Multi-Modal Rag(Text + Image) :

📄 Multimodal RAG System with Cohere, ChromaDB, and Gemini via LangChain (LCEL)

🔄 Overview

This pipeline processes a PDF by extracting text and images, generating embeddings using Cohere, storing them in ChromaDB, and answering user questions using Gemini via a LangChain LCEL prompt.

📌 Step-by-Step Breakdown

🧾 1. User Question Input

- The user provides a **natural language query**.
- This query will be transformed into an embedding to match against the stored document chunks and images.

📄 2. PDF Processing

2.1 Text Extraction

- **Tool:** `fitz` (from `PyMuPDF`)
- Each page's text is extracted and added to a list of `text_chunks`.

2.2 Image Extraction

- **Tool:** `fitz` for extracting images
- **Tool:** `Pillow (PIL)` for converting, resizing images
- Each image is converted to RGB, resized if too large (based on pixel count), and encoded as Base64.

🔍 3. Embedding Generation (Cohere)

3.1 Text Embedding

- **Tool:** `cohere` Python SDK
- **Model:** `embed-v4.0`
- **Input type:** `search_document`
- The extracted text chunks are embedded as float vectors.

3.2 Image Embedding

- **Input:** Base64-encoded PNG images
- **Input type:** `search_document`
- The image is passed as part of a content list and embedded similarly.

🧠 4. Storage in Vector Database (ChromaDB)

- **Tool:** `chromadb.PersistentClient`
- **Storage path:** `"/content/chroma_db"`
- All embedded text and image vectors are added to a collection called `multimodal_embeddings`.

❓ 5. User Query Embedding (Cohere)

- The user query is embedded with:
  - **Model:** `embed-v4.0`
  - **Input type:** `search_query`

🧠 6. Retrieval from ChromaDB

- The top `k` similar text chunks and images are retrieved based on **cosine similarity** between query embedding and stored vectors.
- **Dependency:** `chromadb`, `sklearn.metrics.pairwise.cosine_similarity`

🪄 7. Prompt Formatting via LangChain (LCEL)

- **Tool:** `langchain`, `langchain_core.prompts.ChatPromptTemplate`
- Prompt includes:

- User's question
- Retrieved top `k` text chunks
- Retrieved top `k` images (PIL objects)

- LCEL chains the context and question into a structured message for Gemini.

🤖 8. Response Generation via Gemini LLM

- **Tool:** `google-generativeai` Python SDK
- **Model:** `gemini-2.5-flash`
- Processes the structured prompt and generates a contextual response.
- Handles multimodal content (text + image reasoning).

✅ 9. Final Output

- The generated response is displayed.
- The top relevant image results are also shown using `IPython.display`.

📦 Libraries Used

Purpose	Dependency
Text/Image extraction	`fitz (PyMuPDF)`, `Pillow`
Image encoding/resizing	`base64`, `io`, `Pillow`
Embedding generation	`cohere`
Vector DB storage/query	`chromadb`, `uuid`, `tqdm`
Similarity matching	`sklearn.metrics.pairwise`
Prompt creation	`langchain`, `ChatPromptTemplate`
LLM answering	`google-generativeai`
Output display	`IPython.display`