**PolicyDoc QA - Generative Search System for Policy Documents**

**📌 Project Overview**

**Project Name:** PolicyDoc QA  
**Technology Stack:** Python, Streamlit, LlamaIndex, OpenAI GPT (gpt-4o-mini), Sentence Transformers  
**Goal:** To enable users to upload any policy PDF and ask natural language questions, with the system returning answers based on document content using generative AI.

**🎯 Project Goals**

* Create a seamless QA interface for policy documents.
* Leverage vector embeddings for efficient document retrieval.
* Enable contextual answering of policy-related questions using an LLM.
* Ensure modularity and reusability of the codebase.

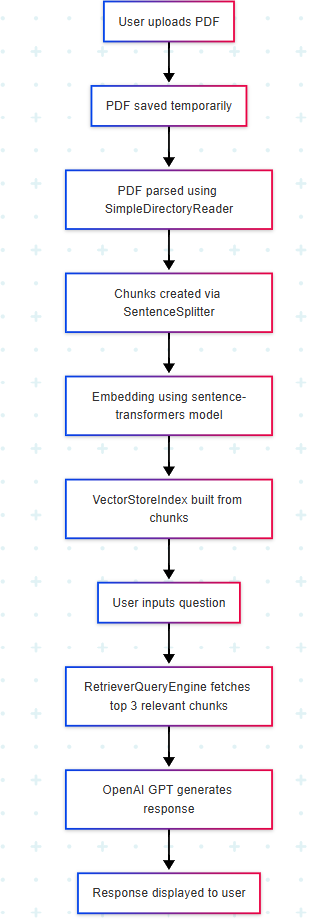
**📂 Data Sources**

* User-uploaded PDF policy documents.
* Extracted text from PDFs is processed and split into chunks using the SentenceSplitter from LlamaIndex.

**🧠 Design Choices**

1. **Frontend:**
   * Used **Streamlit** for a lightweight, interactive UI.
2. **Backend Processing:**
   * Utilized **LlamaIndex** for document ingestion, chunking, embedding, and retrieval.
   * Selected **sentence-transformers/all-MiniLM-L6-v2** for embedding as a fast and efficient model.
   * Used **OpenAI GPT (gpt-4o-mini)** for generating answers.
3. **System Architecture Components:**
   * PDF upload and temporary file handling.
   * Vector index creation from parsed document chunks.
   * RetrieverQueryEngine for similarity-based query processing.
4. **Settings Management:**
   * Used .env file for securely loading OpenAI API keys.

**➡️ Flow Chart**

****

**⚠️ Challenges Faced**

* Handling large PDF files within memory and time constraints.
* Ensuring accurate and relevant chunking of text.
* Reducing hallucination in LLM-generated answers.
* Optimizing latency between indexing and query response.