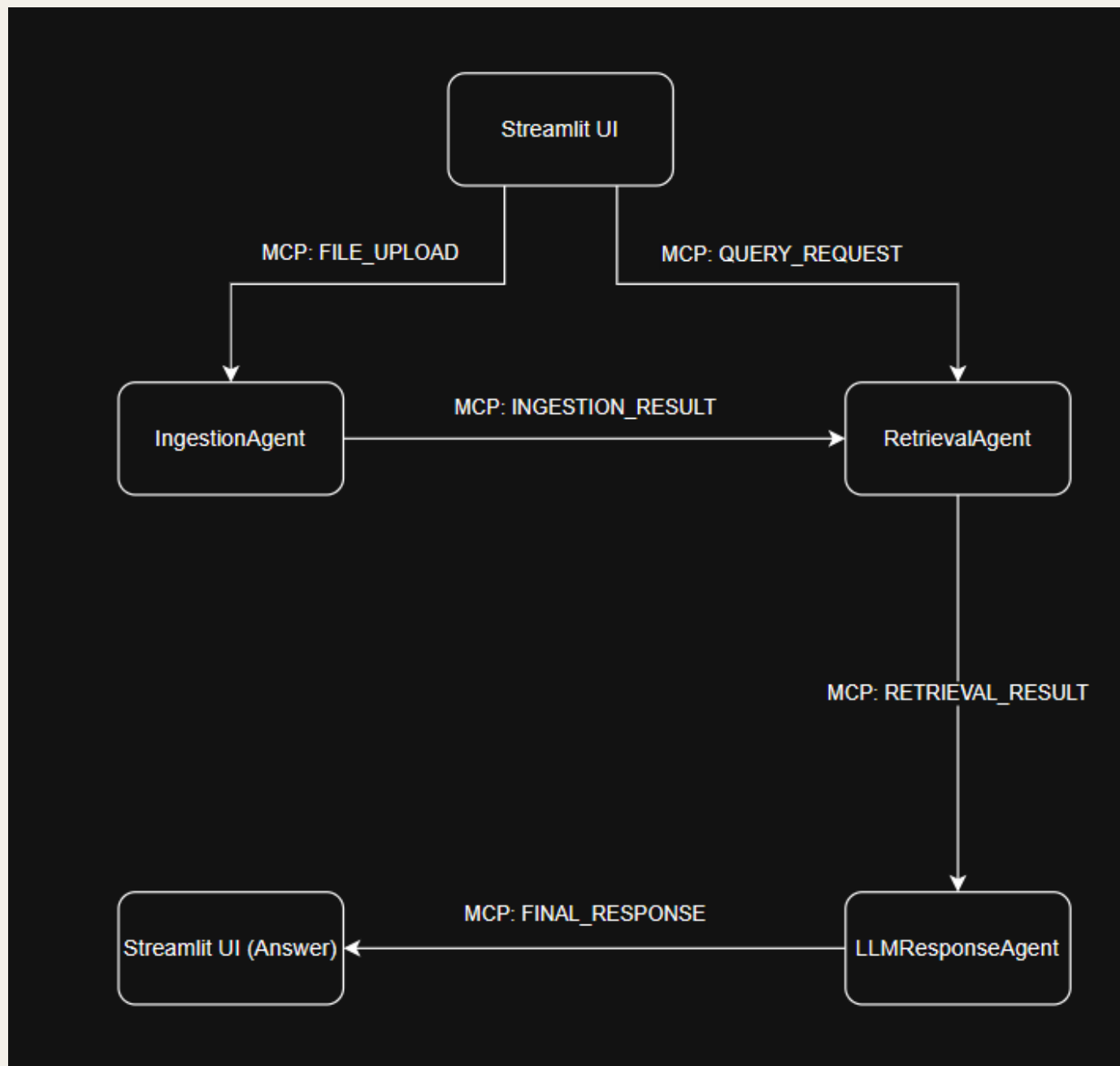


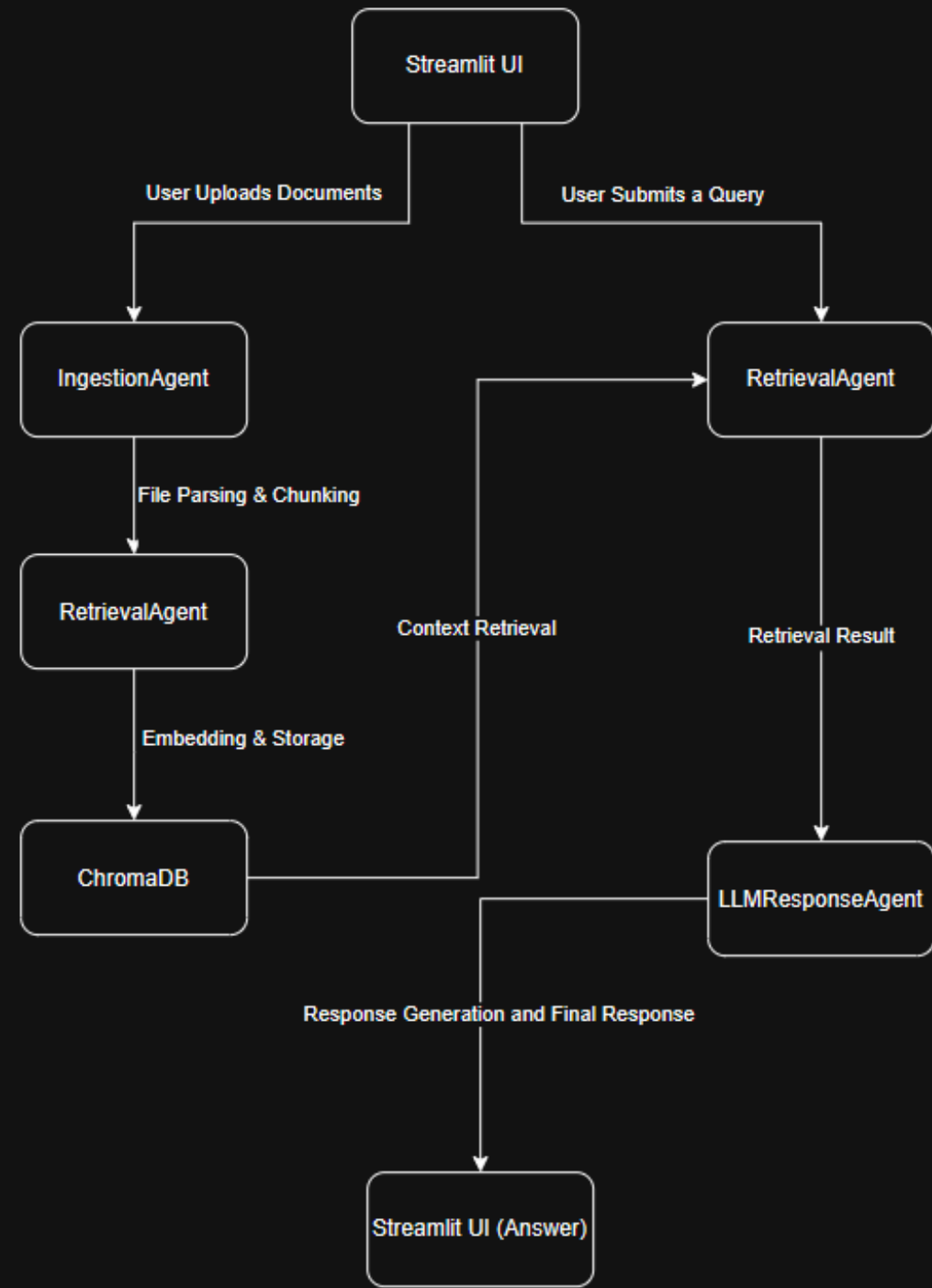
Agentic RAG Chatbot for Multi- Format Document QA using Model Context Protocol (MCP)

Siddartha Kommu

Agent-based architecture with MCP integration



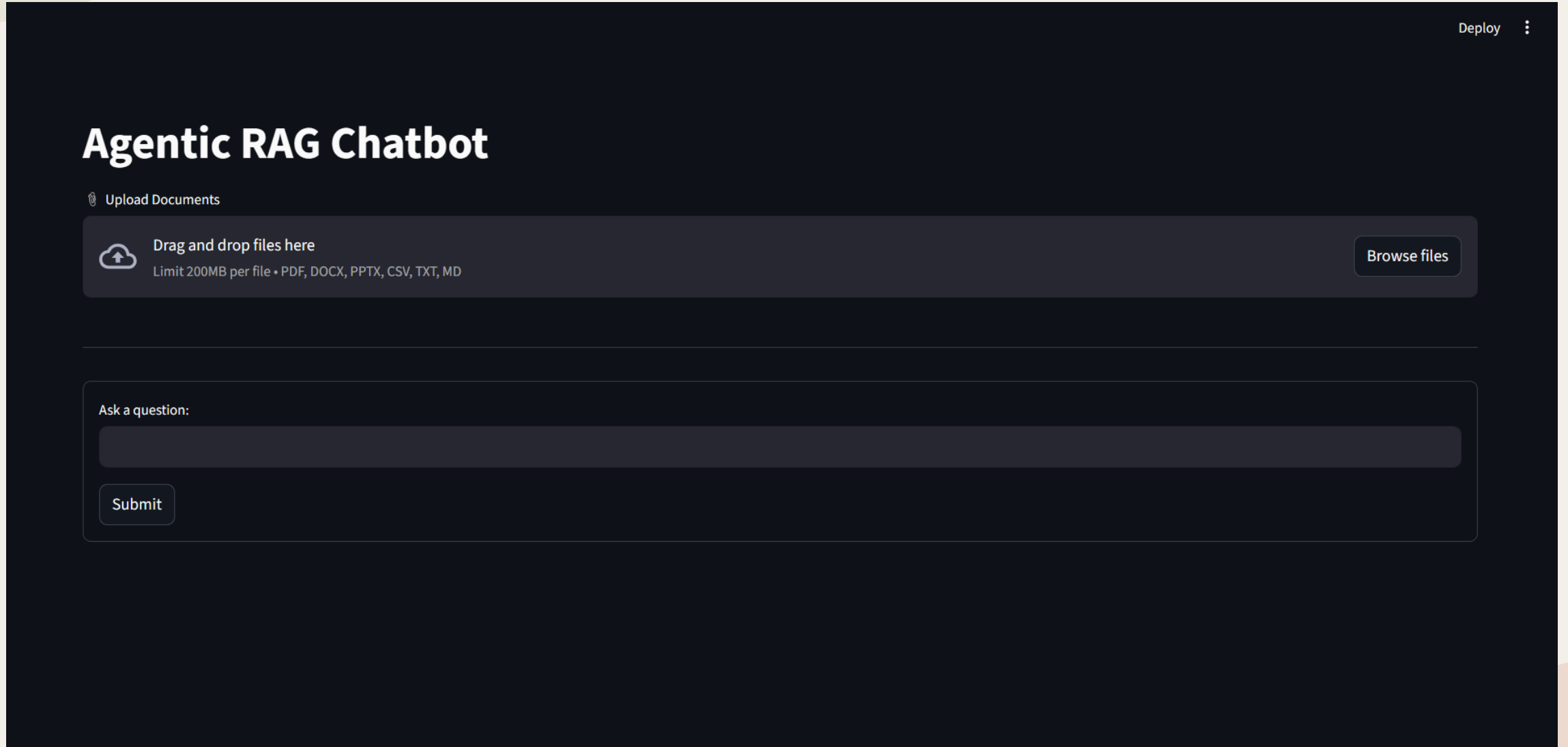
System flow diagram



Tech Stack

- Python,
- Streamlit
- Sentence-Transformers (all-MiniLM-L6-v2)
- ChromaDB
- Google Generative AI (Gemini 1.5 Flash)

UI SCREENSHOTS



CHATBOT INTERFACE

Agentic RAG Chatbot

📁 Upload Documents



Drag and drop files here

Limit 200MB per file • PDF, DOCX, PPTX, CSV, TXT, MD

Browse files



04 Generators in Python - NikhilSharma X KirkYagami.pdf 108.8KB



03 Text Chunking in RAG Systems.pdf 135.1KB



Chat History

Q1: what are generators and explain chunking

A: Generators in Python are special functions that produce a sequence of values one at a time, instead of generating an entire list at once. They achieve this using the `yield` keyword instead of `return`. This "lazy evaluation" means they only compute and return the next value when requested, making them memory-efficient, especially when dealing with large datasets. The analogy of a book with a bookmark is apt – the generator "remembers" its position (state) after each `yield`, resuming from where it left off the next time it's called. This state-saving behavior is key to their efficiency.

Chunking isn't explicitly defined in the provided text, but it's a common application of generators. Chunking refers to processing data in smaller, manageable units (chunks) rather than all at once. With generators, you can create a function that yields each chunk of data. This allows you to process huge files or datasets without loading the entire thing into memory at once. For example, you could read a large file line by line (each line being a chunk), or read it in fixed-size blocks of bytes (each block being a chunk). This approach prevents memory overflow and allows for more efficient processing. The provided text hints at this real-world example with its mention of processing large files efficiently.

Top 3 Source Chunks



Uploaded 2 documents and Asked questions from them
and Got the answer back with the source chunks

Challenges Faced

- **Chunking Logic**
- Splitting long docs into meaningful ~400-char chunks
- **Streamlit State Management**
- Avoiding double-submit / rerun issues
- **ChromaDB Migration**
- Upgrading to new PersistentClient API
- **MCP Coordination**
- Ensuring correct traceable message passing