🏠 Agentic RAG Chatbot

A Multi-Format Document QA Bot Using MCP & Agentic Architecture

Divyansh Gautam

Date: 25th July 2025

🔍 Problem Statement & Objective

The Goal:

Build an **Agentic Retrieval-Augmented Generation (RAG)** chatbot that:

- Accepts **PDF, PPTX, DOCX, TXT, CSV** files
- Answers user queries with document-grounded responses
- Uses **structured agent communication** via Model Context Protocol (MCP)
- Offers a real-time chatbot interface

🏛 Architecture Diagram

```
User → Streamlit UI
        └→ IngestionAgent (Parse documents)
                  └→ RetrievalAgent (Embed & Retrieve Chunks)
                            └→ LLMResponseAgent (Generate Answer)
                                      └→ Display Response + Source
```

All agent communication is via structured **MCP messages** with sender, receiver, type, trace_id, and payload.

MCP Message Format

```
{
  "sender": "RetrievalAgent",
  "receiver": "LLMResponseAgent",
  "type": "RETRIEVAL_RESULT",
  "trace_id": "abc-123",
  "payload": {
    "retrieved_context": ["chunk1", "chunk2"],
    "query": "Summarize the Q2 performance"
  }
}
```

This protocol ensures **clear, modular agent coordination**.

Model Comparison

Feature	Mistral 7B (Colab)	FLAN-T5 Base (Streamlit UI)
Performance	Excellent results	Medium clarity
Resource Needs	High (GPU required)	Lightweight (runs locally)
Deployment Feasibility	No (locally)	Yes

Mistral Output Sample:

Q: What were Q2 improvements?
A: Reduced churn by 10%, Improved delivery TAT by 25%, Introduced voice-based ordering.

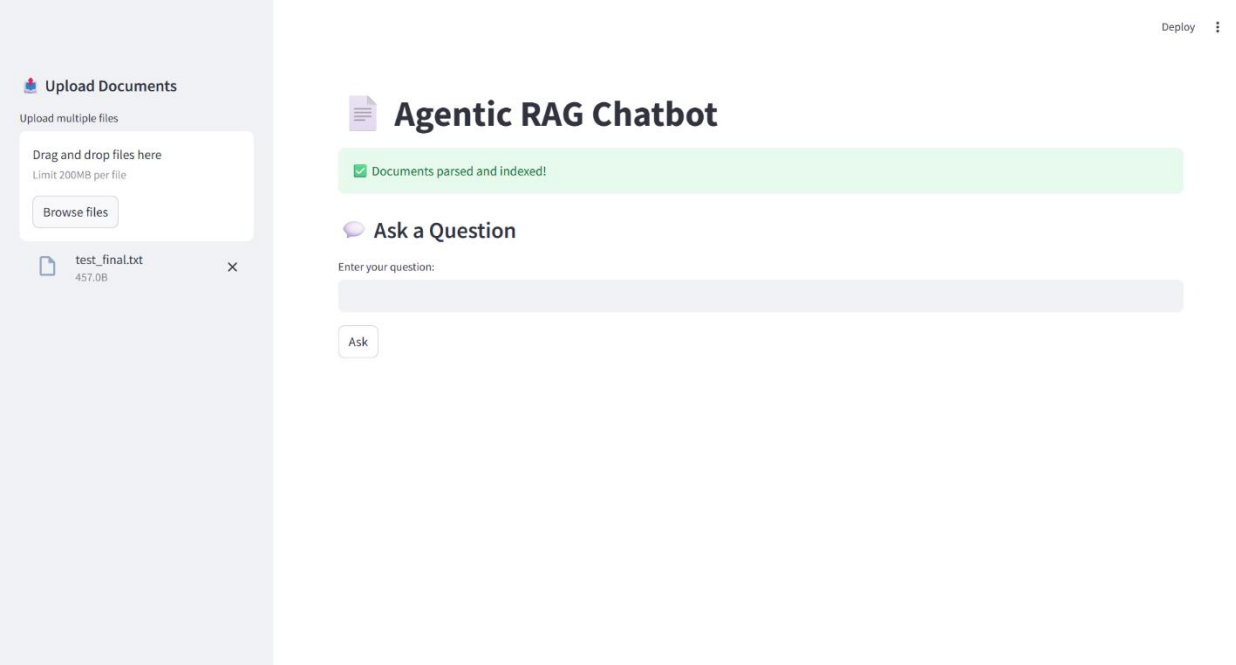
FLAN Output Sample:

Q: What were Q2 improvements?
A: Repeats context, partial info, less concise

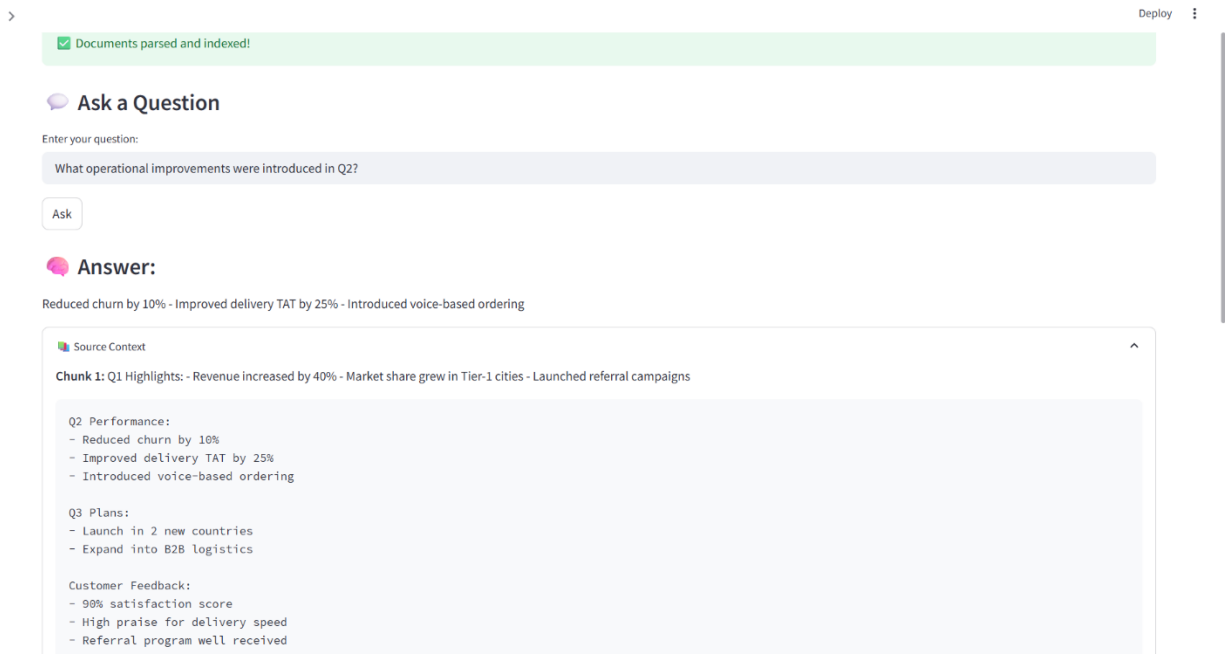
Streamlit UI Snapshots

More screenshots are available in the Screenshots Folder. Some of them are below:

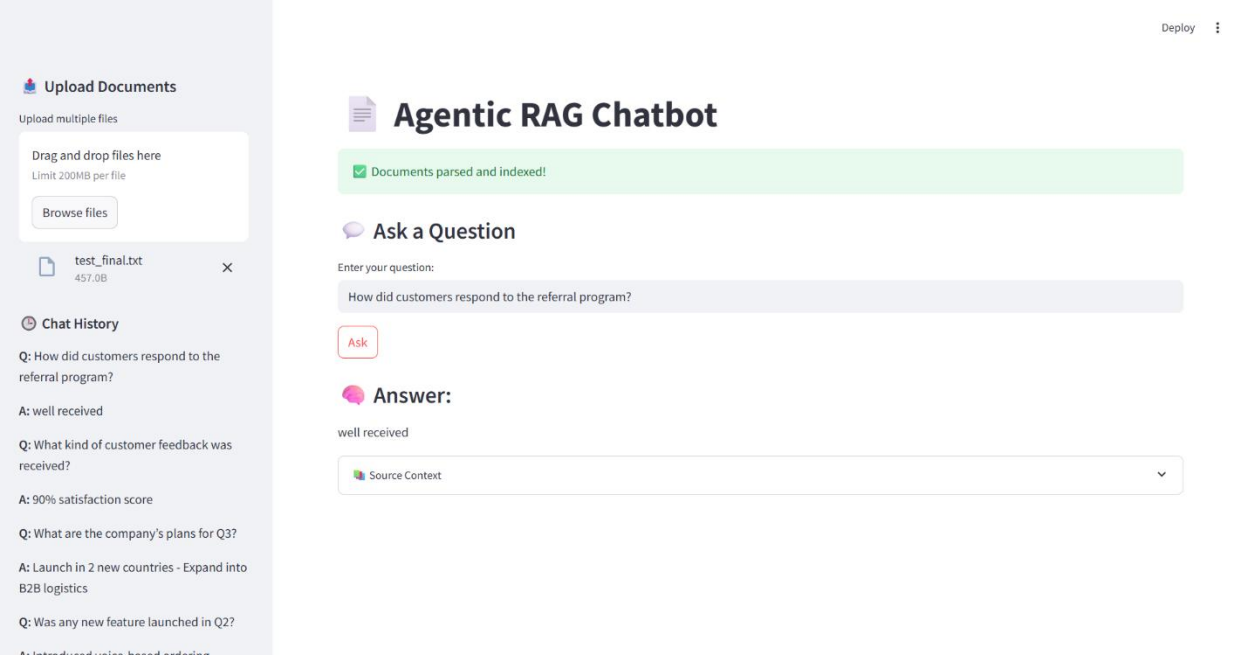
File Upload & Document Parsing



Q2 Improvement



Referral Campaign Response



Tech Stack

- **LLMs**: Mistral-7B, FLAN-T5-base, FLAN-T5-large
- **Embeddings**: MiniLM (via sentence-transformers)
- **Vector DB**: FAISS
- **Parsing**: PyMuPDF, python-docx, python-pptx
- **Frontend**: Streamlit
- **Backend Logic**: MCP agent handlers in Python

Challenges & Learnings

- Mistral couldn't be deployed via Streamlit locally due to system RAM limits
- Needed chunking strategies to prevent noisy answers
- Streamlit required model-light architecture for real-time UX
- PPTX files occasionally had malformed `shape.text` → handled

🌟 Future Improvements

- Add a **CoordinatorAgent** to manage all agent flow
- Serve Mistral via **API** (e.g., Hugging Face Inference endpoint)
- Add **query-type detection** (metrics, timelines, summaries)
- Smarter chunking: sentence-boundary and title-aware

📄 Deliverables Recap

- ✅ GitHub Repository with all code & test files
- ✅ README.md with architecture, instructions, sample results
- ✅ Screenshots of UI flow
- ✅ This PPT (Architecture + Workflow + Comparison)
- ✅ Video Walkthrough (App Demo + Architecture + Code Tour)

🎯 Conclusion

This project demonstrates the power of modular agent-based design when combined with context-aware semantic search and strong LLM backends. Despite hardware limitations, the system creatively leveraged FLAN-T5 for UI deployment and Mistral for deep QA in notebook. The result is a scalable, interpretable, and high-performing chatbot capable of document-grounded answers across file formats.

🙏 Thank You!

Submission by:

Divyansh Gautam GitHub: Agentic-RAG-Chatbot