Pharma Regulatory Chatbot - Final Project Report

**Project Title:** Pharma Regulatory Chatbot Using Mistral-7B and Streamlit UI.

**Author:** Tvisha Shetty

**Tools Used:** Python, LangChain, Hugging Face, FAISS, Together API, Streamlit

**Abstract:**  
This project involves building a smart pharma regulatory chatbot that reads regulatory PDF documents (SOPs, FDA guidelines, cGMP policies) and answers user queries based on document context. It integrates LangChain for chunking, FAISS for vector search, Mistral-7B via Together API for natural language understanding, and Streamlit for an interactive UI.

This chatbot ensures regulatory compliance , thereby reducing manual document lookup.

**Introduction:**

In the pharmaceutical industry, ensuring adherence to regulatory norms and guidelines is critical for quality control and compliance. This project presents an intelligent chatbot system designed to simplify access to standard operating procedures (SOPs), cGMP guidelines, and FDA regulations. By leveraging powerful natural language processing (NLP) and retrieval techniques, the chatbot allows quality assurance (QA) and quality control (QC) professionals to query extensive documentation using natural language and receive accurate, context-bound answers instantly. The system brings together modern AI capabilities such as LangChain, Hugging Face embeddings, FAISS vector databases, and Mistral-7B LLM, wrapped within a secure, user-friendly Streamlit interface.

**Objective:**

The primary objective of this project is to develop a secure, organization-ready chatbot that can intelligently answer queries related to pharmaceutical regulatory documents such as SOPs, FDA guidelines, and cGMP policies. It aims to automate the retrieval of regulatory knowledge from uploaded documents, eliminate guesswork by enforcing context-limited answers, and ensure accuracy through structured prompt engineering. The chatbot supports multi-document ingestion, interactive querying, and is optimized for deployment in VPN-secured private networks using a hidden password system to restrict access.

**Goal:** Build a chatbot that answers queries about SOPs, cGMP guidelines, or FDA regulations using pre-loaded PDFs .  
**Tech:** LangChain + OpenAI/Mistral + local PDF data (offline) + CNN + NLP  
**Value:** Helps QA/QC teams quickly access process steps and regulatory norms.

**Project Highlights:**

* Successfully integrated LangChain and Mistral-7B to create a context-aware regulatory assistant.
* Achieved seamless ingestion of PDF and DOCX files, with auto chunking, vector embedding, and storage using FAISS.
* Developed CLI and Streamlit UI prototypes, later unified into a secure and styled front-end interface.
* Implemented strict prompts to avoid hallucinations and enforced a fallback message: “Not found in context.”
* Introduced a secure login system using a hidden .env password, ideal for enterprise deployments without individual user management.
* Added features such as chat export, auto-scroll, and multi-file handling for an enhanced user experience.
* Conducted thorough internal testing and validation of LLM outputs against source documents to improve reliability.

**Tools & Technologies:**

1. **LangChain**: Used for managing document processing.
2. **FAISS**: For fast vector-based similarity searches.
3. **Hugging Face Embeddings**: Converts text chunks into numerical vector representations.
4. **Together.ai API**: Hosts and accesses the Mistral-7B-Instruct model.
5. **Streamlit**: For building a sleek, interactive web interface.
6. **PyPDFLoader / DirectoryLoader**: For reading text from PDFs.
7. **.env / dotenv**: Secure handling of API keys and environment variables.
8. **CNN + NLP Techniques**: Ensures query parsing and document understanding.

**System Architecture:**

1. **PDFs and DOCs Uploaded** via UI or backend path.
2. Text is **split into chunks** using RecursiveCharacterTextSplitter.
3. Chunks are **embedded** using Hugging Face embedding models.
4. **Embeddings are stored** in FAISS vector database.
5. On query, top k similar chunks are **retrieved** from FAISS.
6. Mistral-7B is queried via Together API using a **strict custom prompt**.
7. Answer is rendered in a styled Streamlit **chat interface**.

**Setup :**

* Loaded all .pdf files in bulk using DirectoryLoader.
* Used RecursiveCharacterTextSplitter with chunk size 500 and overlap 50.
* Used all-MiniLM-L6-v2 embedding model.
* Stored embeddings into a FAISS vector store.
* Printed chunk stats and page length to verify proper loading.

**Challenges:** - Ensuring all PDFs are valid and loaded. - Debugging directory path issues.

**Outcome:** - Successfully created a ready-to-query vector DB with regulatory content.

**Backend Development :**

* Created a multi-PDF chatbot.
* Integrated Mistral-7B-Instruct model via Together.
* Strict prompt added to ensure non-hallucinated answers.
* Implemented logic to read multiple file paths dynamically.

**Challenges:** - Handling empty or invalid file paths.

**Solution:** - Added filtering logic for endswith(".pdf") and os.path.exists().

**Frontend UI :**

* Created a minimal, dark-themed UI using Streamlit.
* Added PDF uploader, chat input, download chat, and styled CSS.
* Bot messages and user messages are color-coded with bubbles.

**Key UI Features:** - Live QA, multi-PDF upload, exportable history.

**Challenges:** - Streamlit reactivity and rerun issues.

**Final Main Integration :**

* Unified authentication, backend logic, and frontend chat into one file.
* Added login screen via session state.
* Auto-scroll to latest message with JavaScript anchor.
* Implemented general query interface if no PDFs uploaded.

**Security Note: ORGANIZATION-WIDE SINGLE ACCESS:**

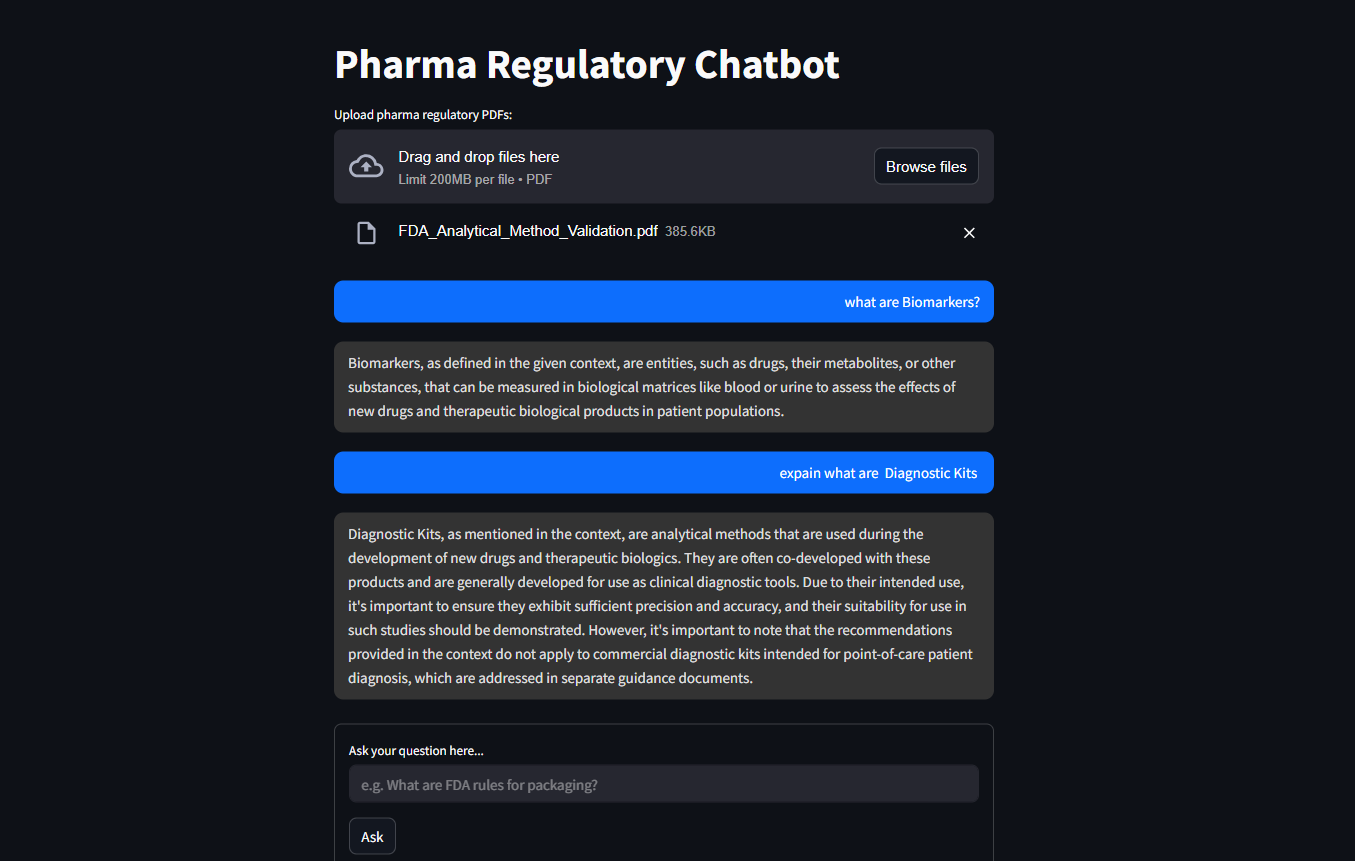
* - Hidden password in an .env file. - No hardcoding of the password in the codebase.
* - Rotate password periodically — especially if shared widely.
* - Best for private networks or internal organizational access via VPN/firewalls.
* - No need to manage a database for individual user-level credentials.

**Issues Faced & Solutions**

| Issue | Solution |
| --- | --- |
| API key not loading | Used load\_dotenv() twice in main and backend |
| Hallucinated answers | Strict custom prompt with “Not found in context” |
| Invalid PDF paths | Filtered files in CLI |
| Uploading large PDFs | Introduced tempfile handling + cleanup with os.remove() |

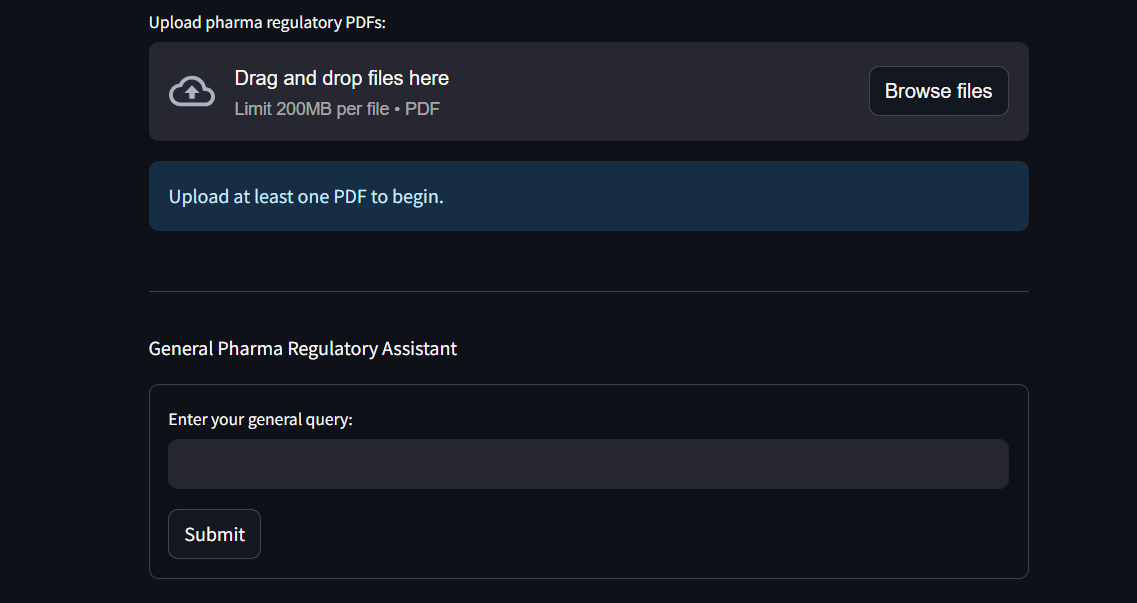
**Screenshots and Visual Flow**

1. **Login Page** – Secure password-based entry.
2. **PDF Upload + Chat UI** – Clean center layout with chat boxes



1. A screenshot of a computer

   AI-generated content may be incorrect.**Sidebar** – New Chat + Export Chat option
2. **Fallback Mode** – General query box if no PDFs uploaded.



**Conclusion & Future Scope:**

This chatbot system showcases how advanced LLMs and vector stores can be integrated into real-world enterprise use cases such as pharmaceutical compliance. It automates document reading and regulatory question answering with reliability and precision.

This project demonstrates the potential of combining large language models with retrieval-augmented generation (RAG) pipelines in highly regulated domains like pharmaceuticals. By automating information access from regulatory documents, the chatbot improves operational efficiency, reduces compliance errors, and supports internal audit readiness. Through thoughtful design, robust architecture, and strict response validation, the chatbot ensures trustworthy outputs suitable for real-world enterprise deployment. The work sets a strong foundation for further enhancements such as document source mapping, accuracy scoring, and scanned document support in future iterations.

**Future Enhancements:**

- Add OCR for scanned PDFs.

- Add long-term memory and session persistence.

- Create document source highlighting in answers.

- Build response accuracy scoring for QA testing.

**Thank you.**

**End of Report.**