AI-Driven Caregiver Recommendation App (Proof of Concept)

1. Introduction

This project demonstrates a simple web application that:

- Lets a user describe an elderly person's health condition via a form.
- Finds and recommends suitable caregivers based on that description.
- Explains why each caregiver is a good match.

2. How It Works (High-Level)

a. Frontend (Django):

- A form asks for the elderly patient's condition (= query).
- When submitted, triggers the recommendation process.

b. Retrieval (RAG):

- We prepared a small 'caregivers.xlsx' file with sample profiles.
- Loaded profiles into memory and converted them into 'documents'.
- Converted text to vector embeddings (semantic fingerprints).
- Stored embeddings in a FAISS index so we can search by similarity.

c. Generation (LangChain + LLM):

- We use a local Hugging Face 'text-generation' pipeline (GPT-2 placeholder).
- A 'prompt template' tells the model:
  * Only return bullet points.
  * Show 'Name' and 'Reason' for each match.
- The model uses the top-3 retrieved profiles as context to generate clear, concise recommendations.

3. Key Code Components

- utils.load_caregivers():
  Reads 'caregivers.xlsx', creates Document objects for each profile.

- utils.build_vectorstore():
  Embeds profiles and builds an in-memory FAISS index.

- retriever.get_recommendations(query):
  1) Finds the top matching profiles from FAISS.
  2) Packs their text into a single 'context' string.
  3) Calls the LLM chain to generate bullet-format advice.

- views.recommend():
  Handles the web request, calls get_recommendations,
  and passes the results to the HTML template.

4. User Experience

- The user types in a description like:
  '75-year-old with chronic kidney disease and mobility issues'.
- The page shows:
  * A bullet list of recommended caregivers with reasons.
  * A list of the source profiles used.

5. Setup & Environment

- Python 3.11.4 (managed by pyenv), in a virtual environment (venv).
- Main libraries installed:
  Django, pandas, reportlab,
  langchain, langchain_huggingface, langchain_community,
  transformers, faiss-cpu

6. Next Steps

- Swap GPT-2 with a more powerful local model (e.g. LLaMA or Mistral).
- Persist FAISS index to disk for faster startup.
- Add user authentication and save recommendation history.
- Implement a feedback loop (user rates matches to improve accuracy).
- Enhance the UI for mobile and desktop views.