Paves Technologies – AI Chatbot Documentation

1. Project Overview

This AI-powered chatbot allows users to interact with company documents using a conversational interface built with Streamlit. It uses Retrieval-Augmented Generation (RAG) to fetch relevant chunks from uploaded PDF documents and generate meaningful responses via a language model (LLM).

2. Technology Stack

- Frontend/UI: Streamlit

- Backend: Python (RAG architecture)

- PDF Parsing: PyPDF2

- Vector Database: In-memory (can be replaced with FAISS, Chroma, etc.)

- LLM Client: GROQ API (configurable)

- Document Chunking: Custom logic based on sentence boundaries and overlapping context

3. Folder Structure

- app.py → Main Streamlit interface

- document_processor.py → Extracts and chunks text from PDFs

- rag_system.py → Central engine connecting documents, vectors, and LLM

- vector_store.py → Stores vector embeddings and provides similarity search

- llm_client.py → Interface to connect with GROQ or another LLM

- utils.py → Helper functions (logging, formatting, etc.)

- requirements.txt → Python dependencies

4. How It Works

1. Users upload PDFs via the sidebar.

2. `DocumentProcessor` extracts and chunks the content.

3. `VectorStore` embeds the chunks into vector form.

4. When a user asks a question, `RAGSystem` retrieves top relevant chunks.

5. These chunks are passed to the `LLMClient` to generate an answer.

6. The answer, along with sources, is shown in the chat window.

5. Features

- Upload and process multiple PDFs

- Store session history using Streamlit session state

- Display source citations for transparency

- Clear conversation and system status tracking

- Custom CSS styling for branding

6. How to Run

1. Install dependencies:

pip install -r requirements.txt

2. Launch app:

streamlit run app.py

3. Upload PDFs and chat with them.

 Note: Ensure a valid GROQ API key is present in `app.py` or passed via environment variable `GROQ_API_KEY`.

7. Security Note

For production use, move API keys to a `.env` file or a secure secrets manager.

Never hard-code sensitive keys in source code (as done temporarily for local testing).

8. Future Enhancements

- Replace in-memory store with FAISS or ChromaDB for scalability

- Enable multi-user document handling and access control

- Add OpenAI or other LLM providers as backend switch options

- Deploy via Docker or Streamlit Cloud