



# AGENTIC RAG CHATBOT

for Multi-Format Document QA



-BY NIKHIL SUKTHE



# PROBLEM STATEMENT

- Build a Retrieval-Augmented Generation (RAG) chatbot
- Capable of answering questions from diverse document formats
- Must follow an agent-based architecture
- Use Model Context Protocol (MCP) for agent communication

# PROJECT GOALS

- Develop modular, agent-based QA system
- Implement MCP messaging
- Provide a user-friendly document upload + query interface

# TECH STACK

## Languages & Libraries:

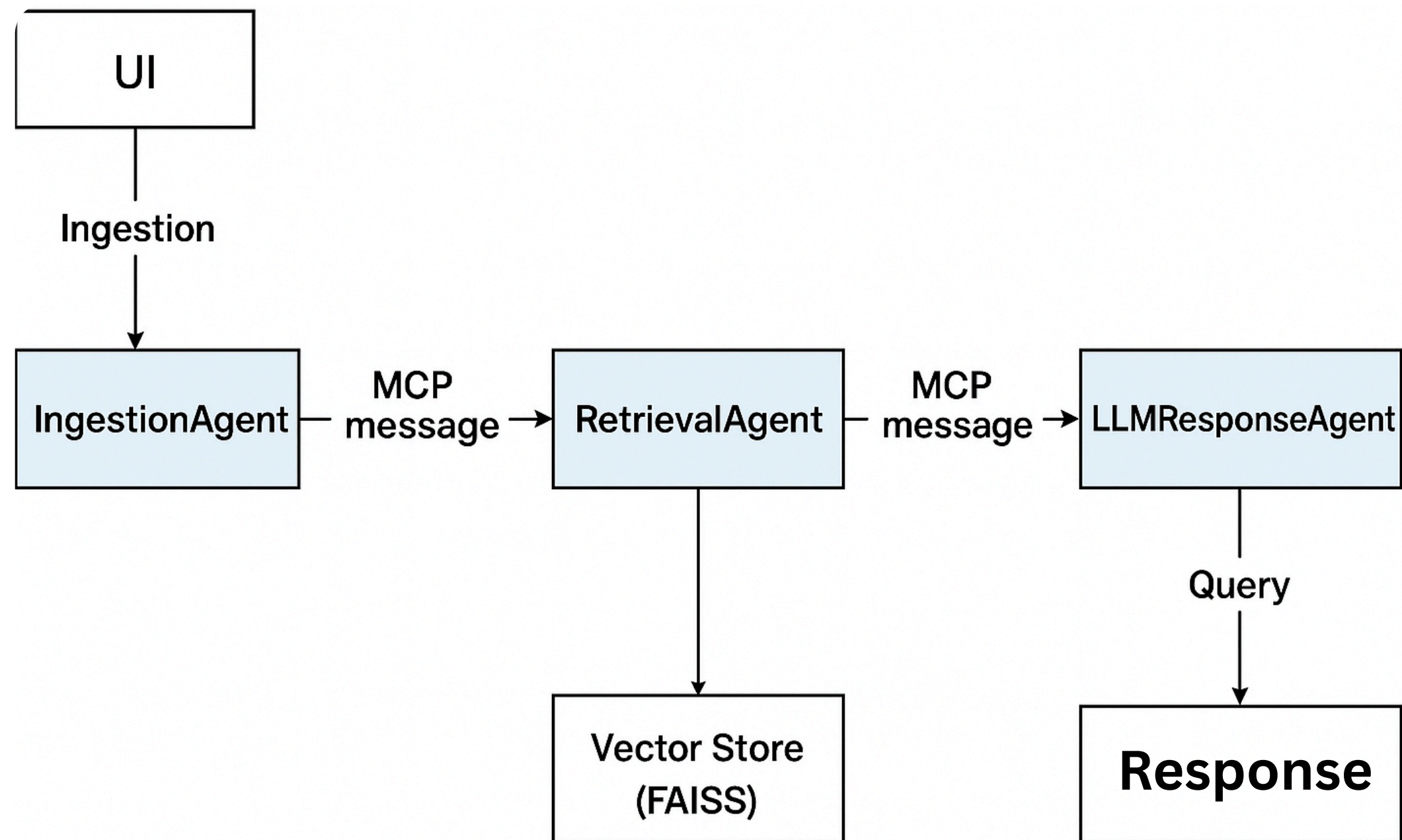
- Python 3
- Streamlit (UI)
- FAISS (Vector Store)
- Cohere API (LLM)
- Sentence Transformers (Embeddings)
- python-docx, python-pptx, PyMuPDF, pandas (Parsers)

# TECH STACK

## Architecture Principles:

- Agent-based modularity
- In-memory MCP messaging
- RAG + vector retrieval

# AGENT-BASED ARCHITECTURE WITH MCP INTEGRATION




# SYSTEM WORKFLOW


## STEPS:

- User uploads document via UI
- UI sends DOCUMENT\_UPLOAD message via MCP
- IngestionAgent parses document + stores chunks in Vector Store
- User enters question
- UI sends QUERY\_REQUEST message via MCP
- RetrievalAgent retrieves relevant context
- Sends CONTEXT\_RESPONSE to LLMResponseAgent
- LLMResponseAgent forms final prompt, sends to Cohere API
- Answer displayed on UI

# UI SNAPSHOTS


Deploy

 **Agentic RAG System**

 **Document Upload**

Upload your document to begin intelligent analysis and querying

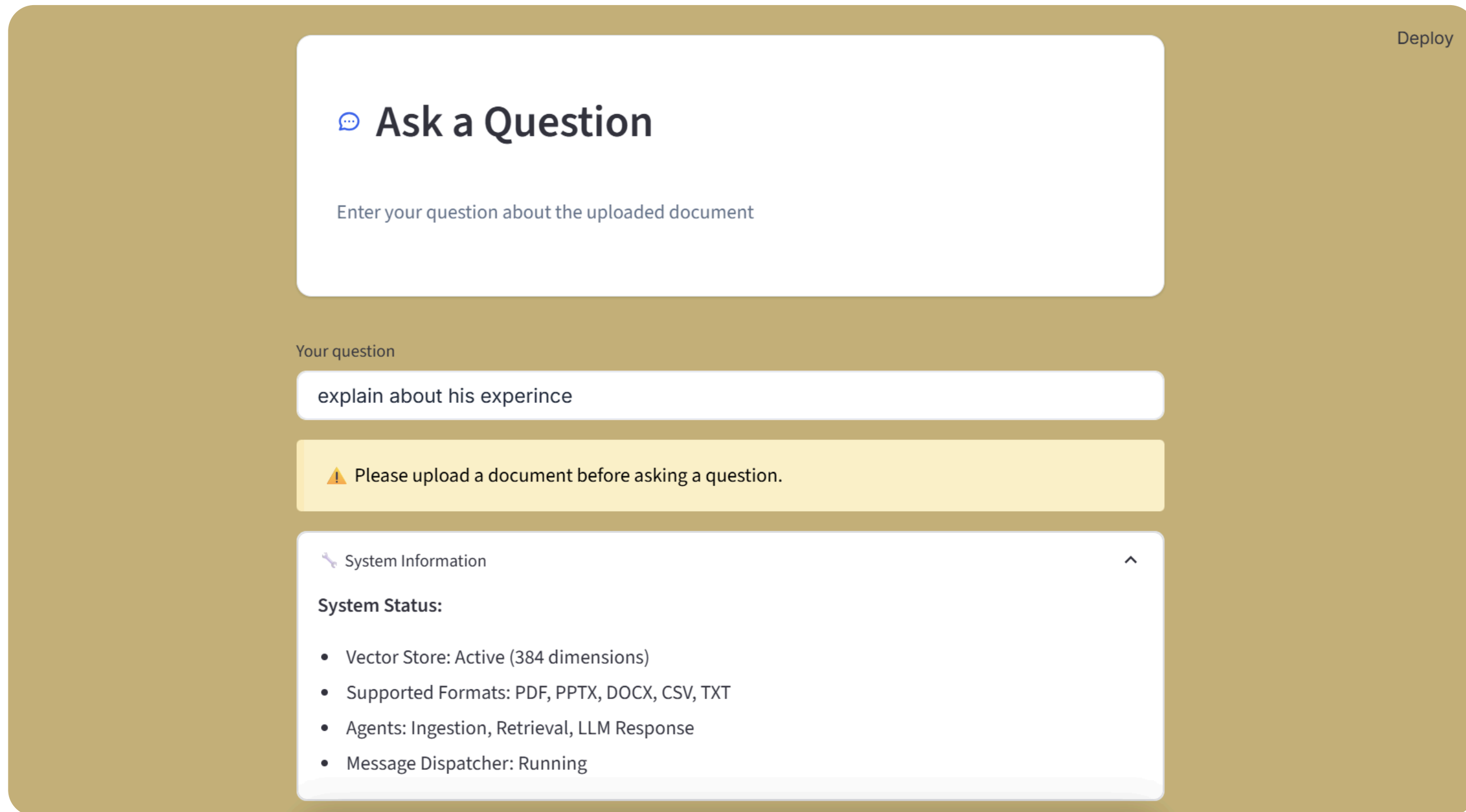
Choose a file

 Drag and drop file here  
Limit 200MB per file • PDF, PPTX, DOCX, CSV, TXT

Browse files



# UI SNAPSHOTS



# CHALLENGES

Challenges include ensuring document chunk consistency across formats, tuning FAISS indexing for smaller datasets, effectively handling multi-turn queries with context retention, and adapting response formatting to suit different contexts.

# FUTURE SCOPE

Future scope includes enhancing multi-turn conversational memory for deeper, more contextual interactions; expanding support for cloud-based vector databases like Pinecone and Chroma to improve scalability and performance; evolving the system into a robust API-based microservices architecture for greater modularity; and integrating additional LLMs such as Gemini and Mistral to offer broader model flexibility and capabilities.

# CONCLUSION

This project successfully demonstrates the power of an agent-based Retrieval-Augmented Generation (RAG) system built on the Model Context Protocol (MCP). By integrating modular agents for document ingestion, retrieval, and LLM response generation, the system offers a scalable, multi-format document QA solution. The architecture ensures clean message passing, efficient document parsing, and reliable context-aware answers — paving the way for enterprise-ready, intelligent chatbot systems capable of handling diverse and dynamic knowledge sources.



**THANK YOU**