF THE PROPOSED SYSTEM

The proposed DOC GPT system offers several advantages over traditional document retrieval and conversational AI systems. The integration of semantic retrieval, OCR processing, vector similarity search, and locally hosted Large Language Models allows the system to provide efficient and context-aware document interaction.

1) Improved Semantic Understanding: DOC GPT uses semantic embeddings to understand the contextual meaning of user queries instead of relying only on traditional keyword-based search methods. This allows the system to retrieve information based on meaning rather than exact keyword matching, resulting in more accurate and relevant responses.
2) Context-Aware Response Generation: The Retrieval-Augmented Generation (RAG) approach enables the system to generate responses using contextual information retrieved from uploaded documents. This reduces irrelevant or misleading outputs during conversational interaction and improves overall response quality.
3) Privacy-Focused Local Deployment: The system uses locally hosted Large Language Models through Ollama, allowing document processing and response generation to occur without sending user data to external cloud services. This improves privacy and provides better control over sensitive information.
4) Support for Scanned Documents: The integration of OCR technologies such as Tesseract OCR enables the system to process scanned and image-based PDF documents. This allows users to interact with documents that do not contain selectable digital text.
5) Efficient Semantic Retrieval: The FAISS vector database supports fast indexing and similarity search operations for high-dimensional embeddings. This improves retrieval speed and allows the system to efficiently process large document collections.
6) Real-Time Conversational Interaction: DOC GPT provides a real-time conversational interface where users can ask questions and receive responses instantly. The conversational approach improves user interaction and makes document navigation more convenient.
7) Voice Input Support: The system includes voice input functionality using browser-based speech recognition APIs. Users can interact with the chatbot using voice commands in addition to standard text input.
8) Modular and Scalable Architecture: The modular architecture of the system improves maintainability and scalability. Individual modules such as document processing, embedding generation, retrieval operations, and AI inference can be upgraded or modified independently.

9) Reduced Dependency on External Services: Since the system operates locally, it reduces dependency on internet connectivity and third-party cloud APIs. This enables offline functionality and improves reliability during document interaction.

10) Enhanced User Experience: The integration of semantic retrieval, conversational AI, OCR support, and responsive frontend technologies provides a user-friendly environment for interacting with uploaded documents efficiently and accurately.

VIII. CONCLUSION

This paper presented DOC GPT, a Retrieval-Augmented Generation based intelligent document chatbot designed for context-aware interaction with uploaded documents. The system combines semantic retrieval, OCR processing, vector similarity search, and locally hosted Large Language Models to provide efficient and accurate document-based conversational responses.

The proposed system successfully addresses several limitations of traditional keyword-based document retrieval systems by using semantic embeddings and contextual retrieval techniques. The integration of FAISS vector indexing improved retrieval efficiency, while the use of reranking mechanisms enhanced the relevance of retrieved document information. OCR integration also enabled the system to process scanned and image-based PDF documents effectively.

The implementation of locally hosted Large Language Models through Ollama improved privacy and reduced dependency on external cloud services. This local deployment approach allowed secure document interaction while maintaining offline functionality and better control over sensitive information. Experimental evaluation demonstrated that the system was capable of generating context-aware responses with improved semantic retrieval accuracy and stable conversational interaction. The modular architecture of DOC GPT also provides flexibility for future improvements and scalability.

Overall, the proposed DOC GPT system demonstrates how Retrieval-Augmented Generation can be combined with semantic search and local AI inference to create an efficient and privacy-focused intelligent document assistant.

Future enhancements may include support for multilingual document processing, multi-document reasoning, cloud synchronization, advanced reranking models, and fine-tuned domain-specific language models.

IX. FUTURE SCOPE

The proposed DOC GPT system provides an effective platform for intelligent document interaction using Retrieval-Augmented Generation and locally hosted Large Language Models. Although the current implementation demonstrates efficient semantic retrieval and context-aware response generation, several improvements can be introduced in future versions of the system.

One possible enhancement is multilingual document support. The current system mainly focuses on English-language documents. Future development can include multilingual embedding models and OCR support to process documents written in different languages. Another area for improvement is multi-document reasoning. At present, the system retrieves information from uploaded documents individually. Future implementations can support cross-document retrieval and reasoning, allowing the chatbot to combine information from multiple sources while generating responses.

The OCR module can also be improved using advanced OCR frameworks with better accuracy for low-quality scanned documents and handwritten text. Improved preprocessing techniques may further increase extraction quality for image-based PDFs.

Future versions of the system may also include cloud synchronization and distributed storage support. This would allow users to securely access documents and conversation history across multiple devices while maintaining privacy controls.

The integration of fine-tuned domain-specific language models is another possible enhancement. Specialized models trained on legal, medical, academic, or enterprise datasets could improve response quality for domain-specific applications.

Performance optimization is also an important future direction. GPU acceleration, optimized vector indexing techniques, and lightweight embedding models may improve response speed and retrieval efficiency for large-scale document collections.

Additional features such as mobile application support, collaborative document interaction, advanced user access control, and real-time streaming responses can further improve usability and system scalability.

Overall, the modular architecture of DOC GPT provides flexibility for future expansion and allows the system to adapt to evolving developments in conversational AI, semantic retrieval, and intelligent document processing technologies.

X. ACKNOWLEDGEMENT

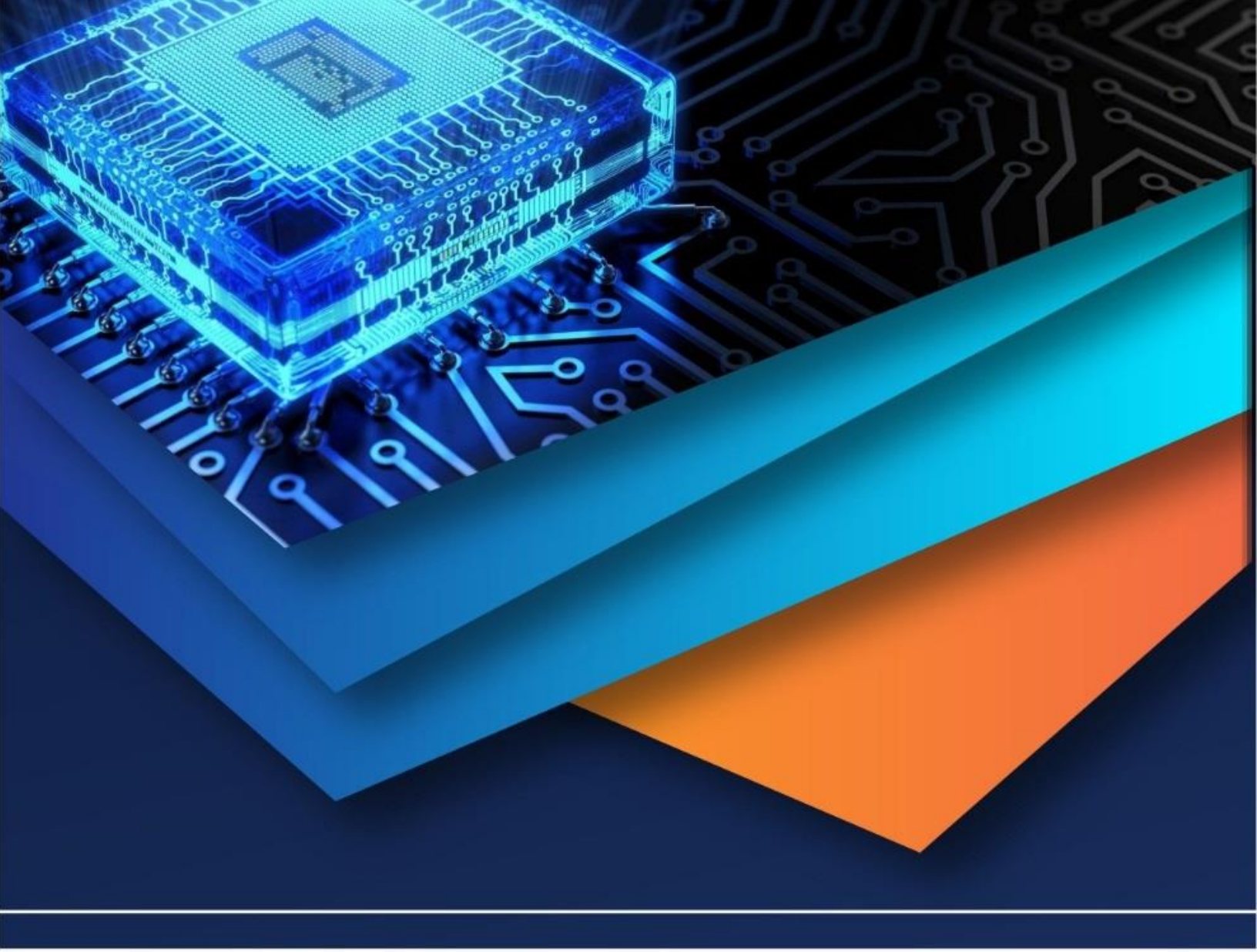
The authors would like to express their sincere gratitude to the Department of Computer Applications, Seshadripuram Academy of Business Studies, Bengaluru University, for providing guidance, resources, and support during the development of this project.

The authors also thank the project guide, faculty members, and peers who contributed valuable suggestions and technical assistance throughout the implementation and documentation process.

Special acknowledgment is given to the open-source developer community and the contributors of technologies such as FastAPI, Next.js, FAISS, Ollama, Hugging Face Transformers, and Tesseract OCR, which played an important role in the successful development of the DOC GPT system.

REFERENCES

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), 2020.
[2] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with FAISS," IEEE Transactions on Big Data, 2019.
[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, 2019.
[4] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), 2017.
[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proceedings of EMNLP, 2019.
[6] "FastAPI Documentation." Available: https://fastapi.tiangolo.com/
[7] "Ollama Documentation." Available: https://ollama.com/
[8] "Hugging Face Transformers Documentation." Available: https://huggingface.co/docs/transformers/index
[9] "FAISS Documentation." Available: https://faiss.ai/
[10] "Sentence Transformers Documentation." Available: https://www.sbert.net/
[11] "Tesseract OCR Documentation." Available: https://tesseract-ocr.github.io/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY