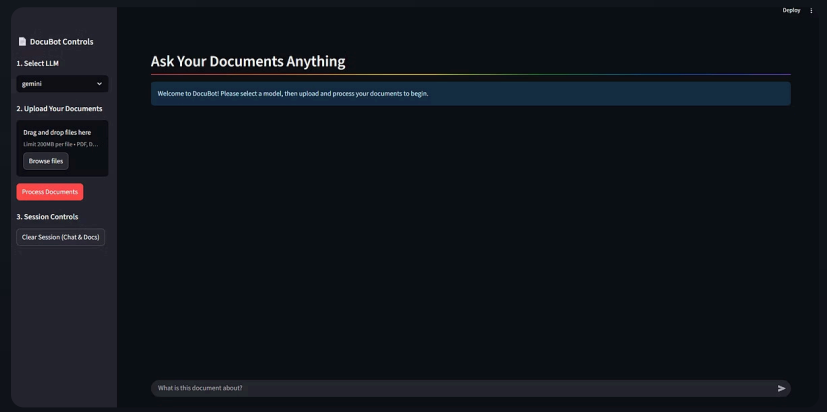


Agentic RAG Chatbot

Multi-Format Document QA using Model Context Protocol (MCP)

Technical Implementation Overview

Project Architecture

The application uses a decoupled architecture with a FastAPI backend (the 'brain') and a Streamlit frontend (the 'face').

User Interface

Streamlit frontend for user interaction and file uploads

Backend API

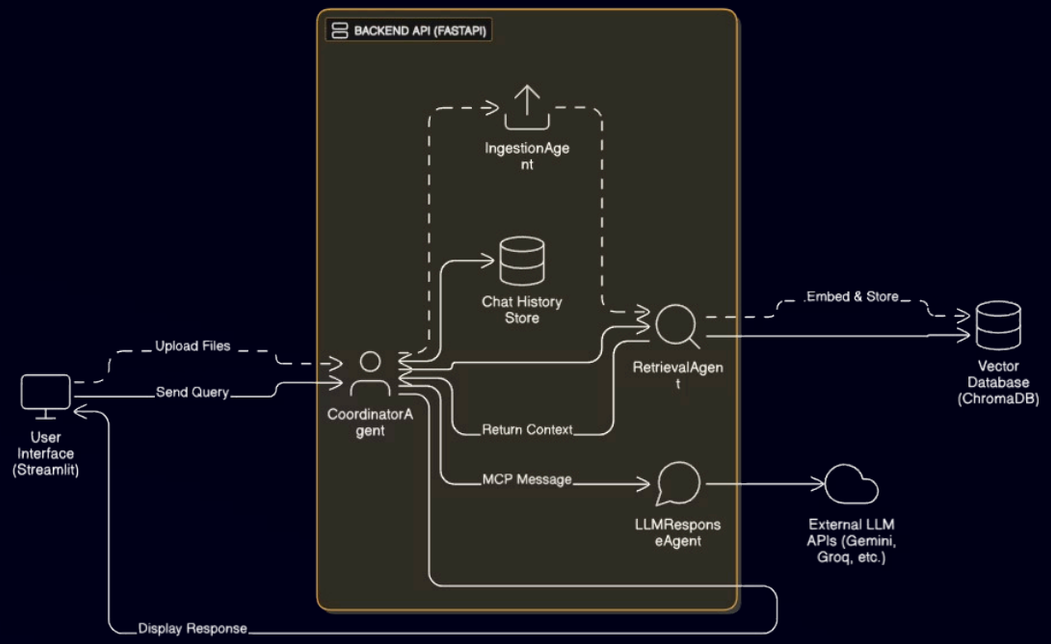
FastAPI CoordinatorAgent with three specialist agents

Vector Database

ChromaDB stores document embeddings for semantic search

External LLM APIs

Pluggable interface for Gemini, Groq, and Hugging Face



System Flow & Message Passing

The lifecycle of a user query involves coordinated communication between agents using MCP messages.

1

User Query

HTTP POST /query from Streamlit UI to CoordinatorAgent

2

Document Retrieval

RetrievalAgent searches Vector DB for relevant context

3

MCP Message Creation

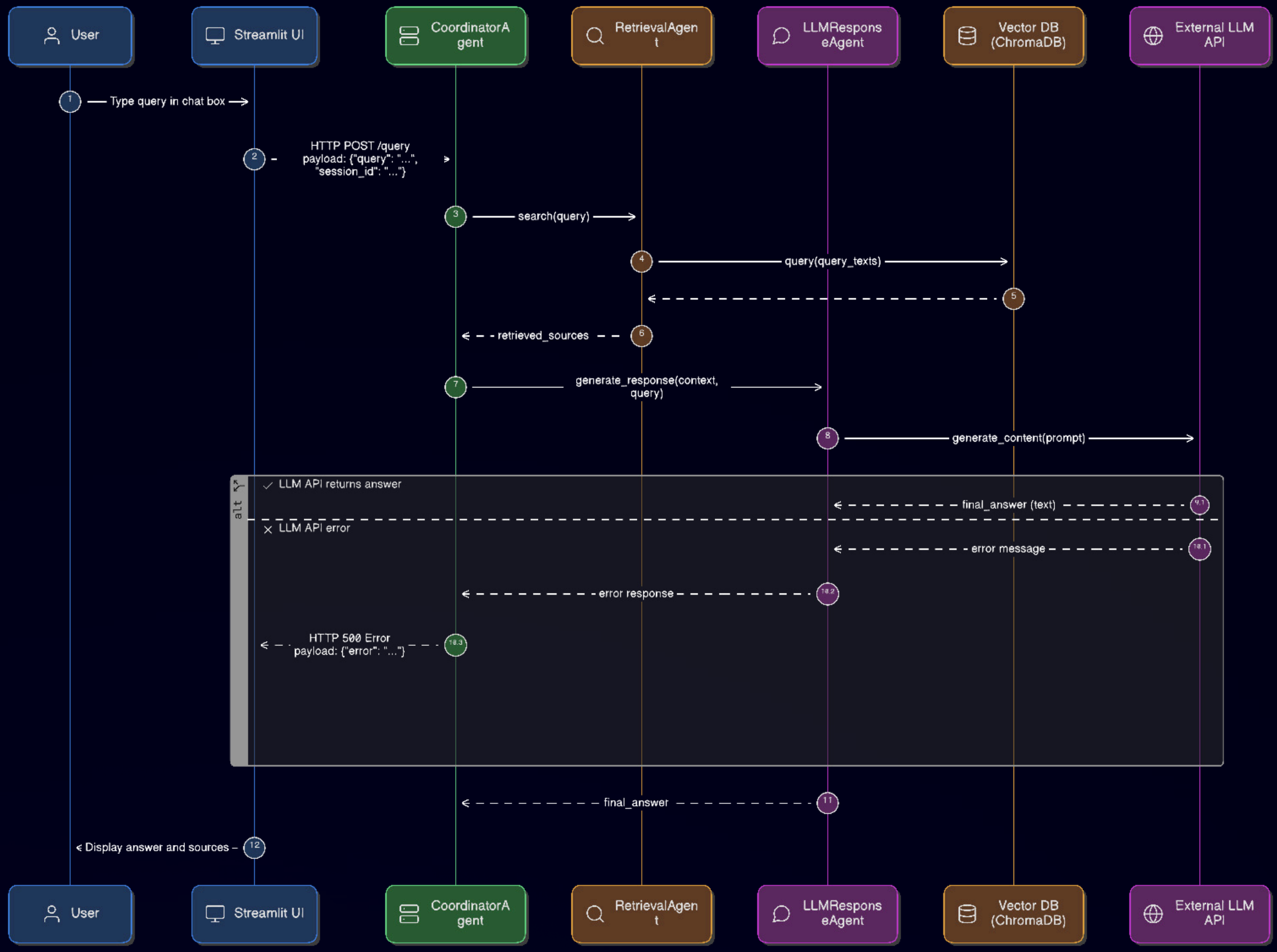
CoordinatorAgent creates structured message payload

4

Response Generation

LLMResponseAgent generates final answer using context

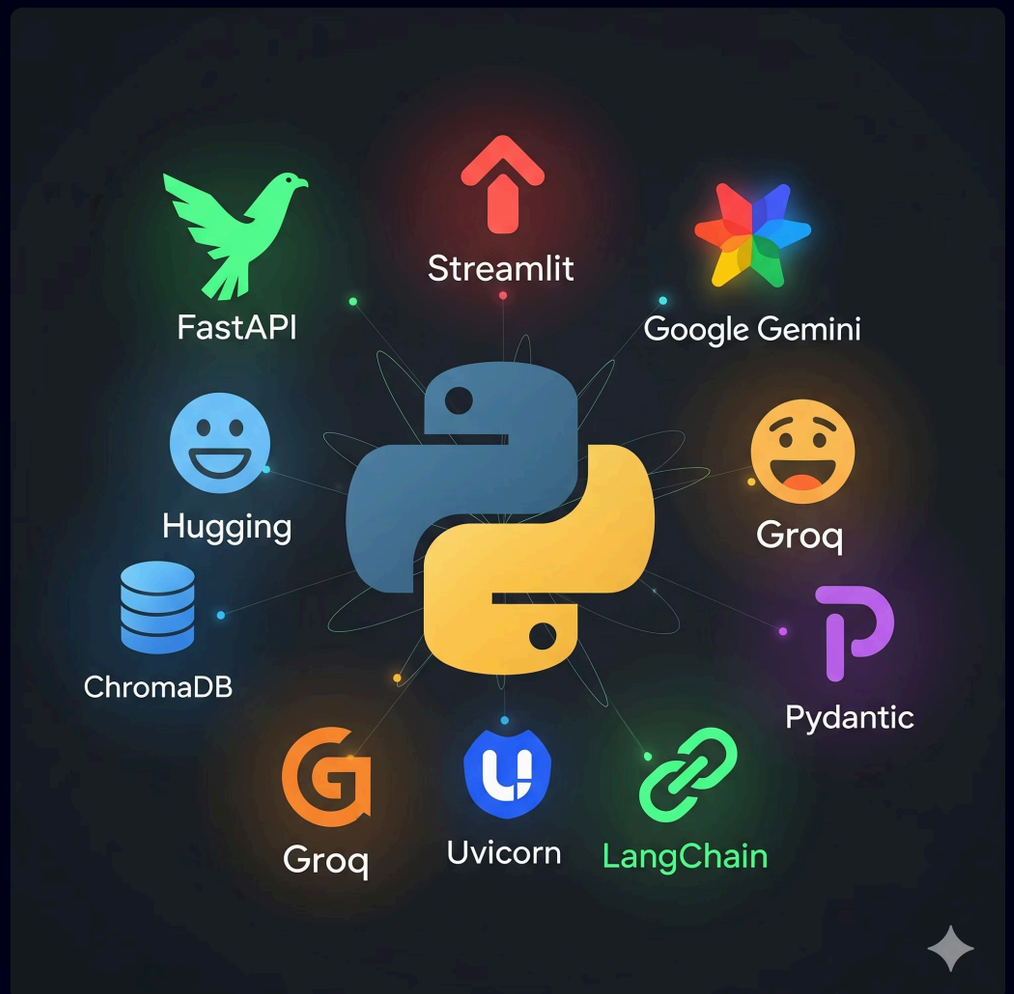
```
{
 "sender": "CoordinatorAgent",
 "receiver": "LLMResponseAgent",
 "type": "CONTEXT_RESPONSE",
 "payload": {
 "retrieved_context": "[...]",
 "query": "..."
 }
}
```



Technology Stack

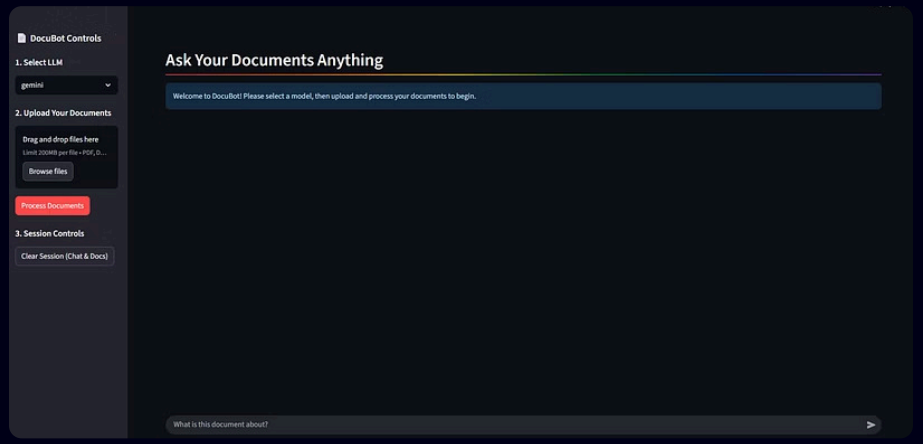
Tech Stack

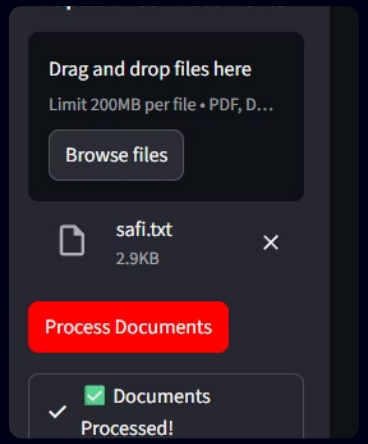
- Backend: FastAPI, Uvicorn
- Frontend: Streamlit
- Vector DB: ChromaDB (in-memory)
- Embeddings: SentenceTransformers
- LLMs: Google Gemini, Groq, Hugging Face
- Core Libraries: Pydantic, LangChain
- Package Manager: uv

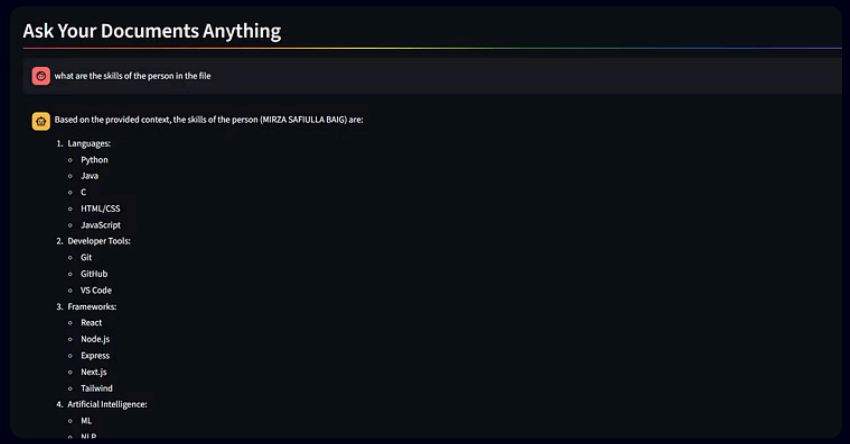


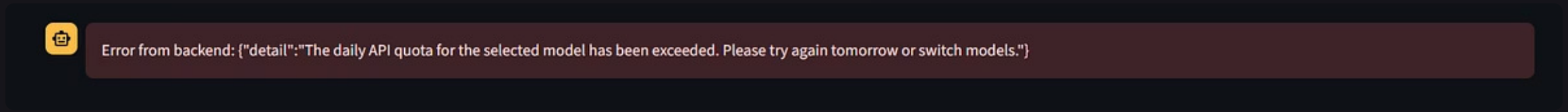
Live application interface with document processing and chat functionality

Application Demo









Key Challenges & Solutions

Retrieval Failure

Challenge: Queries failed to find relevant chunks due to semantic differences.

Solution: Increased top_k retrieved documents for broader context.

API Rate Limiting

Challenge: Gemini API quota exhaustion caused backend crashes.

Solution: Implemented robust error handling with user-friendly messages.

Prompt Engineering

Challenge: LLM confusion between task instructions and document content.

Solution: Clear persona definition and strong context delimiters.

Future Improvements

Persistent Storage

Upgrade to disk-based ChromaDB and Redis cache for production-ready stateful application.

Advanced Retrieval

Implement Multi-Query Retrieval with LLM-generated question variations for resilient search.

Asynchronous Processing

Move large document ingestion to background task queue for non-blocking user experience.