**Working Architecture Overview**:  
  
The system consists of three main layers:

A. Frontend Layer (Streamlit)

* Purpose: User interface for file upload and question input.
* Components:
  + File uploader for CSV, Excel, or PDF.
  + Text input for natural language queries.
  + Display area for results (JSON, charts, summaries).
* Interactions:
  + Sends uploaded file and query to the backend API using HTTP POST requests.
  + Receives processed results (answers, charts, summaries) and displays them.

B. Backend Layer (FastAPI)

* Purpose: Handles incoming requests, orchestrates agents, and returns results.
* Components:
  1. FastAPI app
     + / endpoint: Health check.
     + /query endpoint: Accepts file + question.
  2. Orchestrator Agent
     + Determines which agent to use:
       - Data Intelligence Agent - for CSV/Excel.
       - Research Assistant Agent - for PDFs.
     + Routes the file and question to the correct agent.
* Interactions:
  1. Receives file and question from frontend.
  2. Saves uploaded file to uploads/ directory.
  3. Sends file to the appropriate agent.
  4. Returns agent’s response as JSON.

C. Agent Layer

1. Data Intelligence Agent

* Purpose: Handle tabular data analysis.
* Process:
  1. Reads CSV/Excel using pandas.
  2. Creates a LangChain Pandas Data Frame Agent powered by OpenAI (Chat OpenAI).
  3. Executes queries like:
     + Natural language questions about the data.
     + Plot/chart generation.
  4. Returns either:
     + Answer (string).
     + Chart (base64 encoded image).

2. Research Assistant Agent

* Purpose: Handle PDF research papers.
* Process:
  1. Reads PDF using PyMuPDF (fitz).
  2. Splits content into chunks using Recursive Character TextSplitter.
  3. Embeds chunks into vector database using OpenAI Embeddings + FAISS.
  4. Supports:
     + Summarization (first 3000 characters).
     + Keyword/Method/Result extraction.
     + General Q&A using LangChain QA chain.
  5. Returns processed results as JSON.

D. OpenAI Layer

* Purpose: Provides LLM & embeddings.
* Interactions:
  + Used by both agents:
    - Chat OpenAI for text generation (Q&A, summaries).
    - OpenAI Embeddings for PDF vectorization.
* API Key Management:
  + Stored in .env → loaded using python-dotenv.
  + Accessible via os.environ ["OPENAI\_API\_KEY"].

Data Flow Diagram:

User

Orchestrator

FastAPI Backend

File Storage

Streamlit Frontend

Research Assistant Agent

Data Intelligence Agent

Text Extraction

Pandas DataFrame

OpenAI Embeddings

OpenAI API

FAISS Vector Store

Chart Generation

FastAPI Response

OpenAI API

Results JSON

Streamlit Display

Summarises & QA