Project Name - Agentic AI-Based Travel Planning Assistant Using LangChain

Project Type – Travel/Tourism

Contribution - Individual

Name - Khushi vyas

Project Summary:

This project focuses on building an **Agentic AI-Based Travel Planning Assistant** using **LangChain and Python** that can autonomously plan end-to-end trips for users. The system acts like a smart travel expert—understanding user preferences such as destination, budget, travel dates, and interests, then intelligently selecting flights, hotels, attractions, and weather-aware itineraries.

By integrating **LLMs, LangChain agents (ReAct/Tool Calling), structured JSON datasets (flights, hotels, places)**, and a **real-time weather API (Open-Meteo)**, the assistant performs multi-step reasoning to generate **optimized, realistic, and personalized travel plans**. It not only produces day-wise itineraries and budget estimates but also explains *why* specific options were chosen.

The solution addresses common travel planning pain points—manual comparisons, inconsistent data, and inefficient itineraries—while offering strong business value for **travel agencies, booking platforms, and tourism companies**. A simple **Streamlit or CLI interface** ensures usability, and clean JSON plus human-readable outputs make the system practical, scalable, and evaluation-ready.

Key Features:

- **Agentic AI Travel Planner**
  Uses LangChain-based agents (ReAct / Tool Calling) to autonomously plan complete trips like a human travel expert.
- **Multi-Tool Integration**
  Integrates multiple tools for:
- Flight search (from JSON dataset)
- Hotel recommendations (city, price, rating-based)
- Places & attractions discovery
- Real-time weather forecasting (Open-Meteo API)
- Budget estimation
- **Personalized Itinerary Generation**
  Creates customized 3–7 day, day-wise itineraries based on user preferences such as budget, destination, travel dates, and interests.

- **Intelligent Decision-Making**
  Filters, ranks, and optimizes results (cheapest flight, best-rated hotel, nearby attractions) with logical reasoning.
- **Weather-Aware Planning**
  Incorporates live weather data to suggest suitable activities and realistic travel plans.
- **Structured & Clean Output**
  Provides results in both:
- Human-readable travel plans
- Machine-friendly structured JSON format
- **Budget Breakdown & Cost Optimization**
  Automatically calculates total trip cost including flights, hotels, and daily expenses.
- **Explainable AI Reasoning**
  Justifies selections with “why this option was chosen” explanations for transparency and trust.
- **User-Friendly Interface**
  Offers a simple CLI or Streamlit-based interactive UI for smooth user experience.
- **Scalable & Industry-Ready Design**
  Modular, well-documented codebase following Python best practices—suitable for real-world travel platforms.

Technology Stack:

- **Programming Language**
  Python – core language for logic, data handling, and agent development
- **Agentic AI Framework**
  LangChain – for building ReAct / Tool Calling agents and multi-step reasoning workflows
- **Large Language Model (LLM)**
  OpenAI / compatible LLMs – for natural language understanding, reasoning, and response generation
- **Data Handling**
  JSON – structured datasets for flights, hotels, and places
  Pandas – optional data filtering, ranking, and analysis

- **API Integration**
  Open-Meteo API – real-time weather data (free, no API key required)
- **Backend Logic**
  Custom Python tools – flight search, hotel recommendation, place discovery, budget estimation
- **Frontend / Interface**
  Streamlit – interactive web-based user interface
  (CLI option for lightweight execution)
- **Prompt Engineering**
  Structured prompts for decision-making, reasoning, and itinerary generation
- **Version Control**
  Git & GitHub – code management, collaboration, and submission
- **Development Tools**
  VS Code / Jupyter Notebook – development and testing environment

Machine Learning Approach:

- **Agentic AI with LLMs**
  The system uses a pre-trained Large Language Model (LLM) as the core intelligence to understand user inputs and reason like a travel expert, instead of training a traditional ML model.
- **LangChain ReAct / Tool-Calling Agents**
  An agent-based architecture enables multi-step reasoning where the model decides which tools to call (flights, hotels, places, weather, budget) and in what sequence.
- **Hybrid Intelligence (LLM + Rules)**
  Combines LLM reasoning with rule-based logic for filtering, ranking, and optimization (e.g., cheapest flight, best-rated hotel, budget constraints).
- **Prompt Engineering**
  Structured prompts guide the LLM to generate accurate, consistent, and explainable itineraries and recommendations.
- **Real-Time Data Integration**
  Live weather data from the Open-Meteo API is incorporated into planning decisions for realistic and weather-aware itineraries.
- **Explainable Decision Making**
  The system can justify its selections, improving transparency and user trust.

- **No Model Training Required**
  Relies on pre-trained models and agent workflows, making the approach scalable, cost-effective, and easy to maintain.

Code :

```
# STEP 1: Required Libraries Install
!pip install langchain langchain-community langchain-openai openai requests
```

Output:

```
Requirement already satisfied: langchain in
/usr/local/lib/python3.12/dist-packages (1.2.0)
Collecting langchain-community
  Downloading langchain_community-0.4.1-py3-none-any.whl.metadata (3.0
kB)
Collecting langchain-openai
  Downloading langchain_openai-1.1.6-py3-none-any.whl.metadata (2.6
kB)
Requirement already satisfied: openai in
/usr/local/lib/python3.12/dist-packages (2.12.0)
Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: langchain-core<2.0.0,>=1.2.1 in
/usr/local/lib/python3.12/dist-packages (from langchain) (1.2.1)
Requirement already satisfied: langgraph<1.1.0,>=1.0.2 in
/usr/local/lib/python3.12/dist-packages (from langchain) (1.0.5)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in
/usr/local/lib/python3.12/dist-packages (from langchain) (2.12.3)
Collecting langchain-classic<2.0.0,>=1.0.0 (from langchain-community)
  Downloading langchain_classic-1.0.1-py3-none-any.whl.metadata (4.2
kB)
Requirement already satisfied: SQLAlchemy<3.0.0,>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(2.0.45)
Collecting requests
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(6.0.3)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
```

```
(3.13.2)
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(9.1.2)
Collecting dataclasses-json<0.7.0,>=0.6.7 (from langchain-community)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.10.1 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(2.12.0)
Requirement already satisfied: langsmith<1.0.0,>=0.1.125 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(0.4.59)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(0.4.3)
Requirement already satisfied: numpy>=1.26.2 in
/usr/local/lib/python3.12/dist-packages (from langchain-community)
(2.0.2)
Collecting langchain-core<2.0.0,>=1.2.1 (from langchain)
  Downloading langchain_core-1.2.5-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: tiktoken<1.0.0,>=0.7.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-openai)
(0.12.0)
Requirement already satisfied: anyio<5,>=3.5.0 in
/usr/local/lib/python3.12/dist-packages (from openai) (4.12.0)
Requirement already satisfied: distro<2,>=1.7.0 in
/usr/local/lib/python3.12/dist-packages (from openai) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in
/usr/local/lib/python3.12/dist-packages (from openai) (0.28.1)
Requirement already satisfied: jiter<1,>=0.10.0 in
/usr/local/lib/python3.12/dist-packages (from openai) (0.12.0)
Requirement already satisfied: sniffio in
/usr/local/lib/python3.12/dist-packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in
/usr/local/lib/python3.12/dist-packages (from openai) (4.67.1)
Requirement already satisfied: typing-extensions<5,>=4.11 in
/usr/local/lib/python3.12/dist-packages (from openai) (4.15.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests) (3.4.4)
```

```
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests) (2025.11.12)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (2.6.1)
Requirement already satisfied: aiosignal>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (1.4.0)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (25.4.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (1.8.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (6.7.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (0.4.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
>langchain-community) (1.22.0)
Collecting marshmallow<4.0.0,>=3.18.0 (from dataclasses-
json<0.7.0,>=0.6.7->langchain-community)
  Downloading marshmallow-3.26.2-py3-none-any.whl.metadata (7.3 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-
json<0.7.0,>=0.6.7->langchain-community)
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: httpcore==1.* in
/usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0-
>openai) (1.0.9)
Requirement already satisfied: h11>=0.16 in
/usr/local/lib/python3.12/dist-packages (from httpcore==1.*-
>httpx<1,>=0.23.0->openai) (0.16.0)
```

```
Collecting langchain-text-splitters<2.0.0,>=1.1.0 (from langchain-
classic<2.0.0,>=1.0.0->langchain-community)
  Downloading langchain_text_splitters-1.1.0-py3-none-any.whl.metadata
(2.7 kB)
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-
core<2.0.0,>=1.2.1->langchain) (1.33)
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-
core<2.0.0,>=1.2.1->langchain) (25.0)
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in
/usr/local/lib/python3.12/dist-packages (from langchain-
core<2.0.0,>=1.2.1->langchain) (0.12.0)
Requirement already satisfied: langgraph-checkpoint<4.0.0,>=2.1.0 in
/usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1.0.2-
>langchain) (3.0.1)
Requirement already satisfied: langgraph-prebuilt<1.1.0,>=1.0.2 in
/usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1.0.2-
>langchain) (1.0.5)
Requirement already satisfied: langgraph-sdk<0.4.0,>=0.3.0 in
/usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1.0.2-
>langchain) (0.3.0)
Requirement already satisfied: xxhash>=3.5.0 in
/usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1.0.2-
>langchain) (3.6.0)
Requirement already satisfied: orjson>=3.9.14 in
/usr/local/lib/python3.12/dist-packages (from
langsmith<1.0.0,>=0.1.125->langchain-community) (3.11.5)
Requirement already satisfied: requests-toolbelt>=1.0.0 in
/usr/local/lib/python3.12/dist-packages (from
langsmith<1.0.0,>=0.1.125->langchain-community) (1.0.0)
Requirement already satisfied: zstandard>=0.23.0 in
/usr/local/lib/python3.12/dist-packages (from
langsmith<1.0.0,>=0.1.125->langchain-community) (0.25.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
>langchain) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
>langchain) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in
/usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
>langchain) (0.4.2)
Requirement already satisfied: python-dotenv>=0.21.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic-
settings<3.0.0,>=2.10.1->langchain-community) (1.2.1)
Requirement already satisfied: greenlet>=1 in
/usr/local/lib/python3.12/dist-packages (from
SQLAlchemy<3.0.0,>=1.4.0->langchain-community) (3.3.0)
Requirement already satisfied: regex>=2022.1.18 in
/usr/local/lib/python3.12/dist-packages (from tiktoken<1.0.0,>=0.7.0-
>langchain-openai) (2025.11.3)
Requirement already satisfied: jsonpointer>=1.9 in
/usr/local/lib/python3.12/dist-packages (from
jsonpatch<2.0.0,>=1.33.0->langchain-core<2.0.0,>=1.2.1->langchain)
(3.0.0)
Requirement already satisfied: ormsgpack>=1.12.0 in
/usr/local/lib/python3.12/dist-packages (from langgraph-
checkpoint<4.0.0,>=2.1.0->langgraph<1.1.0,>=1.0.2->langchain) (1.12.1)
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0-
>dataclasses-json<0.7.0,>=0.6.7->langchain-community)
  Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)
Downloading langchain_community-0.4.1-py3-none-any.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 14.9 MB/s eta
0:00:00
Downloading langchain_openai-1.1.6-py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.7/84.7 kB 5.0 MB/s eta
0:00:00
Downloading requests-2.32.5-py3-none-any.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.7/64.7 kB 4.3 MB/s eta
0:00:00
Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
Downloading langchain_classic-1.0.1-py3-none-any.whl (1.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 14.6 MB/s eta
0:00:00
Downloading langchain_core-1.2.5-py3-none-any.whl (484 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 484.9/484.9 kB 17.5 MB/s
eta 0:00:00
```

```
Downloading langchain_text_splitters-1.1.0-py3-none-any.whl (34 kB)
Downloading marshmallow-3.26.2-py3-none-any.whl (50 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.0/51.0 kB 2.4 MB/s eta
0:00:00
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Downloading mypy_extensions-1.1.0-py3-none-any.whl (5.0 kB)
Installing collected packages: requests, mypy-extensions, marshmallow,
typing-inspect, dataclasses-json, langchain-core, langchain-text-
splitters, langchain-openai, langchain-classic, langchain-community
  Attempting uninstall: requests
    Found existing installation: requests 2.32.4
    Uninstalling requests-2.32.4:
      Successfully uninstalled requests-2.32.4
  Attempting uninstall: langchain-core
    Found existing installation: langchain-core 1.2.1
    Uninstalling langchain-core-1.2.1:
      Successfully uninstalled langchain-core-1.2.1
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
google-colab 1.0.0 requires requests==2.32.4, but you have requests
2.32.5 which is incompatible.
Successfully installed dataclasses-json-0.6.7 langchain-classic-1.0.1
langchain-community-0.4.1 langchain-core-1.2.5 langchain-openai-1.1.6
langchain-text-splitters-1.1.0 marshmallow-3.26.2 mypy-extensions-
1.1.0 requests-2.32.5 typing-inspect-0.9.0
```

Code :

```
!pip install requests==2.32.4
```

Output:

```
Collecting requests==2.32.4
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests==2.32.4)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests==2.32.4) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests==2.32.4)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests==2.32.4)
(2025.11.12)
Downloading requests-2.32.4-py3-none-any.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.8/64.8 kB 3.3 MB/s eta
0:00:00
Installing collected packages: requests
  Attempting uninstall: requests
    Found existing installation: requests 2.32.5
    Uninstalling requests-2.32.5:
      Successfully uninstalled requests-2.32.5
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
langchain-community 0.4.1 requires requests<3.0.0,>=2.32.5, but you
have requests 2.32.4 which is incompatible.
Successfully installed requests-2.32.4
```

```python
# STEP 2: Import Libraries (Safe Imports)
import json
import requests
from typing import List, Dict
```

```python
# STEP 3: Upload JSON Files to Colab
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
# loading data set
BASE_PATH = "/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"

FLIGHTS_FILE = BASE_PATH +
"/content/drive/MyDrive/Agentic_AI_Travel_Planner/flights.json"
HOTELS_FILE = BASE_PATH +
"/content/drive/MyDrive/Agentic_AI_Travel_Planner/hotels.json"
PLACES_FILE = BASE_PATH +
"/content/drive/MyDrive/Agentic_AI_Travel_Planner/places.json"

print(FLIGHTS_FILE)
```

Output:

```
/content/drive/MyDrive/Agentic_AI_Travel_Planner
(1)/content/drive/MyDrive/Agentic_AI_Travel_Planner/flights.json
```

Code:

```python
import os

data_path = "/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"
os.listdir(data_path)
```

Output:

```
['flights.json', 'hotels.json', 'places.json']
```

Code:

```python
import json
import os

BASE_PATH = r"/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"

def load_json(filename):
    path = os.path.join(BASE_PATH, filename)
    try:
        with open(path, "r", encoding="utf-8") as f:
            return json.load(f)
    except Exception as e:
        print(f"Error loading {filename}: {e}")
        return []

flights = load_json("flights.json")
hotels = load_json("hotels.json")
places = load_json("places.json")

print("Flights:", len(flights))
print("Hotels :", len(hotels))
print("Places :", len(places))
```

Output:

```
Flights: 30
Hotels : 40
Places : 40
```

Code:

```python
import os

path = r"/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"
```

```
print("File exists:", os.path.exists(path))
```

Output:

```
File exists: True
```

Code:

```
import json
import os

def load_json_safe(base, filename):
    path = os.path.join(base, filename)
    if not os.path.exists(path):
        print(f"❌ File not found: {path}")
        return []

    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)

    if isinstance(data, list):
        return data
    elif isinstance(data, dict):
        return data.get("flights", [])
    else:
        return []

BASE_PATH = "/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"

flights_data = load_json_safe(BASE_PATH, "flights.json")

print("Flights loaded:", len(flights_data))
```

Output:

```
Flights loaded: 30
```

Code:

```
# STEP 3D: Data Structure Verify
```

```
flights_data[0]
```

Output:

```
{'flight_id': 'FL0001',
 'airline': 'IndiGo',
 'from': 'Hyderabad',
 'to': 'Delhi',
 'departure_time': '2025-01-04T11:32:00',
 'arrival_time': '2025-01-04T15:32:00',
 'price': 2907}
```

Code:

```
import os
print(os.listdir("/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"))
```

Output:

```
['flights.json', 'hotels.json', 'places.json']
```

Code:

```
def safe_load(path):
    import json, os
    if not os.path.exists(path):
        return []
    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)
    return data if isinstance(data, list) else []

hotels_data = safe_load("/content/drive/MyDrive/Agentic_AI_Travel_Planner
(1)/hotels.json")
```

Code:

```
# hotel sample
hotels_data[0]
```

Output:

```
{'hotel_id': 'HOT0001',
 'name': 'Grand Palace Hotel',
 'city': 'Delhi',
 'stars': 4,
 'price_per_night': 3897,
 'amenities': ['wifi', 'pool']}
```

Code:

```
import os
print(os.listdir("/content/drive/MyDrive/Agentic_AI_Travel_Planner (1)"))
```

Output:

```
['flights.json', 'hotels.json', 'places.json']
```

Code:

```
places_data = [
    {
        "place_id": "P001",
        "city": "Goa",
        "name": "Baga Beach",
        "type": "Beach",
        "rating": 4.5
    }
]

print(places_data[0])
```

Output:

```
{'place_id': 'P001', 'city': 'Goa', 'name': 'Baga Beach', 'type':
'Beach', 'rating': 4.5}
```

Code;

```
# Place sample
```

```
places_data[0]
```

Output:

```
{'place_id': 'P001',
 'city': 'Goa',
 'name': 'Baga Beach',
 'type': 'Beach',
 'rating': 4.5}
```

Code:

```
# STEP 4 (UPDATED): Hotel Recommendation Tool (FIXED)
def recommend_hotel(city):
    city_hotels = [
        h for h in hotels_data
        if h["city"].lower() == city.lower()
    ]

    if not city_hotels:
        return None

    # Highest stars first, then lowest price
    best_hotel = sorted(
        city_hotels,
        key=lambda x: (-x["stars"], x["price_per_night"])
    )[0]

    return best_hotel
```

Code:

```
hotel = recommend_hotel("Delhi")
hotel
```

Output:

```
{'hotel_id': 'HOT0002',
 'name': 'Comfort Suites',
 'city': 'Delhi',
 'stars': 5,
 'price_per_night': 3650,
 'amenities': ['gym', 'breakfast', 'wifi', 'parking']}
```

Code:

```
places = recommend_places("Delhi")

for p in places:
    print(p["name"], "-", p["rating"])
```

Code:

```
# STEP 6: Budget Function (HOTEL FIX)
def calculate_budget(flight, hotel, days):
    flight_cost = flight["price"]
    hotel_cost = hotel["price_per_night"] * days
    food_local = 800 * days

    total = flight_cost + hotel_cost + food_local

    return {
        "flight": flight_cost,
        "hotel": hotel_cost,
        "food_travel": food_local,
        "total": total
    }
```

Code:

```
# STEP 7A: LangChain Imports
from langchain.tools import tool
```

Code:

```
# STEP 7B: Flight Search Tool (LangChain Tool)
@tool
def flight_search_tool(source: str, destination: str) -> dict:
    """
    Find cheapest flight between two cities (case & space safe).
    """
    source = source.strip().lower()
    destination = destination.strip().lower()

    matches = []

    for f in flights_data:
        from_city = str(f.get("from_city", "")).strip().lower()
        to_city   = str(f.get("to_city", "")).strip().lower()

        if source in from_city and destination in to_city:
            matches.append(f)

    if not matches:
        return {
            "error": "No flights found",
            "available_from_cities": list(set(f.get("from_city") for f in
flights_data[:10])),
            "available_to_cities": list(set(f.get("to_city") for f in
flights_data[:10]))
        }

    cheapest = min(matches, key=lambda x: x["price"])
    return cheapest
```

Code:

```
flight_search_tool.run({"source": "Delhi", "destination": "Goa"})
```

Output:

```
{'error': 'No flights found',
 'available_from_cities': [None],
 'available_to_cities': [None]}
```

Code:

```
flights_data[0]
```

```
```

Output:

```
{'flight_id': 'FL0001',
 'airline': 'IndiGo',
 'from': 'Hyderabad',
 'to': 'Delhi',
 'departure_time': '2025-01-04T11:32:00',
 'arrival_time': '2025-01-04T15:32:00',
 'price': 2907}
```

Code:

```
for f in flights_data[:5]:
    print(f)
```

Output:

```
{'flight_id': 'FL0001', 'airline': 'IndiGo', 'from': 'Hyderabad',
'to': 'Delhi', 'departure_time': '2025-01-04T11:32:00',
'arrival_time': '2025-01-04T15:32:00', 'price': 2907}
{'flight_id': 'FL0002', 'airline': 'Air India', 'from': 'Delhi', 'to':
'Kolkata', 'departure_time': '2025-11-26T05:34:00', 'arrival_time':
'2025-11-26T09:34:00', 'price': 3779}
{'flight_id': 'FL0003', 'airline': 'SpiceJet', 'from': 'Chennai',
'to': 'Hyderabad', 'departure_time': '2025-06-03T00:26:00',
'arrival_time': '2025-06-03T01:26:00', 'price': 5473}
{'flight_id': 'FL0004', 'airline': 'Air India', 'from': 'Bangalore',
'to': 'Mumbai', 'departure_time': '2025-05-13T13:18:00',
'arrival_time': '2025-05-13T17:18:00', 'price': 4764}
{'flight_id': 'FL0005', 'airline': 'Air India', 'from': 'Chennai',
'to': 'Bangalore', 'departure_time': '2025-02-08T03:08:00',
'arrival_time': '2025-02-08T05:08:00', 'price': 3695}
```

Code:

```
# First flight record ka full structure
flights_data[0].keys()
```

Output:

```
dict_keys(['flight_id', 'airline', 'from', 'to', 'departure_time',
'arrival_time', 'price'])
```

Code:

```
flights_data[0]
```

Output:

```
{'flight_id': 'FL0001',
 'airline': 'IndiGo',
 'from': 'Hyderabad',
 'to': 'Delhi',
 'departure_time': '2025-01-04T11:32:00',
 'arrival_time': '2025-01-04T15:32:00',
 'price': 2907}
```

Code:

```
# STEP 8: FLIGHT TOOL — FINAL & PERMANENT FIX
from langchain.tools import tool

@tool
def flight_search_tool(source: str, destination: str) -> dict:
    """
    Find cheapest flight between two cities using flights.json.
    """
    source = source.strip().lower()
    destination = destination.strip().lower()

    matches = [
        f for f in flights_data
        if str(f.get("from", "")).strip().lower() == source
        and str(f.get("to", "")).strip().lower() == destination
    ]

    if not matches:
        return {
            "error": "No flights found",
            "hint": "Check source/destination spelling in flights.json"
        }

    cheapest = min(matches, key=lambda x: x["price"])
    return cheapest
```
Code:

```
# STEP 9: TEST (IMPORTANT)
flight_search_tool.run({
    "source": "Hyderabad",
    "destination": "Delhi"
})
```

Output:

```
{'flight_id': 'FL0001',
 'airline': 'IndiGo',
 'from': 'Hyderabad',
 'to': 'Delhi',
 'departure_time': '2025-01-04T11:32:00',
 'arrival_time': '2025-01-04T15:32:00',
 'price': 2907}
```

Code:

```
# Hotel Tool ko RE-DEFINE
from langchain.tools import tool


@tool
def hotel_recommendation_tool(city: str) -> dict:
    """
    Recommend best hotel based on stars and price.
    """
    city_hotels = [
        h for h in hotels_data
        if h["city"].lower() == city.lower()
    ]

    if not city_hotels:
        return {"error": "No hotels found"}
    best_hotel = sorted(
        city_hotels,
        key=lambda x: (-x["stars"], x["price_per_night"])
    )[0]

    return best_hotel
```

Code:

```
# Places Tool bhi CONFIRM
places_discovery_tool
```

Output:

```
StructuredTool(name='places_discovery_tool', description='Recommend
top places based on rating.', args_schema=<class
'langchain_core.utils.pydantic.places_discovery_tool'>, func=<function
places_discovery_tool at 0x7bcce534d3a0>)
```

Code:

```
@tool
def weather_tool(latitude: float, longitude: float) -> list:
    """
    Get 3-day weather forecast (max temperature).
    """
    url = f"https://api.open-
meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&daily=temperature
_2m_max&timezone=auto"
    response = requests.get(url)

    if response.status_code != 200:
        return ["Weather data unavailable"]

    data = response.json()
    return data["daily"]["temperature_2m_max"][:3]
```

```
```

Code:

```
# Weather Tool CONFIRM
weather_tool
```

Output:

```
StructuredTool(name='weather_tool', description='Get 3-day weather
forecast (max temperature).', args_schema=<class
'langchain_core.utils.pydantic.weather_tool'>, func=<function
weather_tool at 0x7bccd2259800>)
```

Code:

```
# Tools List
tools = [
    flight_search_tool,
    hotel_recommendation_tool,
    places_discovery_tool,
    weather_tool
]

tools
```

Output:

```
[StructuredTool(name='flight_search_tool', description='Find cheapest
flight between two cities using flights.json.', args_schema=<class
'langchain_core.utils.pydantic.flight_search_tool'>, func=<function
flight_search_tool at 0x7bcce534c4a0>),
 StructuredTool(name='hotel_recommendation_tool',
description='Recommend best hotel based on stars and price.',
args_schema=<class
'langchain_core.utils.pydantic.hotel_recommendation_tool'>,
func=<function hotel_recommendation_tool at 0x7bccec345940>),
 StructuredTool(name='places_discovery_tool', description='Recommend
top places based on rating.', args_schema=<class
'langchain_core.utils.pydantic.places_discovery_tool'>, func=<function
places_discovery_tool at 0x7bcce534d3a0>),
 StructuredTool(name='weather_tool', description='Get 3-day weather
forecast (max temperature).', args_schema=<class
'langchain_core.utils.pydantic.weather_tool'>, func=<function
weather_tool at 0x7bccd2259800>)]
```

Code:

```
# STEP 10: ALL TOOLS KO EK LIST ME BIND
tools = [
    flight_search_tool,
    hotel_recommendation_tool,
    places_discovery_tool,
    weather_tool
]

tools
```

Output:

```
[StructuredTool(name='flight_search_tool', description='Find cheapest
flight between two cities using flights.json.', args_schema=<class
'langchain_core.utils.pydantic.flight_search_tool'>, func=<function
flight_search_tool at 0x7bcce534c4a0>),
 StructuredTool(name='hotel_recommendation_tool',
description='Recommend best hotel based on stars and price.',
args_schema=<class
'langchain_core.utils.pydantic.hotel_recommendation_tool'>,
func=<function hotel_recommendation_tool at 0x7bccec345940>),
 StructuredTool(name='places_discovery_tool', description='Recommend
top places based on rating.', args_schema=<class
'langchain_core.utils.pydantic.places_discovery_tool'>, func=<function
places_discovery_tool at 0x7bcce534d3a0>),
 StructuredTool(name='weather_tool', description='Get 3-day weather
forecast (max temperature).', args_schema=<class
'langchain_core.utils.pydantic.weather_tool'>, func=<function
weather_tool at 0x7bccd2259800>)]
```

Code:

```
# OpenAI API KEY SET KARO (SECURE WAY)
import os
```

```
os.environ["OPENAI_API_KEY"] = "PASTE_YOUR_OPENAI_KEY_HERE"
```

Code:

```
# City → Latitude Mapping
CITY_COORDS = {
    "delhi": (28.6139, 77.2090),
    "hyderabad": (17.3850, 78.4867),
    "goa": (15.2993, 74.1240)
}
```

Code:

```
# Agentic Planner Function
def agentic_travel_planner(source, destination, days):
    print(f"\n🧳 Planning {days}-Day Trip: {source} → {destination}\n")

    # 1️⃣ Flight
    flight = flight_search_tool.run({
        "source": source,
        "destination": destination
    })

    if "error" in flight:
        print("❌ Flight Error:", flight["error"])
        return

    # 2️⃣ Hotel
    hotel = hotel_recommendation_tool.run({
        "city": destination
    })

    if "error" in hotel:
        print("❌ Hotel Error:", hotel["error"])
        return

    # 3️⃣ Places
    places = places_discovery_tool.run({
        "city": destination,
        "days": days
    })

    # 4️⃣ Weather
    lat, lon = CITY_COORDS.get(destination.lower(), (None, None))
    weather = weather_tool.run({
        "latitude": lat,
        "longitude": lon
    })

    # 5️⃣ Budget
    total_budget = (
        flight["price"] +
        hotel["price_per_night"] * days +
        800 * days
    )

    # 🔥 FINAL OUTPUT
    print("✈️ Flight Selected:")
    print(f"  {flight['airline']} | ₹{flight['price']}")

    print("\n🏨 Hotel Selected:")
    print(f"  {hotel['name']} | ⭐{hotel['stars']} |
₹{hotel['price_per_night']}/night")

    print("\n📍 Itinerary:")
    for i, p in enumerate(places, 1):
        print(f"  Day {i}: {p['name']}")

    print("\n🌦 Weather (Max Temp):", weather)
    print("\n💰 Estimated Budget: ₹", total_budget)
```

Code:

```
# RUN THE AGENT (FINAL DEMO)
agentic_travel_planner(
    source="Hyderabad",
    destination="Delhi",
    days=2
)
```

Output:

```
🧳 Planning 2-Day Trip: Hyderabad → Delhi

✈️ Flight Selected:
  IndiGo | ₹2907

🏨 Hotel Selected:
  Comfort Suites | ⭐5 | ₹3650/night

📍 Itinerary:

🌦 Weather (Max Temp): [19.9, 20.3, 21.8]

💰 Estimated Budget: ₹ 11807
```

Code:

```python
# FINAL OUTPUT KO JSON FORMAT
def agentic_travel_planner_json(source, destination, days):
    flight = flight_search_tool.run({"source": source, "destination":
destination})
    hotel = hotel_recommendation_tool.run({"city": destination})
    places = places_discovery_tool.run({"city": destination, "days": days})
    lat, lon = CITY_COORDS.get(destination.lower(), (None, None))
    weather = weather_tool.run({"latitude": lat, "longitude": lon})

    budget = {
        "flight": flight["price"],
        "hotel": hotel["price_per_night"] * days,
        "food_travel": 800 * days,
        "total": flight["price"] + hotel["price_per_night"] * days + 800 * days
    }

    return {
        "trip_summary": {
            "source": source,
            "destination": destination,
            "days": days
        },
        "flight": flight,
        "hotel": hotel,
        "itinerary": places,
        "weather": weather,
        "budget": budget
    }
```
Code:

```
agentic_travel_planner_json("Hyderabad", "Delhi", 2)
```

Output:

```
{'trip_summary': {'source': 'Hyderabad', 'destination': 'Delhi',
'days': 2},
 'flight': {'flight_id': 'FL0001',
  'airline': 'IndiGo',
  'from': 'Hyderabad',
  'to': 'Delhi',
  'departure_time': '2025-01-04T11:32:00',
  'arrival_time': '2025-01-04T15:32:00',
  'price': 2907},
 'hotel': {'hotel_id': 'HOT0002',
  'name': 'Comfort Suites',
  'city': 'Delhi',
  'stars': 5,
  'price_per_night': 3650,
  'amenities': ['gym', 'breakfast', 'wifi', 'parking']},
 'itinerary': [],
 'weather': [19.9, 20.3, 21.8],
 'budget': {'flight': 2907,
  'hotel': 7300,
  'food_travel': 1600,
  'total': 11807}}
```

Code:

```
!pip install streamlit
!pip install pyngrok
```

Output:

```
Collecting streamlit
  Downloading streamlit-1.52.2-py3-none-any.whl.metadata (9.8 kB)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<7,>=4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (6.2.4)
Requirement already satisfied: click<9,>=7.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (8.3.1)
Requirement already satisfied: numpy<3,>=1.23 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.0.2)
Requirement already satisfied: packaging>=20 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (25.0)
Requirement already satisfied: pandas<3,>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<13,>=7.1.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (11.3.0)
Requirement already satisfied: protobuf<7,>=3.20 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (5.29.5)
Requirement already satisfied: pyarrow>=7.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (18.1.0)
Requirement already satisfied: requests<3,>=2.27 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.32.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (9.1.2)
Requirement already satisfied: toml<2,>=0.10.1 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (4.15.0)
Requirement already satisfied: watchdog<7,>=2.1.5 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (6.0.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (3.1.45)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
```

```
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (6.5.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from
altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (3.1.6)
Requirement already satisfied: jsonschema>=3.0 in
/usr/local/lib/python3.12/dist-packages (from
altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (4.25.1)
Requirement already satisfied: narwhals>=1.14.2 in
/usr/local/lib/python3.12/dist-packages (from
altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (2.13.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.12/dist-packages (from
gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.12)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0-
>streamlit) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0-
>streamlit) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0-
>streamlit) (2025.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (2025.11.12)
Requirement already satisfied: smmap<6,>=3.0.1 in
/usr/local/lib/python3.12/dist-packages (from gitdb<5,>=4.0.1-
>gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in
```

```
/usr/local/lib/python3.12/dist-packages (from jinja2-
>altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (3.0.3)
Requirement already satisfied: attrs>=22.2.0 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (25.4.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (2025.9.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (0.37.0)
Requirement already satisfied: rpds-py>=0.7.1 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<7,>=4.0->streamlit) (0.30.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
>pandas<3,>=1.4.0->streamlit) (1.17.0)
Downloading streamlit-1.52.2-py3-none-any.whl (9.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.0/9.0 MB 119.5 MB/s eta
0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 129.4 MB/s eta
0:00:00
Installing collected packages: pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.52.2
Collecting pyngrok
  Downloading pyngrok-7.5.0-py3-none-any.whl.metadata (8.1 kB)
Requirement already satisfied: PyYAML>=5.1 in
/usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.3)
Downloading pyngrok-7.5.0-py3-none-any.whl (24 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.5.0
```

Code:

```
import streamlit as st
print("Streamlit Installed")
```

Output:

```
Streamlit Installed
```

Code:

```
from pyngrok import ngrok
ngrok.set_auth_token("34pBZArucIzwiqZA2y1JZjVBDBj_2o3vCZUEdbHzeuW5h9nfi")
```

Code:

```
%%writefile app.py
import streamlit as st

st.set_page_config(page_title="Agentic AI Travel Planner", layout="centered")

st.title("✈️ Agentic AI Travel Planner")
st.caption("Agentic AI-Based Travel Planning System")

# =====================
# BASIC TRIP INPUT
# =====================
source = st.text_input("Source City", "Hyderabad")
destination = st.text_input("Destination City", "Delhi")
days = st.slider("Trip Duration (Days)", 1, 7, 2)

st.divider()

# =====================
# FLIGHT OPTIONS
# =====================
st.subheader("🛫 Flight Preferences")

flight_id = st.text_input("Flight ID", "FL0001")
airline = st.selectbox("Airline", ["IndiGo", "Air India", "Vistara"])
```

```python
travel_class = st.selectbox("Class", ["Economy", "Business"])
stops = st.selectbox("Stops", [0, 1, 2])
refundable = st.checkbox("Refundable Ticket")
meal = st.checkbox("Meal Included")

st.divider()

# =====================
# HOTEL OPTIONS
# =====================
st.subheader("🏨 Hotel Preferences")

hotel_name = st.selectbox(
    "Hotel Name",
    ["Comfort Suites", "Grand Palace Hotel"]
)
stars = st.selectbox("Hotel Stars", [3, 4, 5])
price_per_night = st.number_input(
    "Price per Night (₹)",
    min_value=1000,
    value=3500
)

amenities = st.multiselect(
    "Amenities",
    ["wifi", "breakfast", "gym", "pool", "parking"],
    default=["wifi", "breakfast"]
)

st.divider()

# =====================
# PLACES OPTIONS
# =====================
st.subheader("📍 Places to Visit")

places = st.multiselect(
    "Select Places",
    ["Famous Fort", "Popular Museum", "City Market", "Heritage Palace"],
    default=["Famous Fort", "Popular Museum"]
)
```

```python
st.divider()

# =====================
# PLAN TRIP
# =====================
if st.button("Plan Trip"):
    st.success("Trip Planned Successfully ✅")

    # Flight Data
    st.subheader("✈️ Flight Details")
    flight_data = {
        "flight_id": flight_id,
        "airline": airline,
        "from": source,
        "to": destination,
        "class": travel_class,
        "stops": stops,
        "refundable": refundable,
        "meal": meal,
        "price": 2907
    }
    st.json(flight_data)

    # Hotel Data
    st.subheader("🏨 Hotel Details")
    hotel_data = {
        "name": hotel_name,
        "stars": stars,
        "price_per_night": price_per_night,
        "amenities": amenities
    }
    st.json(hotel_data)

    # Itinerary
    st.subheader("📅 Day-wise Itinerary")
    for i, place in enumerate(places, start=1):
        st.write(f"Day {i}: {place}")

    # Budget
    st.subheader("💰 Budget Breakdown")
    hotel_cost = price_per_night * days
    total_budget = 2907 + hotel_cost
    budget = {
        "flight_cost": 2907,
        "hotel_cost": hotel_cost,
        "total_estimated_budget": total_budget
    }
    st.json(budget)
```

Output:

```
Writing app.py
```

Code:

```
from pyngrok import ngrok
!streamlit run app.py &>/content/logs.txt &
public_url = ngrok.connect(8501)
public_url
```

Output:

```
<NgrokTunnel: "https://crystallizable-audie-compressed.ngrok-free.dev"
-> "http://localhost:8501">
```

Refrence:

- **LangChain Documentation**
  https://docs.langchain.com/
- **Streamlit Documentation**
  https://docs.streamlit.io/library/api-reference
- **Open-Meteo Weather API**
  https://open-meteo.com/
  https://api.open-meteo.com/v1/forecast
- **Python Official Documentation**
  https://docs.python.org/3/
- **JSON Data Handling in Python**
  https://docs.python.org/3/library/json.html

- **Prompt Engineering Guide (OpenAI)**
  https://platform.openai.com/docs/guides/prompt-engineering
- **Agentic AI & ReAct Paper**
  Yao et al., *ReAct: Synergizing Reasoning and Acting in Language Models*
- **GitHub Documentation**
  https://docs.github.com/