🧠 Agentic Research Assistant (RAG) – Project Report

1. Introduction

The Agentic Research Assistant is a document-based question-answering system powered by Retrieval-Augmented Generation (RAG). This tool allows users to upload a PDF, input an OpenAI API key, and ask complex research questions. The system returns accurate, LLM-backed answers along with a transparent reasoning log. The primary goal is to simplify and accelerate document comprehension for researchers, students, and professionals.

2. Objectives

- Implement a functional RAG pipeline using LangChain, FAISS, and OpenAI GPT.
- Enable users to upload PDF documents and interact via a user-friendly interface.
- Provide transparent reasoning steps for each question.
- Support both open-ended and predefined questions with contextual answers.

3. Technology Stack

Component	Technology Used
Frontend	Gradio (v4, dark theme)
Backend	Python
LLM Provider	OpenAI (ChatGPT API)
RAG Framework	LangChain
Vector Store	FAISS
PDF Parsing	PyPDF2 (via LangChain)

4. System Architecture

1. PDF Upload & Processing

- PDF is uploaded via Gradio interface.
- Parsed using PyPDFLoader.

2. Vectorization & Storage

- Text chunks are embedded via OpenAIEmbeddings.
- Stored in a FAISS vector database.

3. Question Answering

- User inputs a question.
- System retrieves relevant chunks using similarity search.
- ChatGPT generates a response via LangChain's RetrievalQA.

4. Output Display

- Final answer and reasoning steps shown with timestamps.

5. Key Features

- Upload any research paper or PDF file.
- Ask natural language questions.
- View intermediate reasoning steps for full transparency.
- Preloaded buttons for common research topics (AI, Quantum, Blockchain, etc.).
- Modern UI using Gradio dark theme.

6. How to Use

1. Run the app using:

- git clone https://github.com/NavinduKojitha/Agentic-Research-Assistant-RAG-.git
- cd Agentic-Research-Assistant-RAG-
- pip install -r requirements.txt
- python app.py

2. Open the local Gradio interface.

3. Upload a PDF and provide your OpenAI API Key.

4. Ask questions or choose a topic button.

5. Review the answer and see the step-by-step reasoning log.

7. Usage Instructions

1. Upload PDF: Choose a research PDF file using the upload box.
2. Enter API Key: Input your OpenAI API key securely in the textbox.

navindukb@gmail.com +94768854960

3. Click 'Load Agent': This initializes the RAG agent with your document.
4. Ask a Question: Type a research-related question or select from the predefined topic buttons.
5. View the Response: The assistant returns a detailed answer and logs each reasoning step with timestamps.

8. Example Use Case

Upload a research paper on Artificial Intelligence.
Ask: "What are the limitations of current AI models discussed in this paper?"
Receive a structured and reasoned response based on the content.

9. Results and Evaluation

The system accurately retrieves context and generates relevant, concise answers for a variety of uploaded documents. The transparent reasoning panel helps users understand the internal steps, making it educational and explainable. The use of OpenAI's LLM ensures high-quality language generation, while FAISS ensures fast vector search.

10. Challenges Faced

- Handling API key security in client facing UI.
- Managing large PDF chunking without exceeding token limits.
- Ensuring FAISS indexing and document retrieval is accurate and performant.

11. Future Improvements

- Add support for multi-file and folder uploads.
- Integrate .env for secure API key management.
- Use streaming responses for faster user feedback.
- Option to fine-tune on custom domain data or switch LLM providers.

12. Conclusion

This project showcases the power of RAG-based systems in real-world document analysis. The combination of LangChain, OpenAI, and FAISS delivers an intelligent assistant that can parse, understand, and respond to complex queries making it a valuable tool for academic and professional use.