📄 **Uploaded PDF** 📂

🔹 **User uploads a PDF file (Digital or Scanned).**
🔹 **Formats Supported:** Text-based PDFs (**Digital**) & Image-based PDFs (**Scanned**).

⬇️

🟦 **Text Extraction** 📜🔍 **(pdf_text_extractor.py)**

🔹 **Extracts text using two methods based on PDF type:**
✅ **Digital PDFs** → Extracts text using **PyMuPDF** (fast & accurate).
🖼️ **Scanned PDFs** →
  🔄 **Converts PDF to images** (pdf2image).
  🧠 **Performs OCR (Optical Character Recognition)** with **Tesseract**.

📌 **Output:** Extracted text saved as .txt for further processing.

⬇️

🟩 **Text Cleaning** 🧹📝 **(text_cleaning.py)**

🔹 **Removes unnecessary elements to enhance data quality:**
🖊️ **Text Normalization** → Converts text to **lowercase** (unicodedata).
🚮 **Noise Removal** → Eliminates **extra spaces, special characters** (regex).
🔍 **Stopword Removal** → Uses **spaCy** to remove unnecessary words.

📌 **Output:** Cleaned, structured text ready for feature extraction.

⬇️

🟧 **Feature Extraction** 📊🔍 **(pdf_feature_extractor.py)**

🔹 **Extracts structured information using NLP & pattern matching:**
🏛️ **Named Entity Recognition (NER)** → Detects **financial, legal, crypto-related terms** (spaCy).
📧 **Extracts Emails & Phone Numbers** → Uses **email_validator** & **phonenumbers**.
⚠️ **Detects Scam Patterns** → **Fuzzy Matching** (thefuzz) finds suspicious phrases.
📊 **Sentiment Analysis** → Evaluates text sentiment using **VADER Sentiment**.

📌 **Output:** JSON file containing **extracted entities, risks, and sentiment scores**.

⬇️

🟧 **Chunking & AI Embedding** 🧠 🔢 **(text_chunker.py)**

🔹 **Splits large text into AI-friendly chunks & embeds them into vectors:**
✂️ **Removes Irrelevant Sections** (e.g., Table of Contents, Headers).
📄 **Breaks text into logical chunks** using **LangChain Recursive Splitter**.
🧠 **Converts text into AI Embeddings** using **Ollama (Nomic-Embed-Text)**.

📌 **Output:**
🗂️ **JSON File** → Contains structured text chunks & embeddings.

⬇️

💾 **Stored in FAISS Vector Database** 🔎 📂 **(vector_database.py)**

🔹 **Stores AI embeddings for fast retrieval using FAISS (Facebook AI Similarity Search):**
🚀 **FAISS Vector Storage** → Enables similarity-based text search.
🔍 **Optimized for AI-powered retrieval** → Enables **semantic search** & **instant results**.

📌 **Output:**
📁 **FAISS Database** → Stores **vectorized document embeddings**.
📊 **Metadata JSON** → Maps document chunks to embeddings.

⬇️