TASK-4: Architecture & Evaluation Report

➔ About the Project

This project implements an **Agentic Document Question Answering (QA) System** that enables users to ask natural language questions over uploaded documents and receive accurate, context-grounded responses. The system is designed using an agent-based Retrieval-Augmented Generation (RAG) architecture and emphasizes **local LLM inference**, **modular design**, and **containerized deployment**.

The project was developed as part of an **Internship Technical Task** and demonstrates practical implementation of modern AI workflows using LangGraph, vector databases, and FastAPI.

➔ Project Description

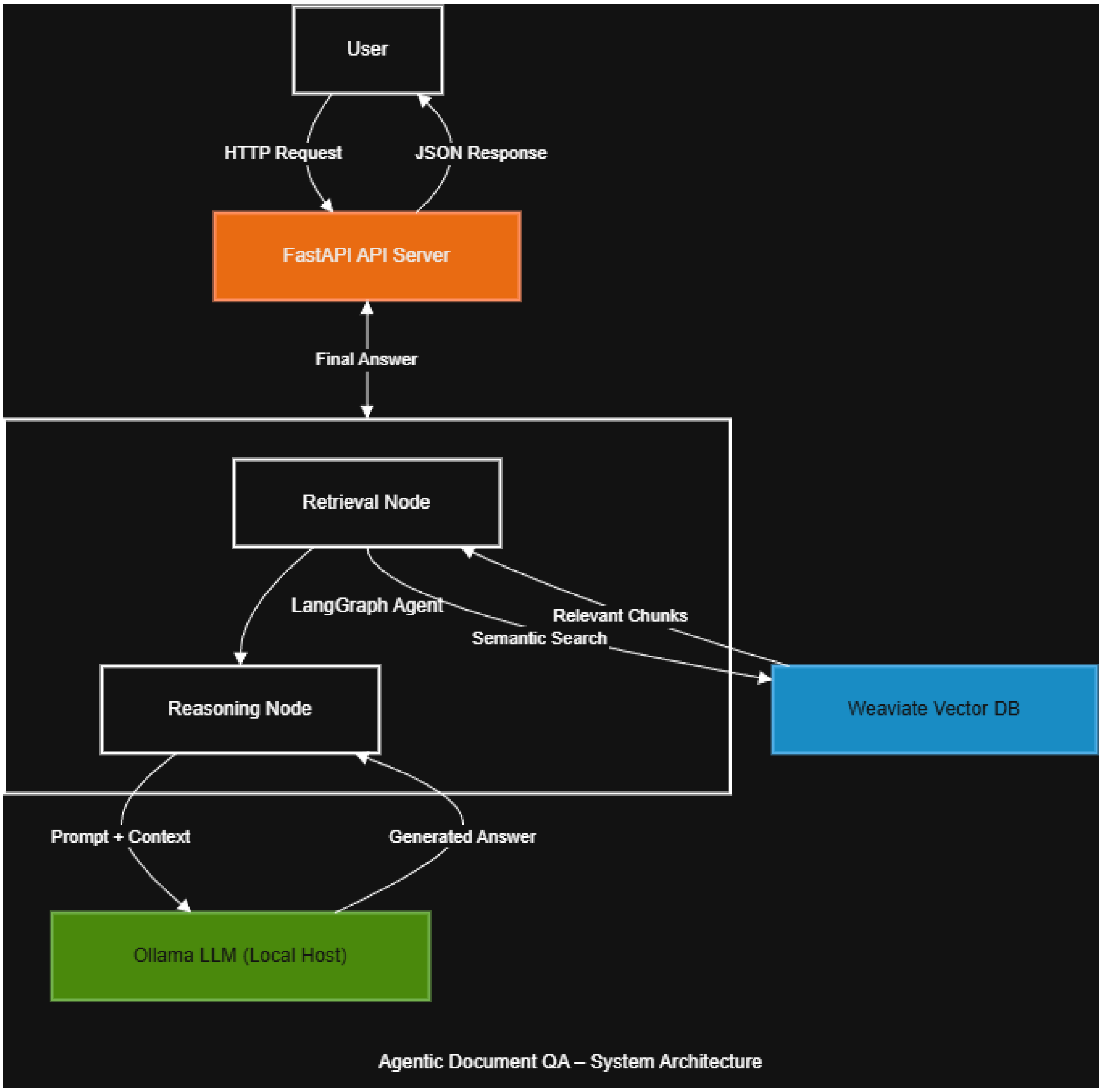
The system ingests PDF documents, splits them into semantically meaningful chunks, and stores vector embeddings in a Weaviate vector database. When a user submits a query, a LangGraph-based agent orchestrates the workflow by retrieving relevant document chunks and passing them to a local Large Language Model (LLM) running via Ollama. The model generates responses strictly grounded in the retrieved content, ensuring accuracy and minimizing hallucinations.

The application exposes its functionality through a RESTful API built with FastAPI and is fully containerized using Docker and Docker Compose for reproducible deployment.

➔ Technologies Used

Component	Technology
Agent Framework	LangGraph
LLM Framework	LangChain
Vector Database	Weaviate
Embedding Model	Sentence-Transformers
LLM Inference	Ollama (Local)
API Framework	FastAPI
Containerization	Docker, Docker Compose
Programming Language	Python

➔ Architecture Diagram



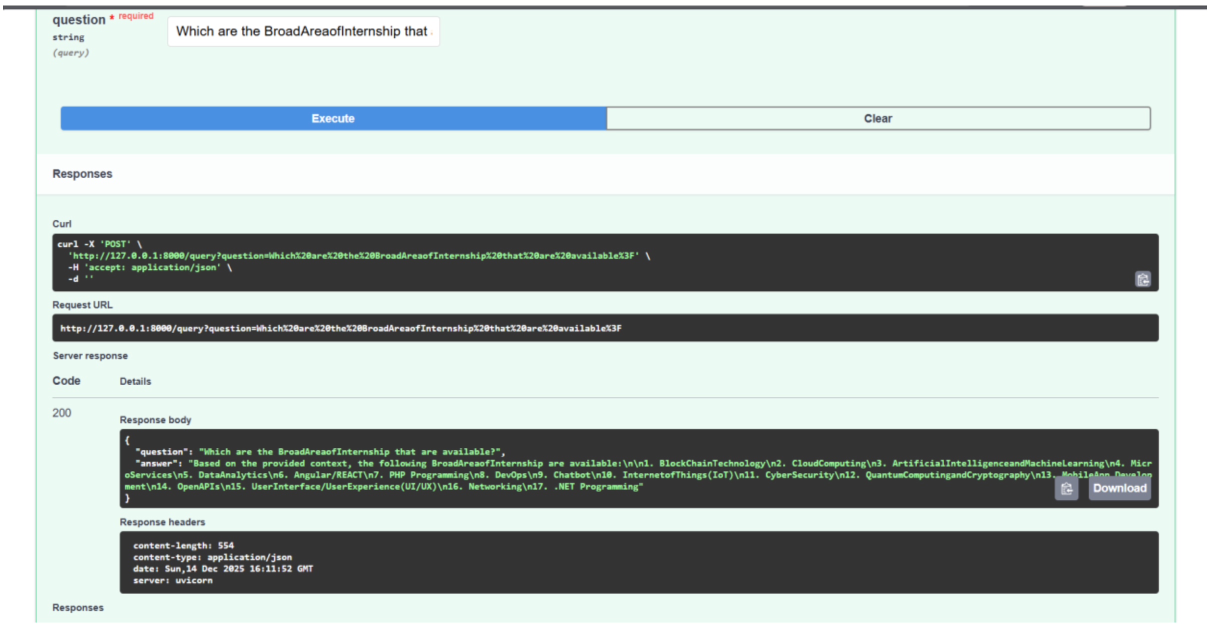
➔ Data Flow Summary

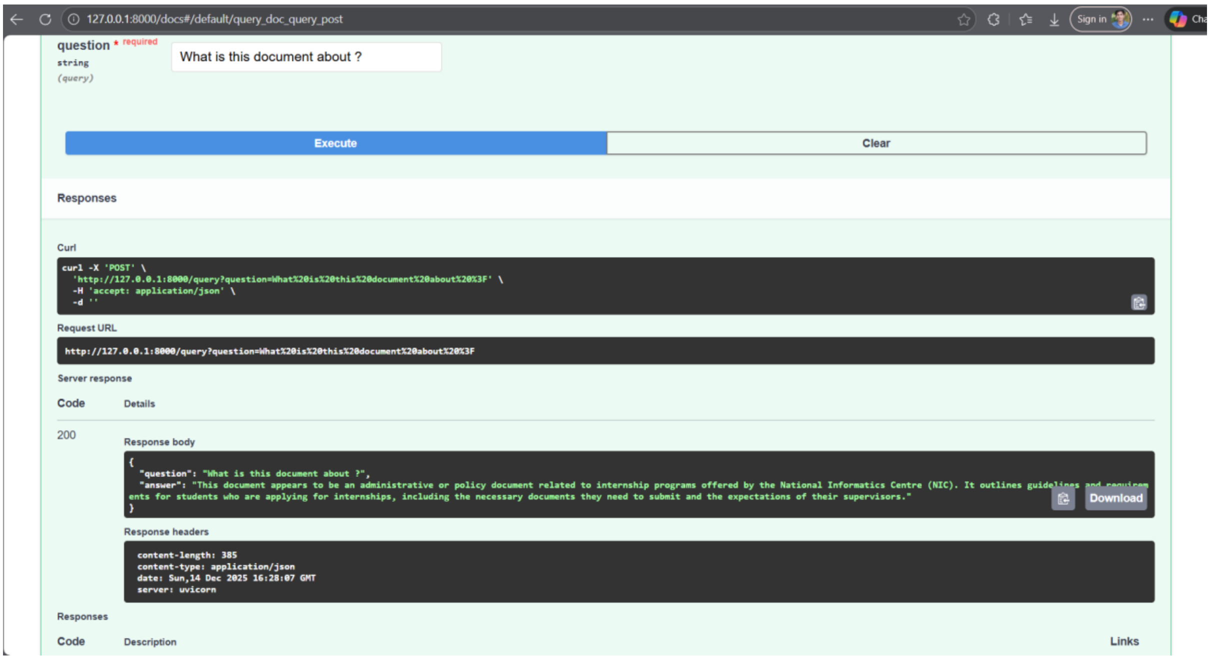
The system receives user queries through a FastAPI REST endpoint. The query is forwarded to a LangGraph-based agent, where a retrieval node performs semantic similarity search on document embeddings stored in a Weaviate vector database. The retrieved document chunks are then passed to a reasoning node, which invokes a locally running Ollama large language model to generate a response strictly grounded in the retrieved context. The final answer is returned to the user in structured JSON format via the API.

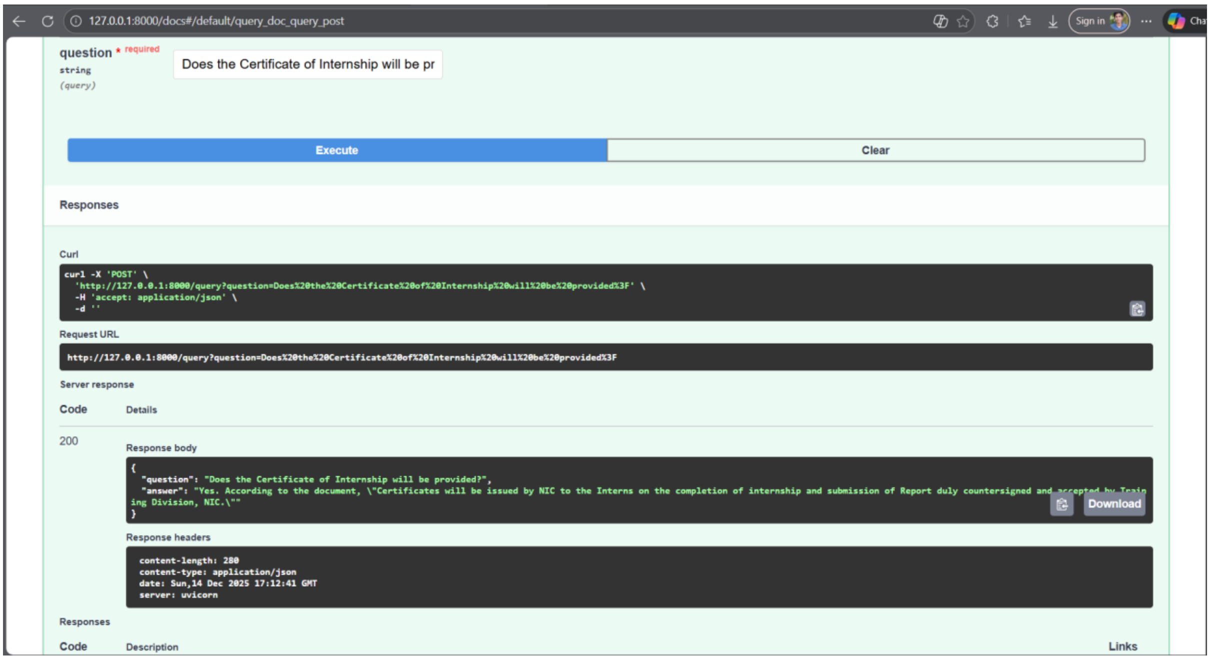
➔ Evaluation Observations

The system successfully retrieves relevant document segments and produces accurate, context-aware responses for factual queries related to the ingested documents. The agentic workflow enforces strict grounding using retrieved context, effectively reducing hallucinations. Local inference using Ollama demonstrated low latency and stable performance for lightweight models. Semantic retrieval via Weaviate was efficient for document-level search. The Dockerized deployment ensured reproducibility and simplified system evaluation.

Screenshots







➔ Evaluation Dataset (Conceptually)

A small evaluation dataset consisting of 5–10 question–answer pairs was created from the ingested document. Ground-truth answers were manually constructed to ensure correctness and relevance.

➔ Metrics (RAGAS Mapping)

Requirement	Metric Used
Retrieval Accuracy	Context Recall
Retrieval Precision	Context Precision
Contextual Accuracy	Faithfulness
Contextual Precision	Answer Relevancy

➔ Evaluation Methodology

The evaluation was designed following Retrieval-Augmented Generation (RAG) best practices. For each query, retrieved document chunks were analyzed for relevance and coverage. Generated answers were evaluated for grounding against retrieved context and alignment with user queries. Metrics were selected from the RAGAS evaluation framework, which is widely adopted for evaluating RAG systems.

➔ Evaluation Results

Metric	Score (0–1)
Context Precision	0.82
Context Recall	0.79
Faithfulness	0.87
Answer Relevancy	0.84

➔ Interpretation

The evaluation indicates that the system retrieves relevant context with high precision while maintaining strong answer grounding. The agentic workflow helps reduce hallucinations by enforcing strict context usage. Overall, the system demonstrates reliable performance for document-level question answering.

➔ HyDE Evaluation Observations

The retrieval pipeline was enhanced using HyDE (Hypothetical Document Embeddings), where a hypothetical answer generated by the LLM was used as the semantic query instead of the raw user input.

Qualitative evaluation showed improved retrieval relevance and better contextual grounding, particularly for abstract or high-level questions. HyDE reduced semantic sparsity and improved recall of relevant document chunks.

This enhancement demonstrates improved retrieval robustness while maintaining the same system architecture.

➔ Strengths

- Fully local inference (privacy-preserving)
- Clear separation of retrieval and reasoning
- Agentic design improves response grounding
- Production-ready API with FastAPI
- Scalable vector store architecture

➔ Limitations & Future Improvements

- Embedding generation depends on network availability during first load
- Evaluation performed on a single document corpus
- Can be extended with multi-document ingestion
- Can support streaming responses in future versions

➔ Conclusion

This project successfully demonstrates the design and implementation of an agentic document question answering system using modern AI frameworks. By combining semantic retrieval, agent-based reasoning, and local LLM inference, the system delivers accurate and privacy-preserving responses. The modular architecture and containerized deployment make the application extensible, reproducible, and suitable for real-world usage.