Agentic RAG Chatbot with MCP

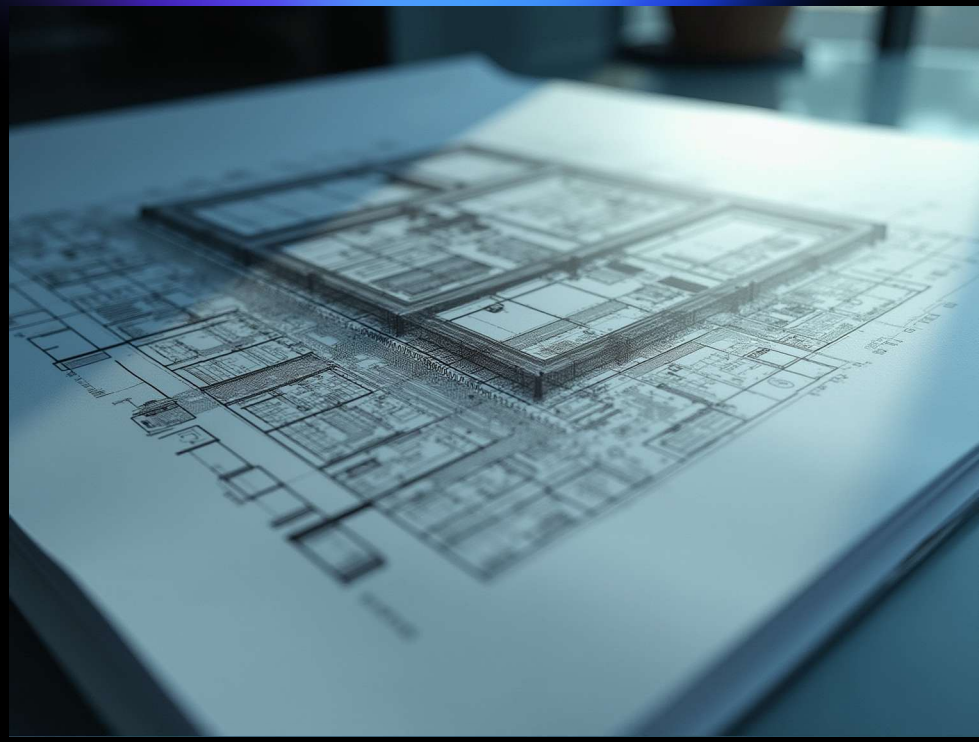
Muhammed Saneeb ek

Introduction

This presentation outlines the architecture and technology stack of the Agentic RAG Chatbot integrated with the MCP, highlighting its core components and how they interact.

01

Architecture



Streamlit UI

The Streamlit UI serves as the front-end interface for users to interact with the chatbot. It allows users to upload documents and send queries, providing a user-friendly experience that facilitates seamless communication between the UI and backend services.

FastAPI Backend

The FastAPI backend processes requests from the Streamlit UI. It coordinates the functions of various agents, manages the ingestion of documents, and interfaces with the ChromaDB for efficient data retrieval and storage.

ChromaDB Flow

The ChromaDB acts as a vector store within the architecture, facilitating efficient data processing. The flow begins when a user uploads documents through the Streamlit UI, which are then processed by the backend. As the IngestionAgent processes the documents, it encapsulates meaningful chunks and stores them in ChromaDB. During query requests, the ConversationalAgent retrieves relevant information from ChromaDB and responds to user inquiries.

02

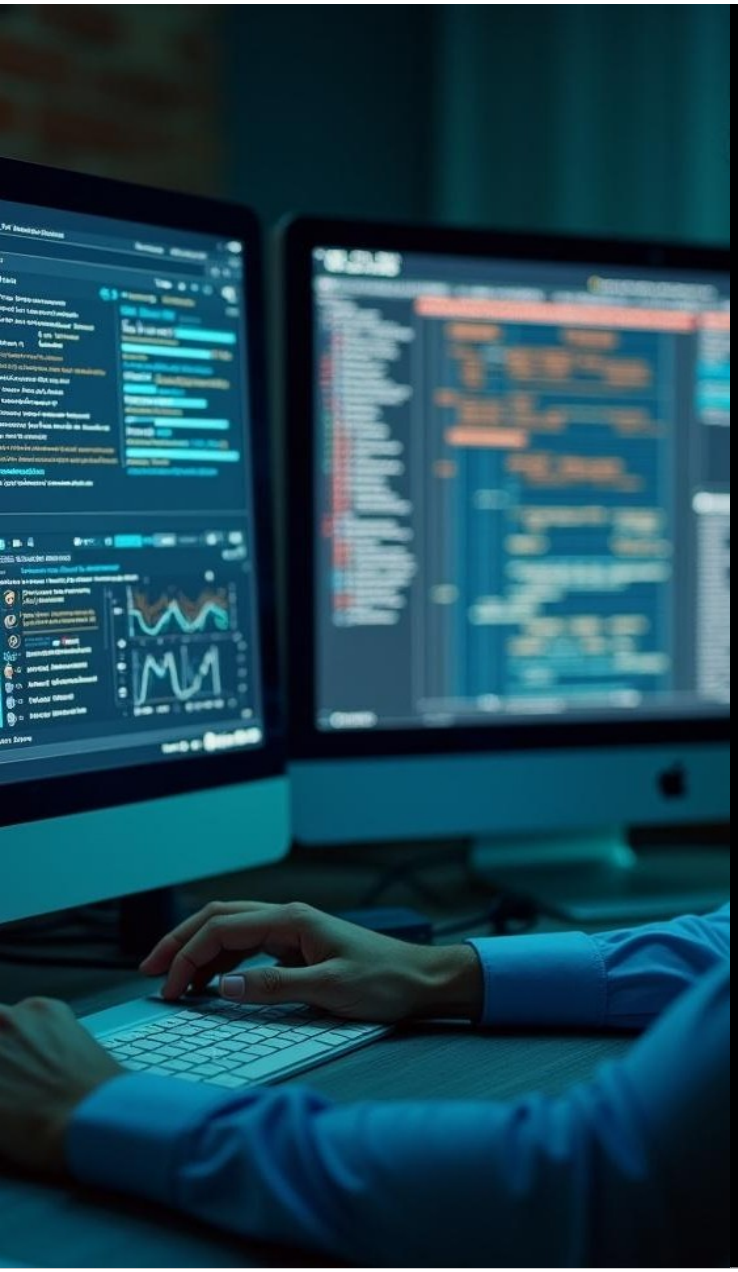
Challenges

Asynchronous Processing Issues

Asynchronous file processing can sometimes lead to errors such as 'read of closed file'. This challenge arises when the file is not handled correctly in threads. To mitigate this issue, it's essential to read the file contents into memory within the main API thread, ensuring that raw data is correctly passed to background processing tasks.

Bot Hallucination Solutions

One of the significant challenges faced by the chatbot is its tendency to hallucinate or provide responses not grounded in the provided documents. This issue has been resolved by implementing a proper chunking strategy in the IngestionAgent, ensuring that the vector store retains meaningful and specific context rather than generic embeddings.



Source Context Display

Displaying source context while using a streaming UI can be complex. A two-step process has been implemented to enhance user experience: the system first streams the initial answer for immediate feedback, followed by a second API call to a dedicated '/context' endpoint to fetch and display the relevant source material.

Conclusions

The architecture of the Agentic RAG Chatbot showcases an effective integration of the Streamlit UI, FastAPI backend, and ChromaDB. Addressing challenges related to asynchronous processing, bot hallucinations, and source context display has significantly improved the overall functionality and user experience.

Thank you!