┌────────────────────┐

│ PDF / Doc Upload │

└────────┬───────────┘

│

┌─────────▼────────────┐

│ LangChain PDF Loader │

└─────────┬────────────┘

│

┌───────────▼──────────────┐

│ Text Splitter (Chunks) │

└───────────┬──────────────┘

│

┌──────────▼─────────┐

│ Generate Embeddings│ ◄───────┐

└──────────┬─────────┘ │

│ │

┌──────▼────────┐ │

│ ChromaDB │◄─────────┘

└──────┬────────┘

│

┌───────────▼─────────────┐

│ User Query (Text) │

└───────────┬─────────────┘

│

┌────────▼───────────┐

│ Search Similar Docs│

└────────┬───────────┘

│

┌─────────▼─────────────┐

│ Inject Docs + Query │

└─────────┬─────────────┘

│

┌───────▼────────────┐

│ Run LLM via Ollama │

└───────┬────────────┘

│

┌─────────▼───────────┐

│ Return Final Answer│

└─────────────────────┘

| **Step** | **Component** | **Description** |
| --- | --- | --- |
| 1 | PDF/Doc files | Input data source |
| 2 | PyPDFLoader | Load documents into LangChain format |
| 3 | RecursiveCharacterTextSplitter | Break documents into manageable chunks |
| 4 | OllamaEmbeddings | Convert text into numerical vector embeddings using a local LLM |
| 5 | Chroma or FAISS | Store vectors for semantic search |
| 6-13 | RetrievalQA or ConversationalRetrievalChain | LangChain chain to implement RAG |
| 11 | Ollama + LLaMA | Local LLM that answers using prompt + relevant chunks |
| 14 | ConversationBufferMemory | Stores chat history for multi-turn interactions |

**🔁 Overall System Flow (End-to-End)**

**⚙️ SETUP PHASE (One-time)**

markdown

Copy code

1. Document Collection

↓

2. Document Loading

↓

3. Text Splitting / Chunking

↓

4. Embedding Generation

↓

5. Vector Store Insertion (ChromaDB)

**🔍 QUERY PHASE (Runtime: User Input → LLM Answer)**

pgsql

Copy code

6. User Query Input (Text)

↓

7. Convert Query → Embedding

↓

8. Vector Search in ChromaDB (Similarity Search)

↓

9. Retrieve Top Relevant Chunks

↓

10. Inject Chunks + Query into LLM Prompt (RAG)

↓

11. Local LLM Inference via Ollama (LLaMA Model)

↓

12. Answer Generation with Contextual Info

↓

13. Output Answer to User

**🧠 CONVERSATION MEMORY FLOW (Optional, for multi-turn chat)**

markdown

Copy code

14. Store User Query + LLM Answer in LangChain Memory

↳ Used in next turns for context retention

**🧪 Example Query Workflow (Actual Flow in App)**

1. **User Asks**: "What are the key benefits of 5G?"
2. **LangChain**:
   * Embeds the query
   * Searches ChromaDB
   * Gets 3 most relevant document chunks
3. **Builds Prompt**:

Context:

- Doc1 snippet...

- Doc2 snippet...

- Doc3 snippet...

Question: What are the key benefits of 5G?

1. **Ollama (LLaMA)**:
   * Generates a natural language answer using the above prompt
2. **LangChain Memory** (optional):
   * Saves the question and answer for future reference
3. **Output**: Shows the answer in terminal or Gradio UI