Assignment: AI-Powered Medical Chatbot for Patient Understanding

Problem Definition and Key Challenges

Medical reports can be confusing and overwhelming for most patients. They're full of technical terms, numbers, and medical jargon that make it hard to understand what's actually going on with one's health. To make this easier, designing an AI-powered chatbot that can explain these reports in plain language and answer patient questions in a friendly, accurate, and secure way is a must.

This chatbot uses a large language model (LLM) to understand natural language questions and respond based on lab report data. While the idea is simple, help people understand their health better the execution comes with some important challenges:

1. Fast Access to the Right Information:
   When a patient asks a question, the chatbot needs to quickly find the relevant part of their lab report. We use a tool called FAISS to make this happen. It's a powerful way to search through large amounts of text really fast.
2. Avoiding Reuploads of the Same Data:
   Patients shouldn't have to upload the same report every time they return. To fix this, we store FAISS indexes locally, so the system remembers past uploads and avoids repeating work.
3. Keeping Answers Grounded in Reality:
   One risk with large language models is that they sometimes hallucinate, making up answers that sound right but aren't. To avoid this, we keep the model's temperature(hyperparameter) low (close to 0), which makes responses more precise and grounded in the actual data.
4. Breaking Reports Into Understandable Pieces:
   Lab reports can be long and detailed. To make them easier to work with, we break them down page by page, which helps the system process them in logical chunks and give better answers.
5. Protecting Sensitive Health Information:
   Since we're dealing with personal medical data, privacy is critical. We make sure there's login protection, user roles, and strict permissions in place to control who can access.
6. Planning for Growth and Scalability:
   Right now, we're using local storage and tools like FAISS to keep things simple. But as the system grows, we're looking ahead to cloud-based solutions like AWS S3 and HIPAA-compliant LLMs, so the chatbot can scale safely and securely.

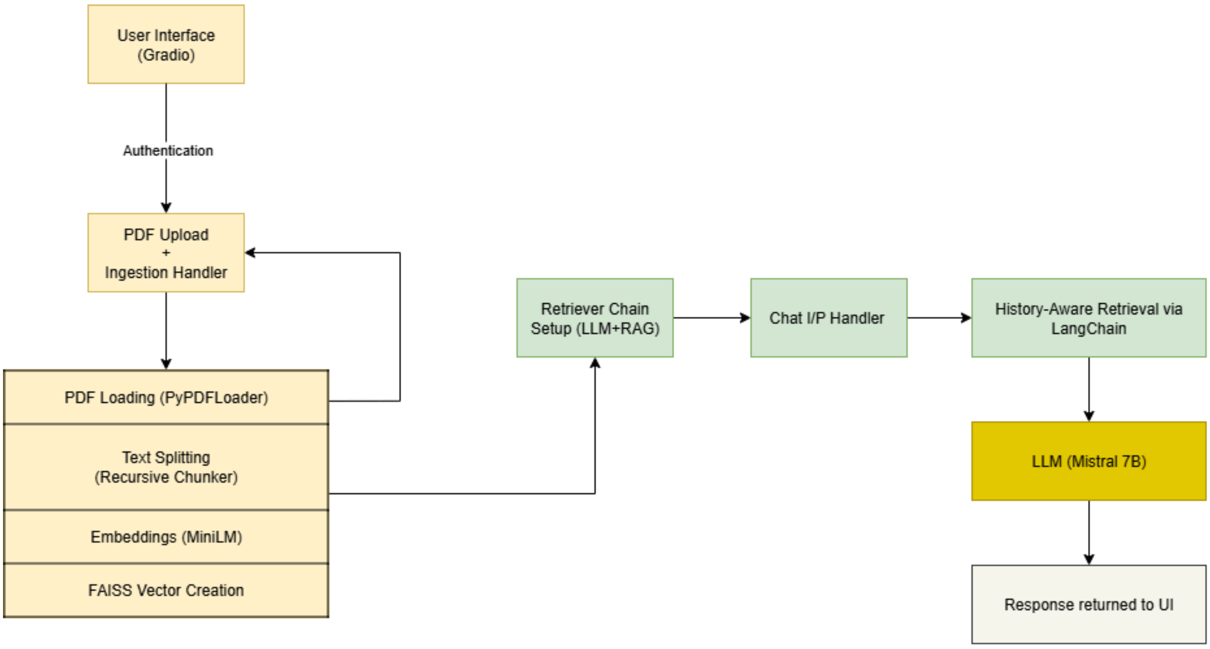
By solving these challenges, we're building a chatbot that's not just smart, but trustworthy one that helps people make sense of their health information without adding stress or confusion.

Proposed Solution Architecture:

1. User Authentication:
   The system begins by verifying user credentials through a secure login interface. Only authorized users can access the chatbot and upload sensitive medical data, ensuring privacy and compliance with data protection regulations like HIPAA.

2. PDF Upload and Preprocessing:
   Authenticated users upload their lab report in PDF format. The uploaded document is parsed, and its content is extracted for downstream processing.
3. Text Segmentation:
   The extracted content is split into smaller, semantically meaningful chunks. A page-wise or recursive character-based chunking strategy is applied to preserve context and improve the quality of document retrieval.
4. Embedding Generation:
   Each chunk is converted into a vector embedding using a lightweight, pre-trained model(MiniLM). These embeddings represent the semantic content of the document segments.
5. Vector Storage using FAISS:
   The embeddings are stored in a local FAISS vector index, which enables fast and efficient similarity search during query retrieval. Existing vector stores are reused when possible to avoid redundant processing.
6. Query Handling and Retrieval Augmentation:
   When a user submits a question, the system uses the recent chat history to refine the query. A retrieval-augmented generation (RAG) pipeline searches the FAISS index for the most relevant chunks corresponding to the refined query.
7. Answer Generation via LLM:
   The retrieved context, along with the user’s question and chat history, is passed to a local large language model (Mistral 7B via LlamaCpp). The model, configured with a low temperature to minimize hallucinations, generates a concise and accurate response.
8. Response Delivery:
   The generated answer is returned to the user through the chatbot interface. The system maintains the chat history to support context-aware multi-turn conversations.

Architecture Flow:


User Interface (Gradio)
Authentication
PDF Upload + Ingestion Handler
PDF Loading (PyPDFLoader)
Text Splitting (Recursive Chunker)
Embeddings (MiniLM)
FAISS Vector Creation
Retriever Chain Setup (LLM+RAG)
Chat I/P Handler
History-Aware Retrieval via LangChain
LLM (Mistral 7B)
Response returned to UI

Strategies for HIPAA Compliance and Addressing Biases:
To ensure data privacy and response reliability, the system incorporates the following measures:

- Localized LLM: All processing occurs locally to prevent data exposure to external servers.
- Access Logs: User actions are logged for traceability and audit compliance.
- FAISS Storage: Embedded medical data is stored securely in a local FAISS index.
- Bias & Hallucination Control: A low-temperature LLM with structured prompts reduces hallucinations and biased outputs.
- Automatic File Deletion: Uploaded PDFs are deleted post-session to limit data retention.

Evaluation Metrics:

1. Hybrid Evaluation Approach: The chatbot's performance can be evaluated using both human feedback and automated metrics to ensure accuracy and usability.
2. RAGAS Framework: The Retrieval-Augmented Generation Assessment Suite (RAGAS) provides quantitative metrics tailored for evaluating RAG-based systems.
    a. Context Precision: Measures how accurately the retrieved document segments align with the user's query. High precision indicates that the chatbot is retrieving the most relevant information from the lab report.
    b. Faithfulness: Assesses whether the generated response is directly grounded in the retrieved content. It helps detect hallucinations and ensures factual consistency.