**Weather Activity & Clothing Assistant**

**By Youssef Omar**

1. **Introduction:**

The Weather Activity & Clothing Assistant is an AI-powered tool that provides users with smart recommendations for outdoor clothing and activities based on real-time weather and knowledge extracted from a detailed country-specific PDF guide. It combines real-time weather APIs, semantic PDF search, and a powerful language model (LLM) to generate human-like responses.

1. **Project Architecture:**

[User Input] ⭢ [Streamlit UI] ⭢ [Weather API Request (OpenWeatherMap)]

⭣

[LLM (Cohere Command-R)] ⭠ [PDF Retrieval via FAISS + MiniLM]

⭣

[Answer displayed to user]

1. **Project Architecture:**

|  |  |
| --- | --- |
| Layer | Tool / Library |
| Frontend / UI | Streamlit |
| Weather API | OpenWeatherMap |
| Embedding Model | SentenceTransformers (MiniLM) |
| Semantic Search | FAISS |
| LLM for Generation | Cohere (Command R ) |
| PDF Parsing | PyMuPDF(fitz) |
| API Config | Streamlit Secrets |

1. **Dataset / PDF Description:**

The assistant uses a provided PDF document containing information on:

* Weather types: sunny, rainy, snowy, and windy
* Countries: Egypt, Japan, Brazil, India, Canada, Australia, UK, USA (California), Russia (Sochi), and South Africa (Cape Town)

For each, it includes:

Appropriate outdoor activities

Recommended clothing items

This document was chunked by the country and weather section, then embedded into a vector database (FAISS) for semantic retrieval.

1. **Functional Workflow:**

* **User Inputs a City and a Question**

Example: "Japan" and "What should I wear in snowy weather?"

* **Real-Time Weather Retrieved**

From OpenWeatherMap API using the location.

* **Semantic Search Over PDF Chunks**

The user question is embedded using MiniLM

FAISS retrieves the most relevant chunks.

* **LLM Generates Final Response**

Cohere’s Command R model combines:

Weather data

Retrieved PDF info.

The user question.

* **Answer Displayed via Streamlit**

Human-readable, natural language guidance is shown.

1. **Key Design Decisions:**

| **Decision** | **Reason** |
| --- | --- |
| **Streamlit** | **Recommended, Fast, clean interface without frontend development** |
| **FAISS (Local)** | **Efficient, offline-friendly semantic search** |
| **MiniLM Embedding** | **Lightweight, fast, accurate enough for PDF content** |
| **Cohere LLM (Command-R+)** | **Free, optimized for RAG, strong language understanding** |
| **RAG Instead of Fine-Tuning** | **Flexible, updatable knowledge base (PDF can be replaced)** |

1. **Challenges and Solutions:**

* **Chunking the PDF Accurately**

**Challenge:** At first, the chunking logic using regular expressions wasn’t capturing country–weather sections cleanly.

**Solution:** After several trials, I refined the regex and metadata extraction to map each chunk accurately to both country and weather type. This was critical for retrieval relevance.

* **API Key Handling and Security**

**Challenge:** At first, the chunking logic using regular expressions wasn’t capturing country–weather sections cleanly.

**Solution:** I switched to using st.secrets both locally (via .streamlit/secrets.toml) and on Streamlit Cloud to securely handle secrets without risking leaks.

* **Limited OpenAI Quota for LLM**

**Challenge:** During early development, I used OpenAI’s text-davinci-003, but hit quota limitations.

**Solution:** I migrated to Cohere’s Command R, which is optimized for retrieval-augmented tasks and offers a generous free tier solving both the cost and performance issues.

1. **Screenshots:**

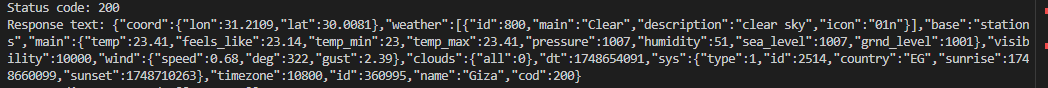
 **Screenshot of the running Streamlit app**

A screenshot of a computer

AI-generated content may be incorrect.

* **Example Q&A result**A screenshot of a computer

  AI-generated content may be incorrect.
* **Open Weather response**

****

1. **Conclusion:**

This assistant successfully combines real-time weather data, semantic document retrieval, and modern LLMs to answer user questions about clothing and outdoor activities across different countries. It demonstrates a practical use case of **Retrieval-Augmented Generation (RAG)** and is structured to be easily deployed and extended.