📄 Implementation Report

Travel Assistant AI using Gemini API

Github Link : https://github.com/ChandruKavi-Dev/Agentic_AI_Workshop/blob/8e2484c2498e2bd45da9713f539a301ec2d40213/Day%205/Travel%20Assistant.zip

---

🔍 Objective

To build an intelligent **AI-powered Travel Assistant** that helps users plan trips by:

- Providing the **current weather** for a chosen city.
- Listing the **top tourist attractions** in that city.
- Returning a **friendly summary** using Google's **Gemini Pro model**.

⚙️ Technologies Used

Component	Tool/Library
Language Model	Gemini Pro via google.generativeai
Weather API	WeatherAPI.com
Search Engine	DuckDuckGo via duckduckgo-search
Platform	Google Colab (Python-based notebook)
HTTP Requests	requests module

🧠 Workflow Overview

User Input (City)

|

|-------> WeatherAPI.com -----> Weather Info

|

|-------> DuckDuckGo Search ---> Top 5 Attractions

|

+-------> Gemini LLM (Prompt) --> Combined Summary

🛠️ Step-by-Step Implementation

1. Environment Setup

- Installed the required libraries:
- !pip install google-generativeai duckduckgo-search requests

2. Gemini API Setup

- Configured the Gemini LLM:
- import google.generativeai as genai
- genai.configure(api_key="YOUR_GEMINI_API_KEY")
- The Gemini model used: "gemini-pro" for text generation.

3. Weather Information

- Used requests to call WeatherAPI.com.
- Constructed a query URL using the user's input.
- Parsed the JSON response to extract:
    - Temperature (in Celsius)
    - Condition (e.g., Sunny, Cloudy)

url = f"http://api.weatherapi.com/v1/current.json?key={KEY}&q={city}"

response = requests.get(url)

condition = data["current"]["condition"]["text"]

temp = data["current"]["temp_c"]

4. Top Attractions

- Used duckduckgo_search.DDGS() to perform a search.
- Queried "Top tourist attractions in {city}".
- Collected and formatted the first 5 attraction snippets.

with DDGS() as ddgs:

  results = ddgs.text(f"Top tourist attractions in {city}")

  attractions = "\n".join([f"- {r['body']}" for r in results[:5]])

5. Generating Final Summary

- Passed both weather and attraction info to Gemini via a custom prompt.
- Prompt template included contextual cues:
    - Role: Travel Assistant
    - Purpose: Summarize trip insights in friendly tone

prompt = f"""

You are a travel assistant. Based on the data below, summarize in a friendly way what a tourist can expect in {city}.

Weather Info:{weather_info}

Attractions:{attraction_info}

💡 Sample Output (City: Tokyo)

The current weather in Tokyo is partly cloudy with a temperature of 28°C.
Some top attractions include the Tokyo Skytree, Senso-ji Temple, Shibuya Crossing, Ueno Park, and Meiji Shrine.
Tokyo offers a vibrant blend of tradition and futuristic excitement, perfect for any traveler!

📌 Conclusion

This project demonstrates how to effectively integrate:

- Real-time **API services**
- Simple **search-based data collection**
- A **powerful large language model** for reasoning and generation

The resulting Travel Assistant is:

- Fast
- Easy to scale
- Real-world applicable

It can be extended into chatbot form or mobile travel assistant apps.