🧩 Code Architecture Overview for PDFnquiry

This document outlines the architecture of the PDFnquiry project, detailing the key components, their responsibilities, and the flow of data within the application.

**🧱 Key Components**

**1. Frontend (React)**

Located in the frontend/ directory.

**Key Roles:**

* Provides a user-friendly chat interface.
* Allows users to upload PDF files and ask questions.
* Sends requests to the backend using Axios.
* Displays answers from the AI system in real time.

**Core File:** QuestionBox.jsx

* Handles UI state (messages, input, loading).
* Sends POST requests to /upload/ and /ask/ endpoints.
* Uses useEffect to auto-scroll chat to the latest message.

**2. Backend (FastAPI)**

Located in the backend/ directory.

**Key Roles:**

* Receives and processes PDF uploads.
* Extracts and indexes document content.
* Handles question answering via AI integration.

**Core Files:**

* main.py: Defines FastAPI routes and logic.
* rag\_utils.py: Contains logic for processing PDFs and retrieving answers.

**Endpoints:**

* POST /upload/: Accepts and stores PDF files, then calls process\_pdf().
* POST /ask/: Accepts user questions, retrieves answers using get\_answer().

**3. RAG Utilities (rag\_utils.py)**

**Key Roles:**

* Parses PDF content.
* Generates vector embeddings using OpenAI APIs.
* Stores and searches document chunks using FAISS.
* Generates context-aware answers via LangChain and GPT models.

**4. Vector Store (FAISS)**

**Key Roles:**

* Efficiently stores document embeddings.
* Performs similarity search to retrieve relevant document parts.
* Supports Retrieval-Augmented Generation (RAG) for better context.

**🔄 Data Flow Overview**

1. **User Uploads PDF**:
   * Sent via React frontend to POST /upload/.
   * PDF is saved and passed to process\_pdf().
   * Text is extracted, chunked, embedded, and indexed with FAISS.
2. **User Asks a Question**:
   * Sent via React frontend to POST /ask/.
   * Question is embedded and compared with stored vectors.
   * Top matches are used as context for GPT via LangChain.
   * AI-generated answer is returned and displayed in chat.

**📦 Deployment**

* Docker containers are used to encapsulate the frontend and backend.
* docker-compose.yml coordinates multi-service deployment.

This modular structure ensures maintainability, scalability, and ease of development across both the frontend and backend.