Project PRD: PaperPal – PDF Chatbot for Research Papers

# 1. Overview

PaperPal is an AI-powered chatbot that enables users to upload academic research papers (PDFs) and ask natural language questions. It retrieves relevant context from the document and provides grounded, citation-linked answers using LLMs via Retrieval-Augmented Generation (RAG).

# 2. Goals

- Simplify reading dense academic PDFs

- Enable question-answering with direct source reference

- Support summarization, highlighting, and citation-aware chat

# 3. Target Users

- Students  
- Researchers  
- Journalists  
- Professionals analyzing dense PDF reports

# 4. Core Features

* 🗂 Upload PDF: User uploads academic or technical PDFs
* 🧠 Ask Questions: LLM answers questions using PDF context
* 🔍 Chunk + Vector Search: Splits document and stores in vector DB
* 📎 Source Citations: Answers link back to exact PDF chunks
* ✍️ Summarization: Generate bullet-point summaries
* 💾 Save & Export Chat: Export Q&A history
* 🧪 Multiple Models: Switch between LLM backends (Ollama/OpenRouter)

# 5. Tech Stack

UI: Streamlit

LLM Interface: LangChain

Vector DB: Chroma / FAISS

Embeddings: OpenAIEmbeddings, SentenceTransformers, ollama:embed

PDF Parser: PyMuPDF / pdfplumber

Backend Model: OpenRouter / Ollama / OpenAI

# 6. Non-Goals

- No OCR support for scanned PDFs  
- No browsing of web-based PDFs  
- Multi-file summarization is future scope

# 7. Success Metrics

* Answer accuracy based on chunk relevance
* Latency < 5s per query
* >80% users find helpful responses
* Full summary under 300 words for most papers

# 8. Timeline

* Week 1: PDF upload + chunking + embedding to Chroma
* Week 2: LangChain Q&A + citation chaining
* Week 3: Streamlit UI + model selection + testing
* Week 4: Polish, add export/save features, deploy

# 9. Folder Structure

pdf-chatbot/  
├── app.py # Streamlit entry point  
├── requirements.txt  
├── README.md  
├── .env # API keys (OpenAI / OpenRouter)  
├── data/  
│ └── sample.pdf # Example file  
├── modules/  
│ ├── pdf\_loader.py # PDF reading and chunking  
│ ├── vectorstore.py # Embedding and Chroma index  
│ ├── rag\_chain.py # LangChain RAG pipeline  
│ └── summarizer.py # Optional summarization  
├── assets/  
│ └── logo.png # Optional branding  
└── utils/  
 └── prompts.py # Reusable prompt templates

# 10. Prompt Templates

📌 Question-Answering Prompt:  
You are a helpful assistant. Use the following context to answer the user's question. Cite page and chunk when relevant.  
  
Context:  
{context}  
  
Question:  
{question}  
  
Helpful Answer:

📌 Summarization Prompt:  
Summarize the following academic paper into 3 concise bullet points.  
  
Text:  
{chunked\_text}  
  
Summary: