**Weather-Activity Assistant**

**1. Overview**

The Weather-Activity Assistant is a Streamlit-powered application designed to provide users with personalized recommendations for activities and clothing based on current weather conditions. It intelligently detects the user's location (either from the query or via IP address), fetches real-time weather data, and then leverages a Large Language Model (LLM) and a RAG (Retrieval Augmented Generation) system to generate contextually relevant suggestions. The core idea is to enrich user queries with real-time weather information and retrieve relevant information from a curated knowledge base (a PDF document in this case) to offer comprehensive advice.

**2. Architecture**

The application follows a modular architecture, with distinct components responsible for specific tasks. The flow is primarily sequential, where the output of one component serves as the input for the next.

User Query (Streamlit UI)

↓

1. Country Detection (LLM)

↓ (If country detected, use it for weather. If not, proceed to IP detection)

2. Location Determination (IP Address or Query-derived)

↓

3. Weather Data Fetching (WeatherAPI.com)

↓

4. Query Regeneration (LLM + Weather Data + Location)

↓

5. RAG System (PDF Loader, Text Splitter, Embeddings, FAISS Vector Store, Retriever)

↓

6. Final Response Generation (LLM + Context from RAG + Rewritten Query)

↓

User Display (Streamlit UI)

**Key Components:**

* Streamlit UI: Provides the interactive web interface for user input and displaying results.
* Country Detection Module: Utilizes an LLM to identify a country from the user's natural language query.
* Location Service: Determines the user's approximate location based on their IP address as a fallback if no location is specified in the query.
* Weather API Integration: Fetches real-time weather data for a given location.
* Query Regeneration Module: An LLM-powered component that takes the original user query, weather details, and location to create an enriched, more informative query.
* Retrieval Augmented Generation (RAG) System:

**Document Loader**: Loads information from a PDF document.

**Text Splitter**: Breaks down the document into smaller, manageable chunks.

**Embedding Model**: Converts text chunks into numerical vector representations.

**Vector Store (FAISS**):Stores and enables efficient similarity search over the

embedded document chunks.

**Retriever**: Fetches relevant document chunks based on a query.

* Main LLM (Mixtral-8x7B-Instruct-v0.1): The central intelligence, used for country detection, query regeneration, and generating the final user-facing response.

**3. Setup and Installation**

To run this application locally, follow these steps:

Prerequisites:

* Python 3.8+
* `pip` (Python package installer)

**3.1. API Keys:**

You will need two API keys:

1. Together AI API Key: For accessing the Mistral LLM.

2. WeatherAPI.com API Key: For fetching weather data.

**3.2. Project Structure:**

Ensure your project directory is structured as follows:

weather\_Assistant/

    ├── requirements.txt

├── Weather Activity Clothing Database.pdf

   ├── README.md

 ├── run.py

 └── app/

     ├── config.py

     ├── core.py

     ├── prompts.py

     └── utils.py

**3.3. Installation Steps:**

1. Clone the repository or download it

2. Create a virtual environment

3. Activate the virtual environment

4. Install the required Python packages:

```bash

pip install -r requirements.txt

```

5. Run the Streamlit application:

```bash

streamlit run run.py

```

**4. Design Decisions**

**LLM Choice (Mistral-8x7B-Instruct-v0.1 via Together AI):**

This model is known for its strong performance on instruction-following tasks and is a good balance between capability and computational cost compared to larger models. Together AI provides an accessible API for this model.

**Location Detection Strategy (Query-first, then IP fallback):**

Prioritizing location from the user's query (`detect\_country\_llm`) provides the most precise and user-intended context. If the query is generic ("What should I wear today?"), falling back to IP-based location (`get\_location`) offers a reasonable approximation, enhancing usability without requiring explicit location input.

**Weather API (WeatherAPI.com):**

Provides free access to current weather data, including details like temperature, humidity, wind, and cloud cover, which are crucial for detailed recommendations.

**RAG System (LangChain, HuggingFace Embeddings, FAISS):**

RAG is essential for grounding the LLM's responses in specific, curated knowledge from the PDF. Without it, the LLM might hallucinate or provide generic advice.

* LangChain: Provides a robust framework for building LLM applications, abstracting away much of the complexity of chaining components.
* PyPDFLoader: Direct and simple way to ingest PDF content.
* RecursiveCharacterTextSplitter: Effective for splitting documents while maintaining contextual coherence by allowing overlap.
* sentence-transformers/all-MiniLM-L6-v2: A well-regarded and efficient embedding model suitable for semantic search.
* FAISS: An excellent choice for local vector storage due to its speed and efficiency for similarity search.

**Query Regeneration Step:**

Directly feeding raw weather data into the final prompt can sometimes be less effective. The query regeneration step allows the LLM to synthesize the original query with the weather details into a more coherent and semantically rich input for the final RAG chain. This helps the RAG system retrieve more relevant context and the final LLM generate a more natural-sounding response.

**5. Usage**

1. Start the Application:

Ensure your virtual environment is activated.

Run streamlit run run.py in your terminal.

2. Enter Your Query:

In the "Enter your query:" text box, type your request.

Examples:

"What should I wear in Paris today?"

"Activities to do in Cairo during cold weather."

"Is it rainy in London right now?"

"What should I wear today?" (If no location is specified, the app will try to detect your location via IP).

3. Click "Submit":

4. View Results:

Location:The app will first confirm the detected location (from your query or IP).

Weather Details: Displays the current weather conditions for the determined location.

Recommendations: Provides a summary of the weather, suggested activities, and appropriate clothing.

.