



Agentic RAG Chatbot for Multi- Format Document QA

A Submission for the SLRI Solutions Coding Challenge

Kowshik Padala

July 24, 2025



Architecture: A Coordinated Multi-Agent System

This project is built on a robust, agent-based architecture designed for clarity and scalability. A central CoordinatorAgent orchestrates the workflow, directing tasks to specialized agents.



CoordinatorAgent

The "brain" of the system. It receives user requests and manages the end-to-end RAG pipeline by activating the appropriate agent for each sub-task.



IngestionAgent

Responsible for parsing and extracting text from a wide variety of document formats, including PDF, DOCX, PPTX, CSV, and TXT.



RetrievalAgent

Handles the core "Retrieval" logic. It chunks the extracted text, generates vector embeddings using a Hugging Face model, and manages the FAISS vector store for semantic search.

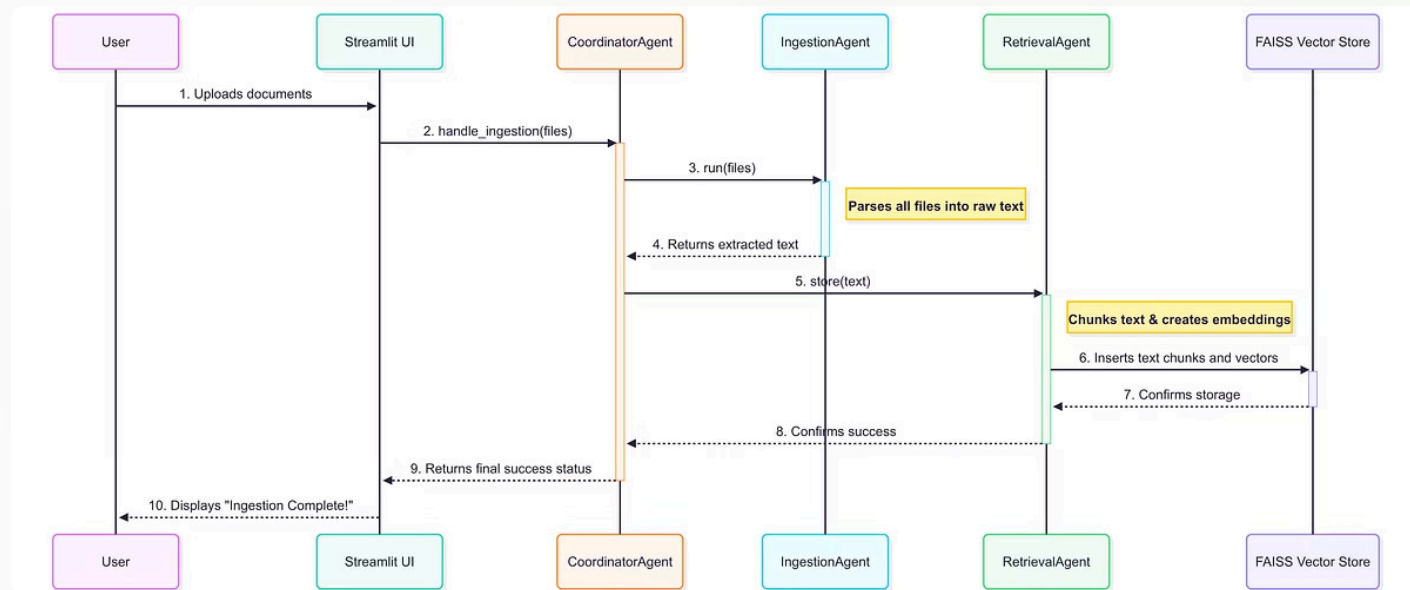


LLMResponseAgent

The final link in the chain. It constructs a precise, context-aware prompt and interfaces with the Groq LLM to generate the final answer.

Model Context Protocol (MCP): All communication between agents is standardized using a structured, JSON-based protocol. This ensures that every message is clear, traceable, and contains a well-defined payload, making the system easy to debug and extend.

System Flow: Ingestion and Querying



Tech Stack & User Interface

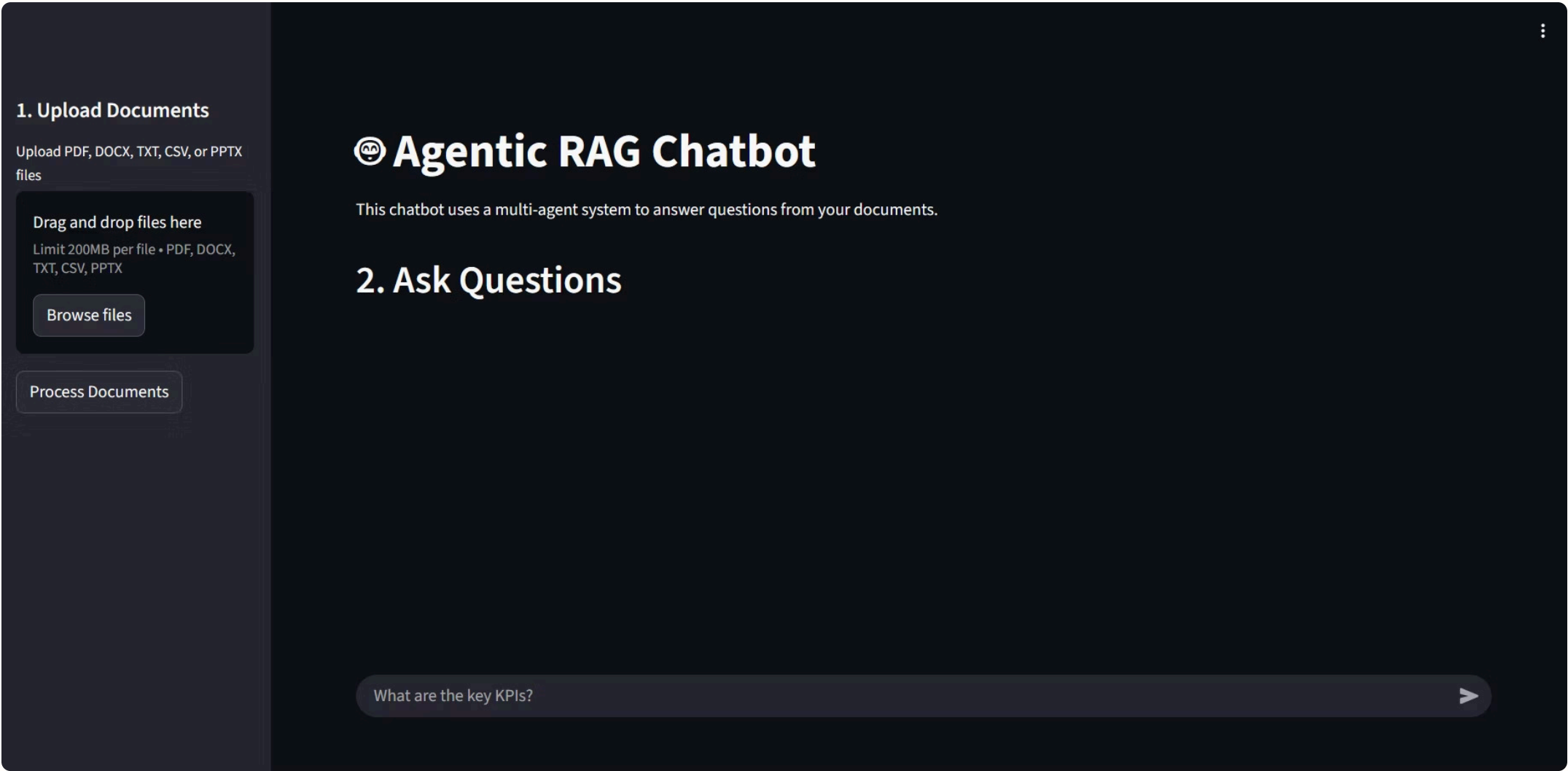
The application leverages a modern, efficient stack to deliver a seamless user experience.

Backend & Orchestration:

- **Language:** Python 3.12
- **Core Framework:** LangChain

Retrieval-Augmented Generation:

- **Embeddings:** all-MiniLM-L6-v2 (via Sentence Transformers)
- **Vector Store:** FAISS (Facebook AI Similarity Search)
- **LLM:** Llama 3 (via Groq API for high-speed inference)



Challenges Faced & Future Improvements

Challenges Encountered:

- **Dependency Management**

The langchain library recently modularized into sub-packages (langchain-community, etc.). Navigating this change required careful dependency management to resolve ModuleNotFoundError errors and ensure a stable environment.

- **Prompt Engineering**

Crafting a robust prompt for the LLMResponseAgent was an iterative process. It was critical to instruct the LLM to answer *only* from the provided context, thereby preventing hallucinations and ensuring the verifiability of answers.

Potential Future Scope:

- **Asynchronous Ingestion**


Implement asyncio to process large documents in the background, preventing the UI from blocking and creating a smoother user experience.

- **Persistent Vector Store**

Replace the in-memory FAISS store with a persistent database like ChromaDB to retain the knowledge base across application sessions.

- **Formal Unit Testing**

Integrate pytest to create unit tests for each agent, enhancing the long-term robustness and maintainability of the codebase.



Thank You

I welcome any questions you may have.

GitHub Repository: <https://github.com/Kowshik4593/Agentic-RAG-Chatbot>

Email: padalakowshik@gmail.com

