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

```
Requirement already satisfied: langchain-openai in /usr/local/lib/python3.12/dist-packages (1.1.4)
Requirement already satisfied: openai in /usr/local/lib/python3.12/dist-packages (2.9.0)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.5)
Requirement already satisfied: langchain-core<2.0.0,>=1.1.2 in /usr/local/lib/python3.12/dist-packages (from langchain) (1.2.2
Requirement already satisfied: langgraph<1.1.0,>=1.0.2 in /usr/local/lib/python3.12/dist-packages (from langchain) (1.0.4)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.12/dist-packages (from langchain) (2.12.3)
Requirement already satisfied: langchain-classic<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langchain-commu
Requirement already satisfied: SQLAlchemy<3.0.0,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community)
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (6.0
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (3.1
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from langchain-commu
Requirement already satisfied: dataclasses-json<0.7.0,>=0.6.7 in /usr/local/lib/python3.12/dist-packages (from langchain-commun
Requirement already satisfied: pydantic-settings<3.0.0,>=2.10.1 in /usr/local/lib/python3.12/dist-packages (from langchain-comm
Requirement already satisfied: langsmith<1.0.0,>=0.1.125 in /usr/local/lib/python3.12/dist-packages (from langchain-community)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (0
Requirement already satisfied: numpy>=1.26.2 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.0.2)
Requirement already satisfied: tiktoken<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from langchain-openai) (0.12
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-
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langcha
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langch
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->lang
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langcha
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->langch
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib/python3.12/dist-packages (from dataclasses-json<0.7
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from dataclasses-json<0.7.0
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->oper
Requirement already satisfied: langchain-text-splitters<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langcha
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/python3.12/dist-packages (from langchain-core<2.0.0,
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from langchain-core<2.0.0
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/python3.12/dist-packages (from langchain-core<2.0.0,>=
Requirement already satisfied: langgraph-checkpoint<4.0.0,>=2.1.0 in /usr/local/lib/python3.12/dist-packages (from langgraph<1
Requirement already satisfied: langgraph-prebuilt<1.1.0,>=1.0.2 in /usr/local/lib/python3.12/dist-packages (from langgraph<1.1
Requirement already satisfied: langgraph-sdk<0.3.0,>=0.2.2 in /usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1
Requirement already satisfied: xxhash>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from langgraph<1.1.0,>=1.0.2->langcha
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1.125->langc
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1.125->la
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/python3.12/dist-packages (from pydantic-settings<3.0.0,
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from SQLAlchemy<3.0.0,>=1.4.0->langchain
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.12/dist-packages (from tiktoken<1.0.0,>=0.7.0->langch
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/dist-packages (from jsonpatch<2.0.0,>=1.33.0->lang
Requirement already satisfied: ormsgpack>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from langgraph-checkpoint<4.0.0,>=
```

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

```
Collecting requests==2.32.4
  Using cached 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.
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
Using cached requests-2.32.4-py3-none-any.whl (64 kB)
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
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
```

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

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# loading data set
BASE_PATH = "/content/drive/MyDrive/Agentic_AI_Travel_Planner"

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)
```

```
/content/drive/MyDrive/Agentic_AI_Travel_Planner/content/drive/MyDrive/Agentic_AI_Travel_Planner/flights.json
```

```
import os

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

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

```
# STEP 3C: Drive se JSON Load
import json

def load_json_from_drive(file_name):
    try:
        with open(f"{data_path}/{file_name}", "r") as f:
            return json.load(f)
    except Exception as e:
        print(f" Error loading {file_name}:", e)
        return []

flights_data = load_json_from_drive("flights.json")
hotels_data  = load_json_from_drive("hotels.json")
places_data  = load_json_from_drive("places.json")

print("Flights:", len(flights_data))
print("Hotels :", len(hotels_data))
print("Places :", len(places_data))
```

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

```
# STEP 3D: Data Structure Verify
flights_data[0]
```

```
{'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 sample
hotels_data[0]
```

```
{'hotel_id': 'HOT0001',
 'name': 'Grand Palace Hotel',
```

```
 'city': 'Delhi',
 'stars': 4,
 'price_per_night': 3897,
 'amenities': ['wifi', 'pool']}
```

```
# Place sample
places_data[0]
```

```
{'place_id': 'PLC0001',
 'name': 'Famous Fort',
 'city': 'Delhi',
 'type': 'lake',
 'rating': 4.6}
```

```
# 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
```

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

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

```
# STEP 5 (UPDATED): Places Recommendation Tool (FIXED)
def recommend_places(city, limit=3):
    city_places = [
        p for p in places_data
        if p["city"].lower() == city.lower()
    ]

    # Highest rating first
    city_places = sorted(
        city_places,
        key=lambda x: x["rating"],
        reverse=True
    )

    return city_places[:limit]
```

```
places = recommend_places("Delhi")

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

```
Famous Fort - 4.6
Popular Museum - 4.5
Beautiful Temple - 4.2
```

```
# 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
    }
```

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

```
# 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
```

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

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

```
flights_data[0]
```

```
{'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}
```

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

```
{'flight_id': 'FL0001', 'airline': 'IndiGo', 'from': 'Hyderabad', 'to': 'Delhi', 'departure_time': '2025-01-04T11:32:00', 'arriv
{'flight_id': 'FL0002', 'airline': 'Air India', 'from': 'Delhi', 'to': 'Kolkata', 'departure_time': '2025-11-26T05:34:00', 'arri
{'flight_id': 'FL0003', 'airline': 'SpiceJet', 'from': 'Chennai', 'to': 'Hyderabad', 'departure_time': '2025-06-03T00:26:00', 'a
{'flight_id': 'FL0004', 'airline': 'Air India', 'from': 'Bangalore', 'to': 'Mumbai', 'departure_time': '2025-05-13T13:18:00', 'a
{'flight_id': 'FL0005', 'airline': 'Air India', 'from': 'Chennai', 'to': 'Bangalore', 'departure_time': '2025-02-08T03:08:00', '
```

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

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

```
flights_data[0]
```

```
{'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}
```

```
# 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
```

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

```
{'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 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
```

```
@tool
def places_discovery_tool(city: str, days: int = 3) -> list:
    """
    Recommend top places based on rating.
    """
    city_places = [
        p for p in places_data
        if p["city"].lower() == city.lower()
    ]

    city_places = sorted(
        city_places,
        key=lambda x: x["rating"],
        reverse=True
    )

    return city_places[:days]
```

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

```
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 0x7d01a3bce700>)
```

```
@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=
    response = requests.get(url)

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

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

```
# Weather Tool CONFIRM
weather_tool
```

```
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 0x7d018418cd60>)
```

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

tools
```

```
[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 0x7d01a2b52020>),
 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
0x7d01a0daa980>),
 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 0x7d01a3bce700>),
 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 0x7d018418cd60>)]
```

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

tools
```

```
[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 0x7d01a2b52020>),
 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
0x7d01a0daa980>),
 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 0x7d01a3bce700>),
 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 0x7d018418cd60>)]
```

```
# OpenAI API KEY SET KARO (SECURE WAY)
import os
os.environ["OPENAI_API_KEY"] = "PASTE_YOUR_OPENAI_KEY_HERE"
```

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

```
# 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)
```

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

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

🛫 Flight Selected:
  IndiGo | ₹2907

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

📍 Itinerary:
  Day 1: Famous Fort
  Day 2: Popular Museum

🌦 Weather (Max Temp): [22.1, 20.9, 22.2]

💰 Estimated Budget: ₹ 11807
```

```
# 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
    }
```

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

```
{'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': [{'place_id': 'PLC0001',
   'name': 'Famous Fort',
   'city': 'Delhi',
   'type': 'lake',
   'rating': 4.6},
  {'place_id': 'PLC0004',
   'name': 'Popular Museum',
   'city': 'Delhi',
   'type': 'lake',
   'rating': 4.5}],
 'weather': [22.1, 20.9, 22.2],
 'budget': {'flight': 2907,
  'hotel': 7300,
  'food_travel': 1600,
  'total': 11807}}
```

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

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.12/dist-packages (1.52.1)
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) (6.2.2)
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)
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.12/dist-packages (from streamlit) (0.9.1)
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
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->
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-
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->st
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streaml
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit) (2025.
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit) (202
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->stre
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)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->streamlit)
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.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->altair!=5.4.0,!=5.4.1,<7
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=5.4.0,!=5
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (from jsonschema>
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=5.4
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->altair!=5.4.0,!=
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.
Requirement already satisfied: pyngrok in /usr/local/lib/python3.12/dist-packages (7.5.0)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.3)
```

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

```
Streamlit Installed
```

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

```
%%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"])
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"]
)

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)
```

```
Overwriting app.py
```

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

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