my\_rag\_app/

├── app.py # 主頁

├── pages/

│ ├── Build\_Knowledge\_Base.py # Page 1: 建立向量資料庫

│ └── Answering\_Interface.py # Page 2: 問答功能

├── vectorstores/ #資料庫內容

├── Upload\_file.py

├── RAG\_Embedding.py

├── RAG\_LLM\_Generator.py

├── reranking.py

├── full\_file.py

├── UI\_model\_select.py

├── utils.py

├── config.py

└── requirements.txt

|  |  |
| --- | --- |
| 階段 | 建議做法 |
| 🧪 開發初期 | 原始檔存在 ./files/、 向量資料存在本機 FAISS、 metadata 用 JSON 存 |
| 🧑‍💼 部署上線 | 原始檔 → S3 或 DB /  向量 → Qdrant、Weaviate /  metadata → DB (SQLite/Postgres) |
| 👥 多人共用 | 向量庫與原始檔應該能透過 API 存取， 不應只存在 session 或 local folder |

**使用流程概要**

1. 使用者進入「知識庫建立」頁面，選擇或建立資料庫
2. 上傳檔案，系統自動轉成向量並建立資料庫
3. 進入「問答介面」，選擇已有資料庫並設定模型參數
4. 輸入問題，系統檢索相關向量並呼叫 LLM 生成回答
5. 查看回答及來源段落，確認結果與依據

**一、新增模組的說明**

**1. 主程式（app.py）**

* 設定 Streamlit 頁面配置（標題、圖示、版面寬度）
* 展示系統簡介與左側選單說明
* 初始化全局的模型設定（包括圖像模型與語言模型參數）
* 引導使用者從左側選單切換不同功能頁面

**2. 知識庫建立與管理介面（Knowledge\_Interface.py）**

* 以寬版兩欄設計：左欄為現有資料庫管理，右欄為新增資料庫
* **左欄功能**：
  + 列出本地已有向量資料庫（資料夾）清單，供使用者選擇
  + 顯示所選資料庫的詳細資訊（建立時間、包含檔案）
  + 允許刪除單一檔案並重建向量資料庫，保持資料一致性
  + 提供刪除整個資料庫功能，並以勾選確認防止誤刪
  + 可新增檔案至現有資料庫並重建向量資料庫
* **右欄功能**：
  + 提供圖像 Embedding 模型的選擇介面
  + 支援上傳多種格式檔案（PDF、Word、純文字、圖片）
  + 使用者輸入資料庫名稱，建立新的向量資料庫
  + 新資料庫建立成功後顯示段落數量並更新狀態

**3. 問答介面（Answering\_Interface.py）**

* 設定問答頁面基本資訊與標題
* 初始化模型參數、向量資料庫與文件列表
* 掃描本地向量資料庫清單，讓使用者選擇欲使用的資料庫
* 載入向量資料庫與原始文件資料，用於檢索與回答
* 顯示模型設定選單（如 LLM 模型、temperature、top\_p 等）
* 使用者輸入問題並啟動問答流程
* 根據設定的檢索方法（MMR、Reranking、Custom RAG、Basic）執行檢索
* 取得相關段落後，呼叫 LLM 生成答案
* 顯示生成的回答與 LLM 思考過程
* 展示匹配到的原始文件段落及其來源，增加回答透明度

### 4. 向量資料庫管理模組（RAG\_Embedding.py）

負責整體向量資料庫的建立、維護與查詢資源的封裝，包含以下核心功能：

**資料夾與檔案管理**

* 程式啟動時自動建立 VECTOR\_DB\_DIR 根目錄，確保所有資料庫皆儲存於統一路徑下。
* 上傳檔案會儲存於 source\_files/ 子資料夾中，維持資料來源完整可追蹤。

**Metadata 與文件內容管理**

* save\_metadata / load\_metadata：  
  負責 info.json 存取，紀錄資料庫基本描述（包含檔案清單、切片大小、使用模型等）。
* save\_documents / load\_documents：  
  處理 metadata.json 的儲存與還原，紀錄所有 Document 的 page\_content 及 metadata，方便問答時使用原始段落。

**向量庫建立與重建**

* 自定義 OllamaEmbeddings 類別：
* build\_FAISS：
  + 將原始文件切成固定大小的 chunk，並標記 chunk\_id。
  + 使用 Ollama 產生嵌入後建立 FAISS 向量庫。
* save\_vectorstore / load\_vectorstore：
  + 支援儲存與載入 FAISS 本地索引，結合 index.faiss 與 index.pkl。
  + 載入時啟用 allow\_dangerous\_deserialization=True 以支援本地 pickle 文件。

**新增向量資料庫（create\_new\_vector\_db）**

* 使用者可指定資料庫名稱並上傳檔案，系統會：
  1. 建立專屬資料夾並儲存上傳檔案。
  2. 使用 process\_uploaded\_files 處理文件轉換為 Document。
  3. 對文件進行分割與嵌入，建立 FAISS 向量庫。
  4. 寫入 info.json 與 metadata.json，保存所有段落與描述資訊。

**重建向量庫（rebuild\_vector\_db）**

* 可針對既有資料庫新增檔案並整合至向量庫：
  + 模擬 BytesIO 處理新增檔案，並加入原有 Document 列表。
  + 重新建立完整向量資料庫並更新 metadata。

**刪除資料或整庫**

* delete\_files\_from\_db：
  + 支援多檔刪除，會同步移除 source\_files/ 中對應實體檔案。
  + 重新過濾剩餘段落後重建向量庫，保持 info.json 與 metadata.json 同步更新。
* delete\_vector\_db：
  + 移除整個向量資料夾，包含所有原始檔、向量檔、metadata 檔案。

**二、核心功能與特色說明**

**向量資料庫管理**

* 支援多種檔案格式上傳並轉成向量化表徵
* 動態增刪檔案並重建向量資料庫，確保最新資料反映在檢索結果
* 詳細紀錄 metadata 與原始文件內容，支援完整還原與問答依據追蹤
* 向量庫採用 FAISS，結合 Ollama 提供的 Embedding 模型，具備高效檢索能力

**多模型參數調整**

* LLM 生成參數（temperature、top\_p、top\_k）可透過 UI 自訂
* 提供不同檢索方法選擇，符合多種應用需求
* 具圖片 Embedding 模型選擇，適用圖像資料庫建置

**問答流程優化**

* 融合不同檢索策略：MMR 提升多樣性，Reranking 精選相關段落
* Custom RAG 支援長文件結合多段落資訊回答
* 顯示回答來源段落，提升使用者信任與可追溯性