
Project Guide: The AI Travel Planner API

Objective: Build a simple web API that plans a trip. A user will send a destination and a number of days, and the API will return a full travel plan, including an itinerary and a packing list.

How it works: FastAPI will handle the web request and response. **LangGraph** will perform the multi-step "thinking" process to generate the travel plan.

Tools: FastAPI, uvicorn, LangGraph, LangChain

Part 1: The Agent's Brain (The LangGraph Workflow) (25 mins)

First, let's build the core logic for the travel planner. This is what will generate the content.

1. **Define the Agent's Memory (State):**
    - Create a `TypedDict` to hold the agent's information. This memory will track the entire planning process. It needs fields for:
        - `destination`: The city or country to visit (`str`).
        - `days`: The length of the trip (`int`).
        - `itinerary`: The day-by-day plan generated by the agent (`str`).
        - `packing_list`: A suggested list of items to pack (`str`).
2. **Create the Agent's Skills (Nodes):**
    - **Skill 1: `create_itinerary` node:**
        - This function takes the agent's current memory (the state) as input.
        - It uses a `ChatModel` (from LangChain) with a prompt like: `"Create a {days}-day travel itinerary for a trip to {destination}."`
        - It saves the LLM's response into the `itinerary` field in the agent's memory.
    - **Skill 2: `suggest_packing_list` node:**
        - This function also takes the memory as input. It will run *after* the itinerary is created.
        - It uses a `ChatModel` with a prompt that uses the itinerary for context: `"Based on this itinerary for {destination}, suggest a packing list: {itinerary}"`
        - It saves the response into the `packing_list` field in the memory.
3. **Build the Workflow (Graph):**
    - Create a `StatefulGraph`.
    - Add your two skills (`create_itinerary` and `suggest_packing_list`) as nodes.
    - Set `create_itinerary` as the starting point.
    - Draw an edge from `create_itinerary` to `suggest_packing_list`.
    - Set `suggest_packing_list` as the end point.
    - Compile the graph to make it runnable. Your agent's brain is now ready!

Part 2: The Public Interface (The FastAPI Endpoint) (15 mins)

Now, let's wrap your LangGraph agent in a web API so people can use it.

1. **Set up FastAPI:**
    - In a Python file (e.g., `main.py`), import `FastAPI`.
    - Create an app instance: `app = FastAPI()`.
2. **Define the Input:**
    - Using **Pydantic's** `BaseModel`, define a class that describes the JSON data your API will accept.
    - Python

```
None
```

```
from pydantic import BaseModel

class TripRequest(BaseModel):
    destination: str
    days: int
```

3. **Create the API Endpoint:**
    - Create a POST endpoint, for example, at `/plan-trip`.
    - This function will accept the `TripRequest` data.
    - Inside the function, you will:
        - Get the `destination` and `days` from the request.
        - Run your compiled **LangGraph** app, giving it the destination and days as the starting input.
        - Return the final memory (state) from the agent. FastAPI will automatically convert this dictionary to a JSON response.

Part 3: Run and Test Your API (5 mins)

Let's see it all work together.

1. **Run the Server:**
    - From your terminal, run your API using `uvicorn`. The command will look like this:
    - `uvicorn main:app --reload`
2. **Test the API:**

- Open your web browser and go to **http://127.0.0.1:8000/docs**.
- FastAPI automatically creates an interactive documentation page. You can use this page to send a test request.
- Try a request with `{"destination": "Tokyo", "days": 5}`.
- You should get back a complete JSON response containing the generated itinerary and packing list! ✈️PACKINGLIST