Architecture Overview / Pipeline Description

The system is built using a **modular pipeline** where each stage handles a specific task, from raw data to final recommendations. Here's the breakdown of how everything fits together:

1. Data Layer (Local CSVs + BigQuery)

- Raw data starts as .csv files: destinations, members, trips, and weather.
- Cleaned and preprocessed in Jupyter/Colab notebooks.
- Loaded into Google BigQuery for efficient filtering using SQL when needed (like budget, travel style, etc.).

Why:
BigQuery handles structured filtering well, especially when we want to sort destinations by cost or rating without loading everything into memory.

2. Embedding Layer (Hugging Face + FAISS)

- Text fields (like user profiles or destination descriptions) are converted into **semantic vectors** using Hugging Face's all-MiniLM-L6-v2.
- These vectors are saved locally using **FAISS**, which supports fast similarity search.

We built **three indexes**:

- Destination embeddings
- Trip history embeddings
- Weather-aware destination embeddings

Why:
This allows for RAG-style recommendations - not just filters, but smart, meaning-based suggestions.

3. Retrieval-Augmented Generation (RAG)

- A user query like "affordable cultural destination in January" is also embedded.
- FAISS returns closest matches from the relevant dataset.
- Those results are displayed with clear reasons: cost, season rating, tags, etc.

Why:
This lets us give meaningful, ranked recommendations - beyond filters, using semantic similarity.

4. Frontend (Streamlit App)

- Streamlit collects user input (age, budget, travel style).
- Calls either the BigQuery SQL filters **or** the FAISS RAG pipelines.

- Outputs travel suggestions, reasons why they were picked, and lets the user export results as CSV or PDF.

Why:
Simple, interactive interface for demoing everything in one place without needing a full web app.

Optional: LLM Assistant (GPT/Gemini)

- Users can upload CSVs and ask free-text questions.
- A small RAG loop extracts relevant chunks and sends them to GPT (or Gemini) to generate human-readable answers.