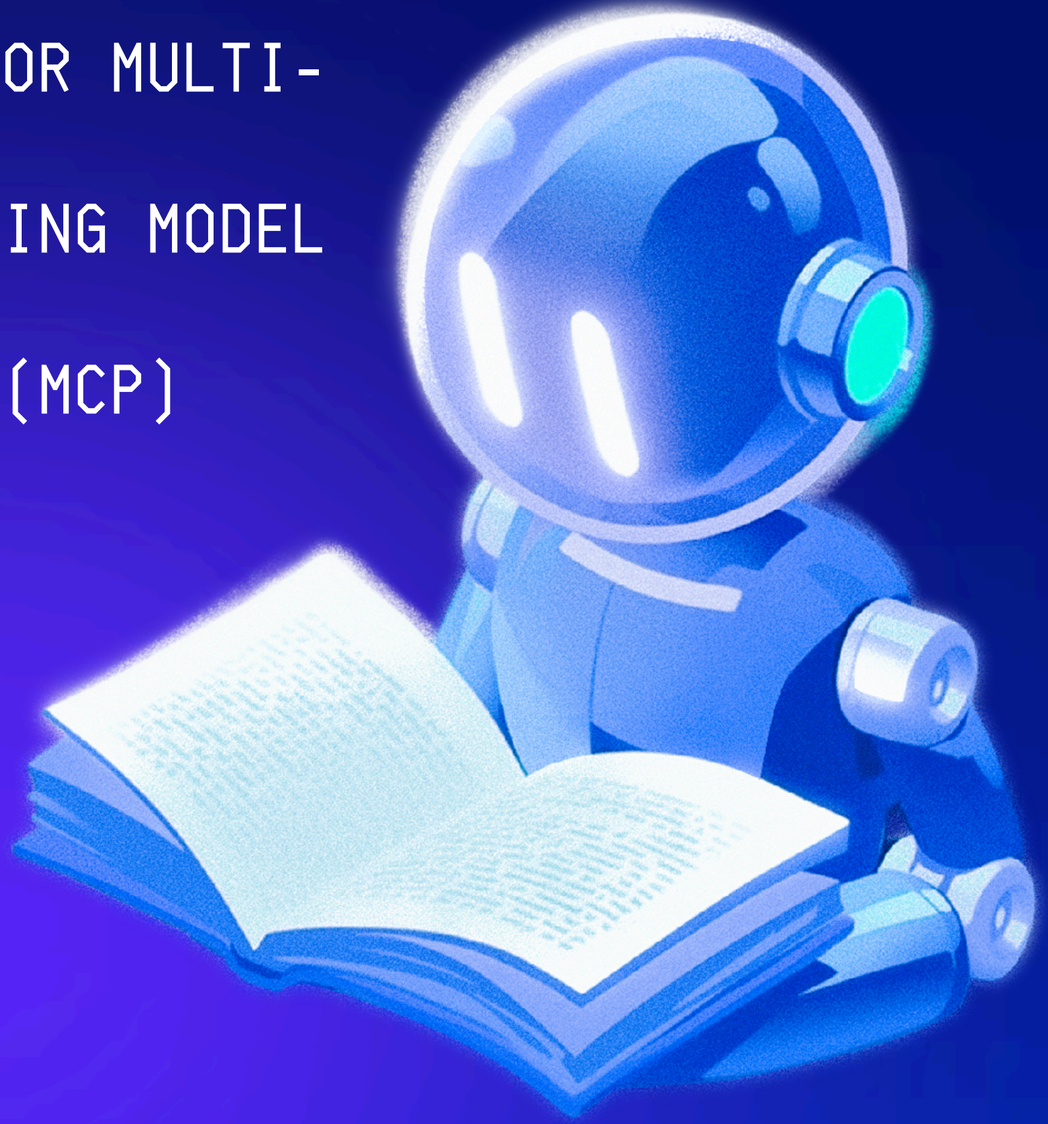
AGENTIC RAG CHATBOT FOR MULTI-FORMAT DOCUMENT QA USING MODEL CONTEXT PROTOCOL (MCP)



By Karthik

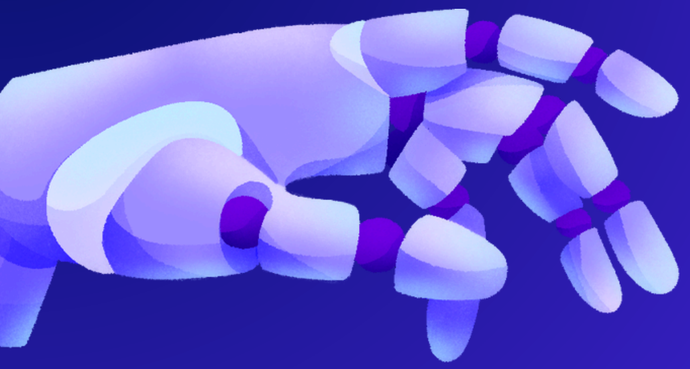
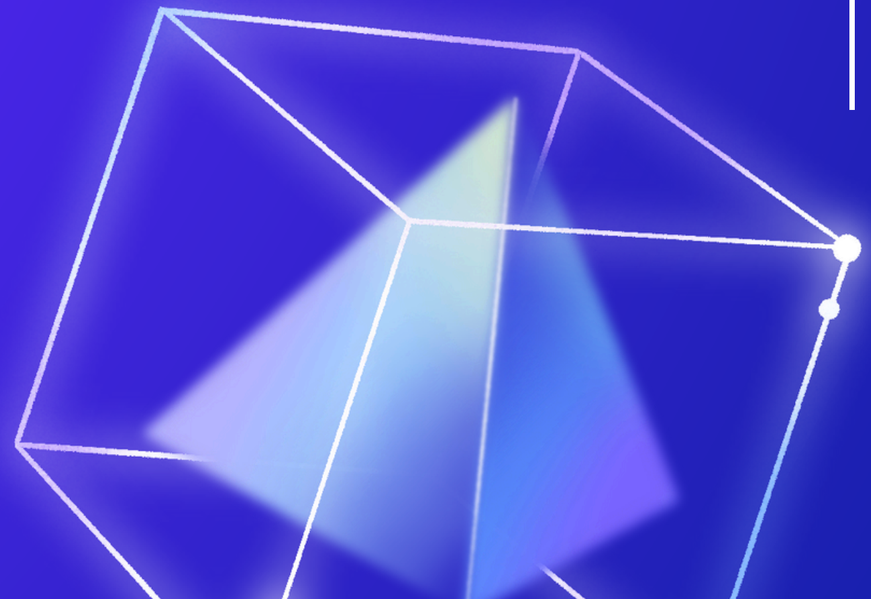


TABLE OF CONTENTS



PROJECT OVERVIEW & TECHNOLOGY STACK

Design a smart and modular chatbot system that can extract answers from documents of various formats (PDF, PPTX, DOCX, CSV, TXT) using a Retrieval-Augmented Generation (RAG) architecture backed by agents.

🧠 Key Highlights:

- Agent-based design for modularity and scalability.
- Integrated LangGraph to simulate agent collaboration via Model Context Protocol (MCP).
- Multi-turn, real-time conversational UI using Gradio.
- Lightweight, fast, and deployed fully on Google Colab.

🔧 Tech Stack:

- LLM: Gemini 2.0 Flash (Fast + Accurate)
- Embeddings: Google Embeddings 001
- Frameworks: LangChain + LangGraph
- Vector Store: ChromaDB
- UI: Gradio
- Platform: Pycharm (runtime environment)

AGENTIC ARCHITECTURE WITH MCP PROTOCOL

🧩 Core Agents:

- IngestionAgent: Reads, parses, and chunkifies uploaded documents.
- RetrievalAgent: Computes embeddings and retrieves top matching chunks.
- LLMResponseAgent: Prepares final prompt using context and invokes the LLM.

🔗 Model Context Protocol (MCP):

- Custom JSON-like message format used for agent communication.
- Example Message:

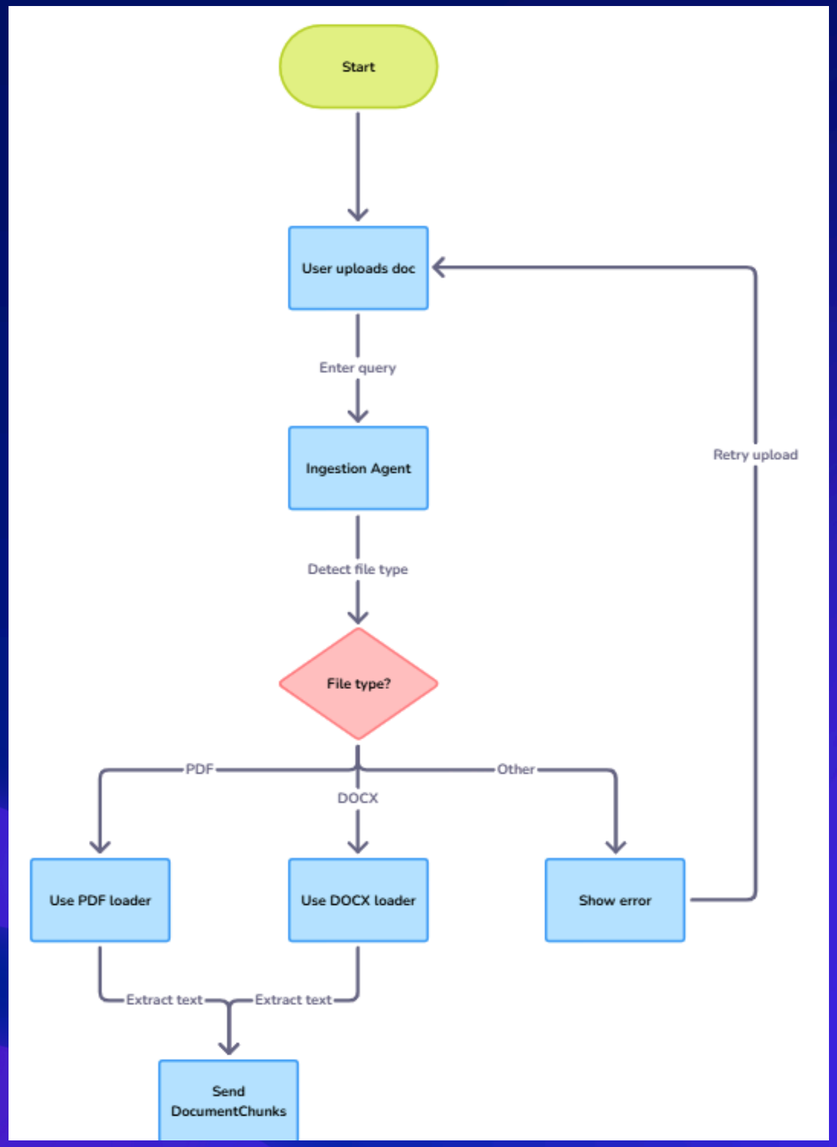
```
{
  "type": "RETRIEVAL_RESULT",
  "sender": "RetrievalAgent",
  "receiver": "LLMResponseAgent",
  "trace_id": "rag-457",
  "payload": {
    "retrieved_context": ["slide 3: revenue up", "doc: Q1
summary..."],
    "query": "What KPIs were tracked in Q1?"
  }
}
```

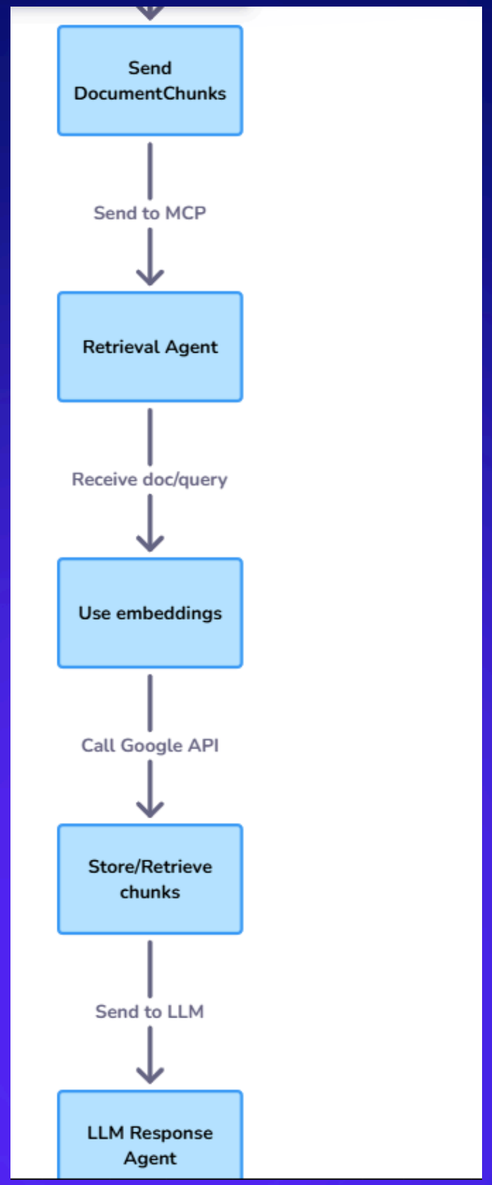
✅ Benefits:

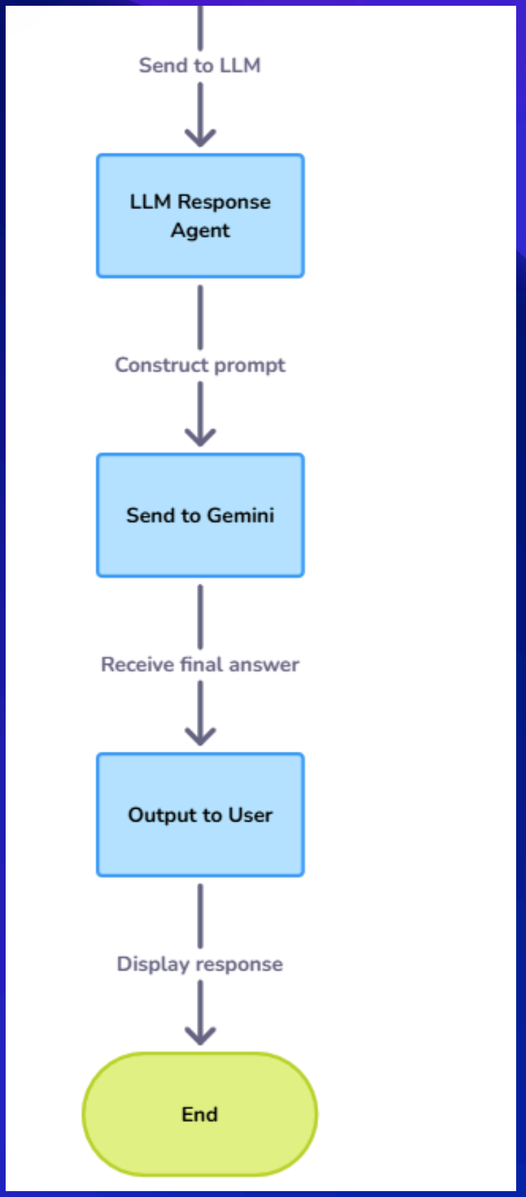
Enables stateless, traceable, and scalable agent communication.

Decouples responsibilities for each task.

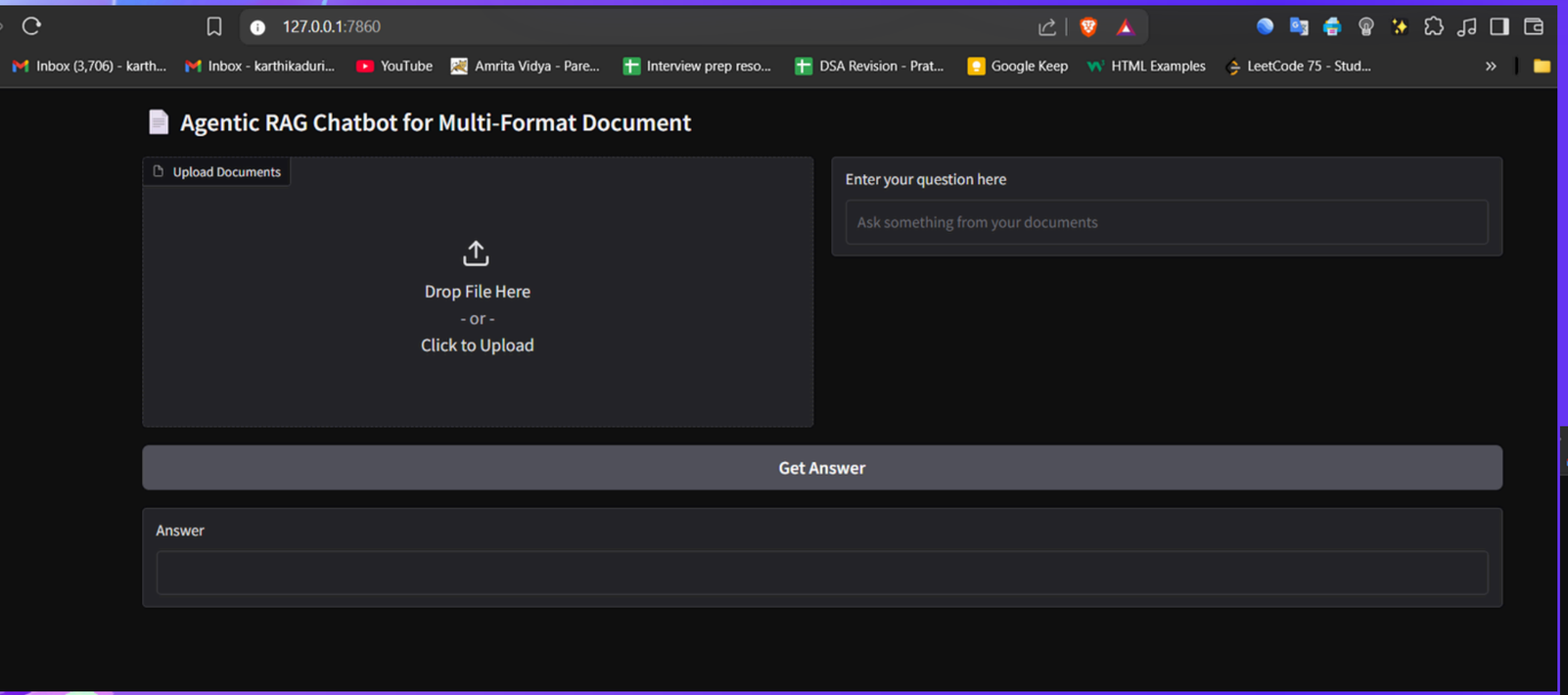
SYSTEM FLOW & LANGGRAPH EXECUTION

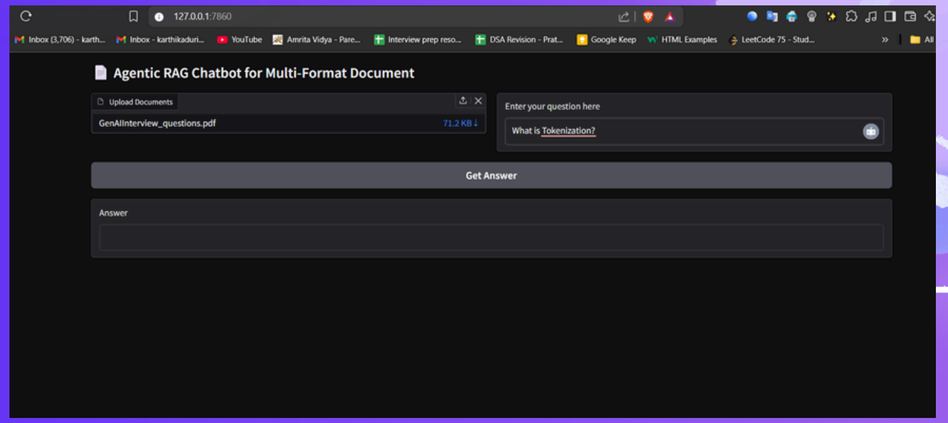


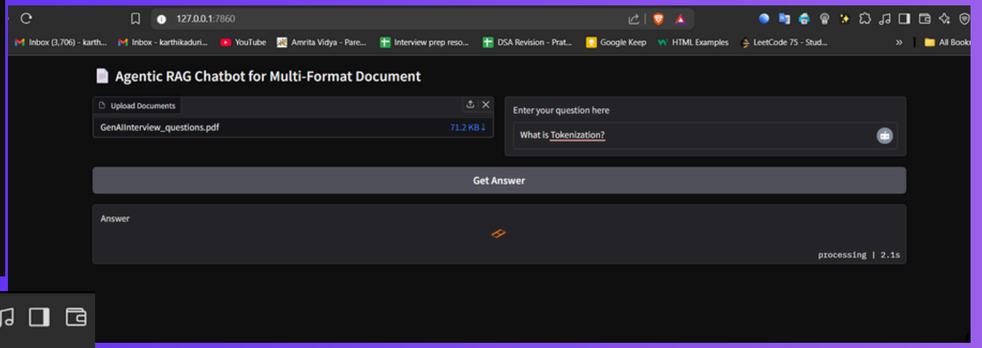


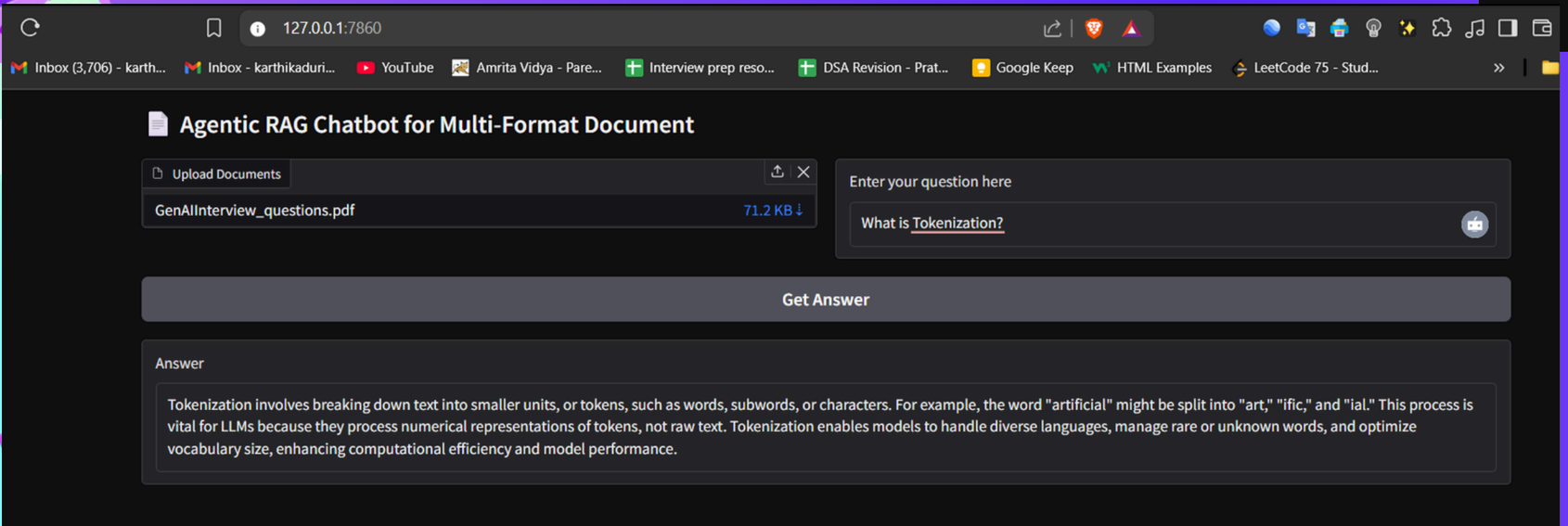


GRADIO UI - APPLICATION SNAPSHOTS









CHALLENGES FACED DURING DEVELOPMENT

- 🧠 Initial LLM integration issues:
- Originally explored VertexAI but replaced it with Gemini Flash due to API restrictions on Colab.
- 📂 Multi-format parsing difficulties:
- Handling edge cases in malformed or image-heavy PDFs and inconsistent DOCX formatting.
- 🔄 LangGraph Complexity:
- Understanding node-based agent flow and structuring MCP message passing took time.
- Initial struggles with Gradio deployment on Colab due to ngrok connection issues so changed to pycharm.

FUTURE SCOPE & ENHANCEMENTS

- 📚 More File Support: Add OCR for scanned documents and images (e.g., invoice PDFs).
- 🔒 User Authentication: Allow per-user history, document access control.
- 💬 Conversation Memory: Use LangChain memory components for multi-turn coherent chat.
- 📊 Response Confidence Scores: Let users understand the confidence of answers.
- 🧪 Experiment with Local LLMs: Reduce cloud dependence and improve privacy.

THANK YOU!