**Document Research & Theme Identification Chatbot**

**🔍 Project Overview**

An AI-powered chatbot that performs research across a large set of documents (PDFs, scanned images), extracts answers to user queries with citations, and identifies common themes across documents.

**🎯 Objective**

* Upload and process 75+ documents (PDF, scanned images).
* Extract text using OCR.
* Store semantic embeddings for retrieval.
* Answer queries with document-level and fine-grained citations.
* Identify and synthesize multiple themes across documents.

**🧠 Tech Stack**

* **Language:** Python
* **Framework:** FastAPI (Backend), Streamlit (Frontend MVP)
* **LLM:** OpenAI GPT, Groq (LLaMA2/3), Gemini
* **Vector Store:** ChromaDB
* **OCR:** PaddleOCR / Tesseract
* **Database:** SQLite / PostgreSQL

**📁 Folder Structure**

chatbot\_theme\_identifier/

├── backend/

│ ├── app/

│ │ ├── api/

│ │ ├── core/

│ │ ├── models/

│ │ ├── services/

│ │ ├── main.py

│ │ └── config.py

│ ├── data/

│ ├── Dockerfile

│ └── requirements.txt

├── docs/

├── tests/

├── demo/

└── README.md

**🧱 System Architecture**

Upload PDF/Images

↓

OCR (PaddleOCR) → Text Extraction

↓

Chunk & Embed Text (LangChain + OpenAI Embeddings)

↓

Store in ChromaDB Vector Store

↓

Query Processing (LangChain Retriever + LLM)

↓

Cited Answer with Metadata (DocID, Page, Para)

↓

Theme Identification (KMeans / BERTopic)

↓

Synthesized Summary + Theme-Citation Mapping

**⚙️ Key Functionalities**

**1. Document Upload & OCR**

* Supports PDF and scanned images.
* Converts pages to text using OCR (PaddleOCR).

**2. Semantic Chunking & Embedding**

* Uses RecursiveCharacterTextSplitter to chunk text.
* Embeds with OpenAI / HuggingFace and stores in ChromaDB.

**3. Query Handling**

* Vector search retrieves top-k chunks.
* LLM generates answer with citations.

**4. Theme Discovery**

* Uses clustering (KMeans) or BERTopic to find recurring ideas.
* Maps themes to document IDs and citations.

**5. Enhanced UI Features (Optional)**

* Streamlit UI for upload/querying.
* Filter by document metadata.
* Toggle docs for query scope.

**📊 Output Format**

**Individual Responses Table**

|  |  |  |
| --- | --- | --- |
| **Document ID** | **Extracted Answer** | **Citation** |
| DOC001 | SEBI violation... | Page 4, Para 2 |
| DOC002 | Late disclosure | Page 2, Para 1 |

**Synthesized Themes**

**Theme 1 – Regulatory Non-Compliance**

DOC001, DOC002 highlight SEBI/LODR violations.

**Theme 2 – Penalty Justification**

DOC001 explicitly justifies penalties under SEBI Act.

**🚀 Deployment**

* **Backend:** Render, Railway
* **Frontend:** Streamlit Cloud / Vercel
* **Models:** Use Groq (LLAMA), Gemini, or GPT (free tiers available)

**🧪 Testing**

* Unit tests for OCR, embedding, retrieval modules.
* Test cases for query → response validity and citation correctness.

**📜 References**

* LangChain documentation
* OpenAI API
* PaddleOCR GitHub