**Travel Planner AI Assistant**

## **1. Introduction**

Travel planning involves searching through a vast amount of information—destinations, transport, hotels, budgets, and itineraries. Travelers often feel overwhelmed when making decisions.

This project builds an AI-powered Travel Planner using conversational AI that interacts with users in natural language. The assistant can:

* Understand user inputs like *“I want to go to Jaipur for 5 days”*.
* Ask clarifying questions (budget, interests, transport preferences).
* Recommend destinations, transport options (nearest airport/railway), and hotels.
* Generate personalized itineraries.

The system is designed to simulate a human-like travel agent experience.

## **2. Data Sources**

The AI uses LLM-powered reasoning (OpenAI/Perplexity Sonar), not a fixed dataset.

**3. Instructions for assistant**

* Assistant should collect the values for the following features : *Origin City, Destination, Budget, Number of Days, Travel Style and Accommodation Preference*
* The values for key 'Accommodation Preference' should be either 'budget', 'mid-range', or 'luxury' based on the user's preference.
* The values for key 'Travel Style' should be either 'relaxation', 'adventure', 'cultural', or 'mixed' based on the user's preference.
* The values for keys 'Number of Days' and 'Budget' should be a numerical value extracted from the user's response.
* Minimum budget for a trip should be >= 5000.
* If the user provides a range for 'Budget', take the upper limit of the range.
* If the user provides a range for 'Number of Days', take the upper limit of the range.
* If the user provides a budget in words, convert it to numerical value. For example, "1 lakh" should be converted to "100000".

**4. Technical Setup**

* Language Model: Perplexity AI *(sonar-pro)* via *OpenAI* client wrapper.
* Programming Language: Python.
* Interface: Jupyter Notebook (real-time input via *input()*).

**5. Core Components**

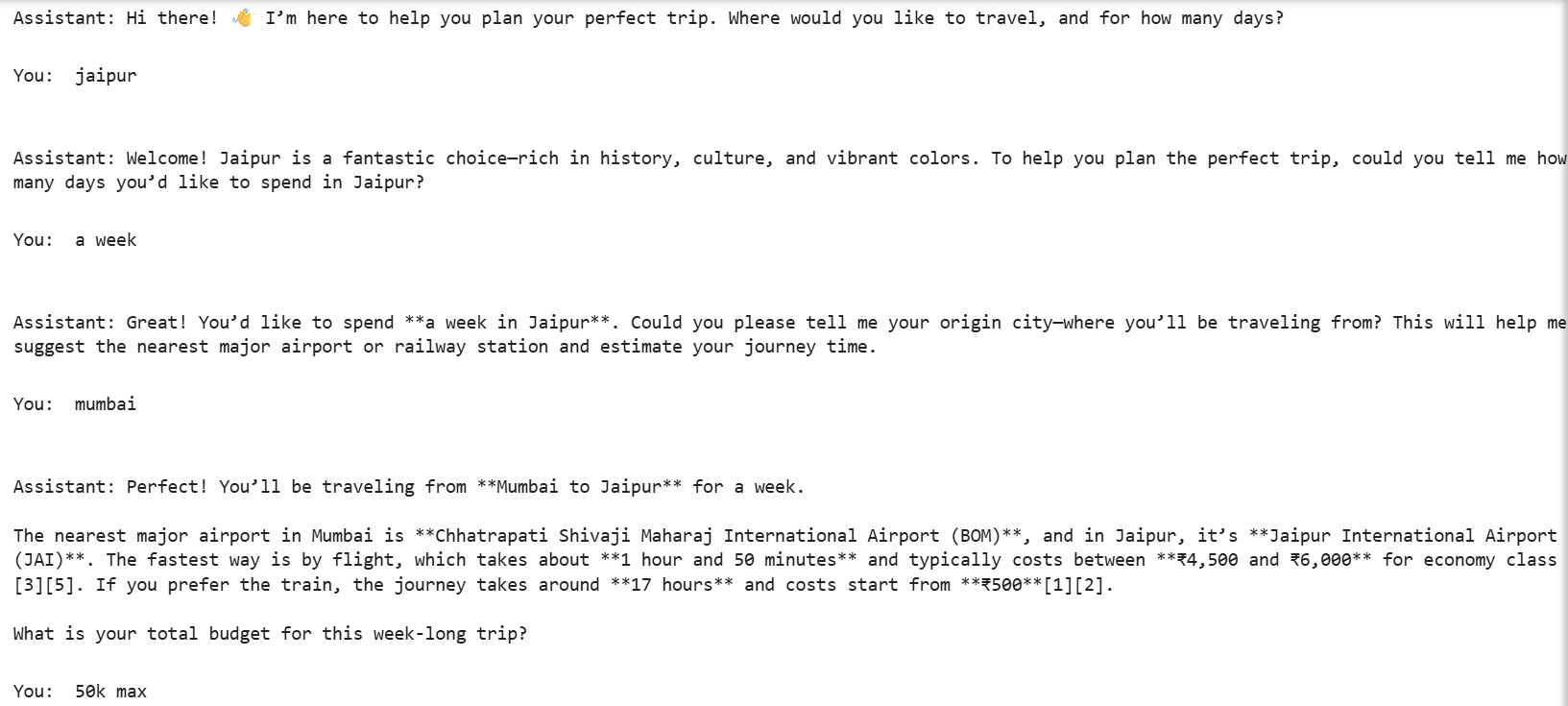
* Dialogue Management System - *dialogue\_mgmt\_system()*
* Conversation Initialization - *initialize\_conversation()*
  + The assistant greets the user and introduces the purpose.
  + Starts with a friendly greeting: *“Hi there! 👋 I’m here to help you plan your next trip. Where would you like to travel, and for how many days?”*
* Chat Completion - *get\_chat\_model\_completions(conversation)*
  + Ensures correct alternation: user → assistant → user → assistant
  + Provides an “exit” option to end the chat gracefully.
* Moderation Layer (optional) *- moderation\_check(response\_assistant)*
  + Screens unsafe/off-topic inputs.
* Intent Confirmation Layer - *intent\_confirmation\_layer\_travel(response\_assistant)*
  + Confirms whether the assistant understood the user’s request.
  + Checks if all information is collected
* Recommendation Engine
  + Suggests destinations, nearest airports/railway stations.
  + Generates detailed itineraries with attractions and hotel options.

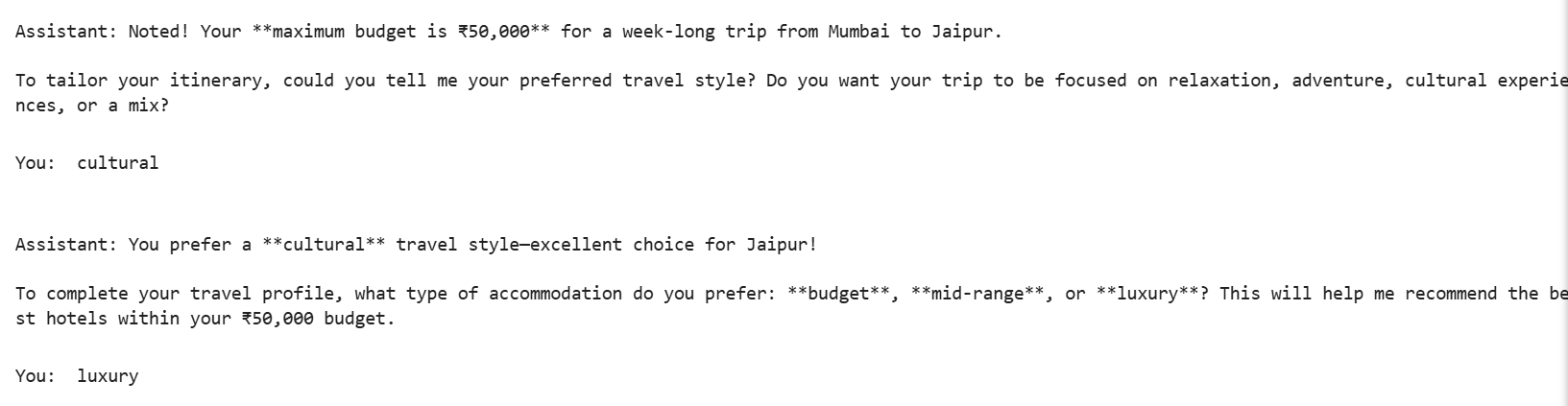
**6. Workflow**

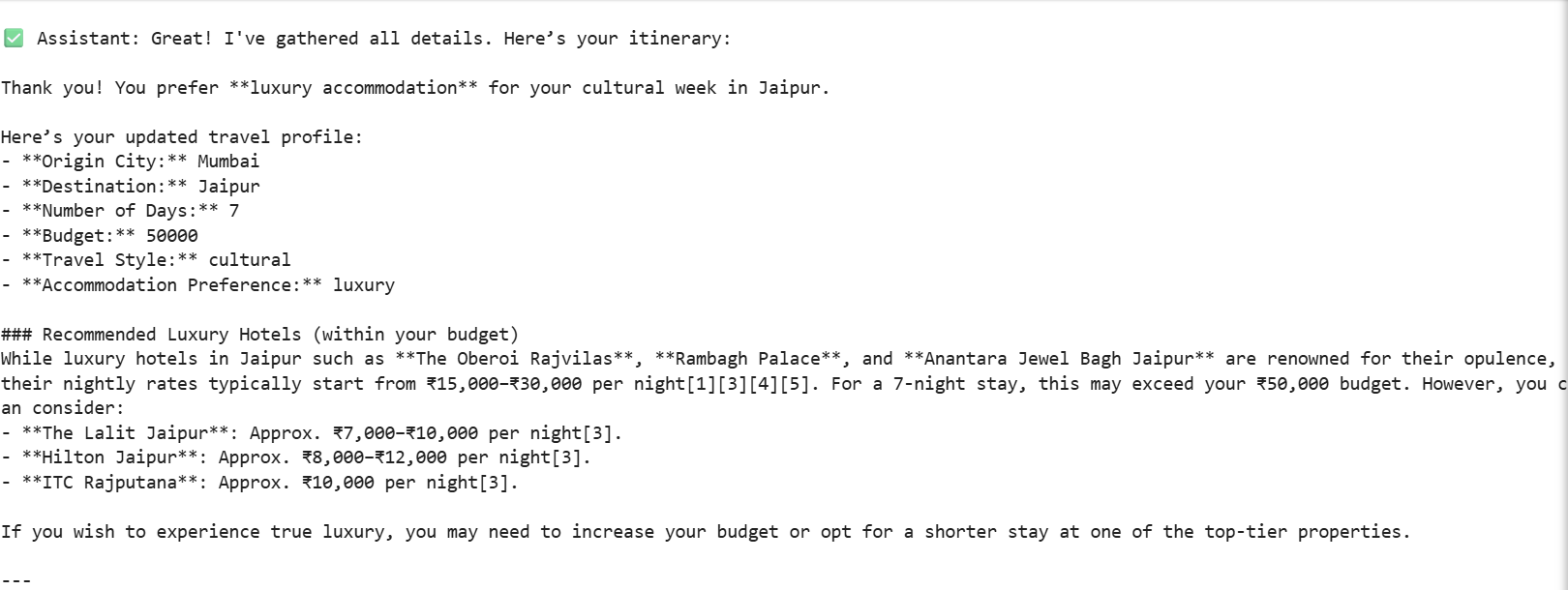
* Conversation Initialization
  + The assistant greets the user and introduces the purpose.
* User Input → Moderation Check
  + If input passes → continue.
  + If input fails → ask user to re-enter valid input (loop back to User Input).
* Assistant Response → Intent Confirmation Layer
  + Checks: Do we have all required information ?
    - No → Ask Next Question → Loop back to User Input.
    - Yes → Proceed to Recommendation Engine.
* Recommendation Engine
  + Suggests itinerary, hotels, travel options.
* Follow up question to continue or Exit Chat
  + Asks a follow up question if the user wants to continue the chat or gracefully ends with goodbye.

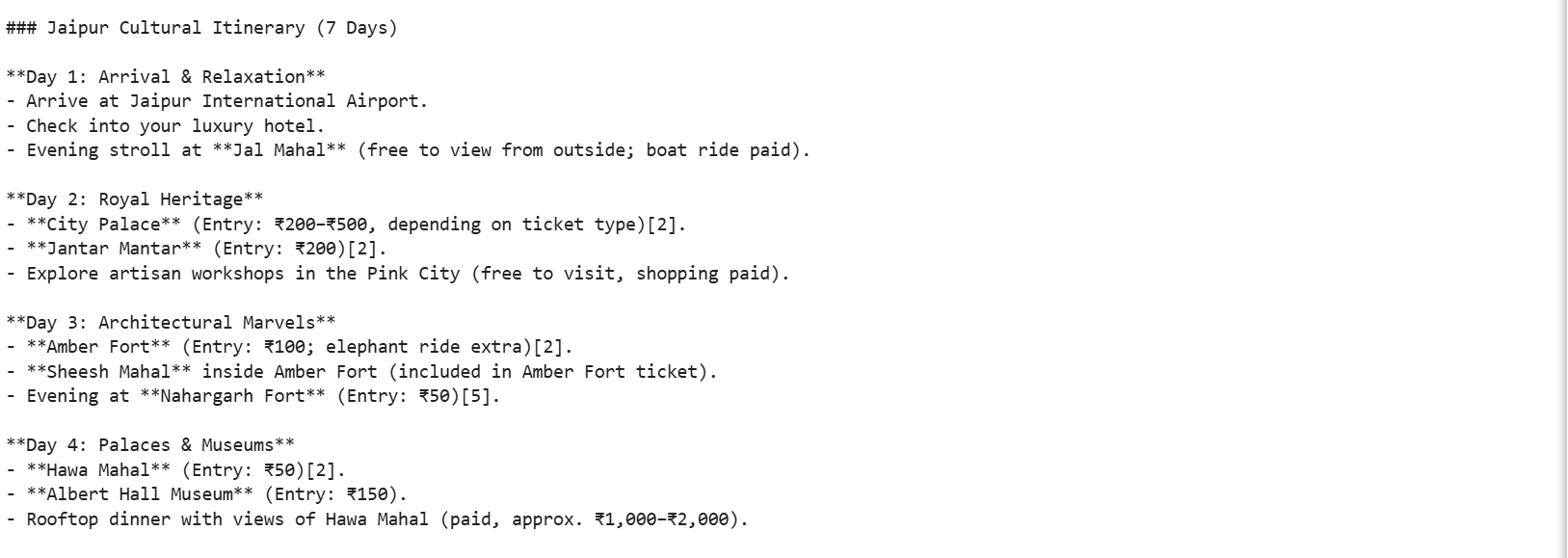
**7. Result**

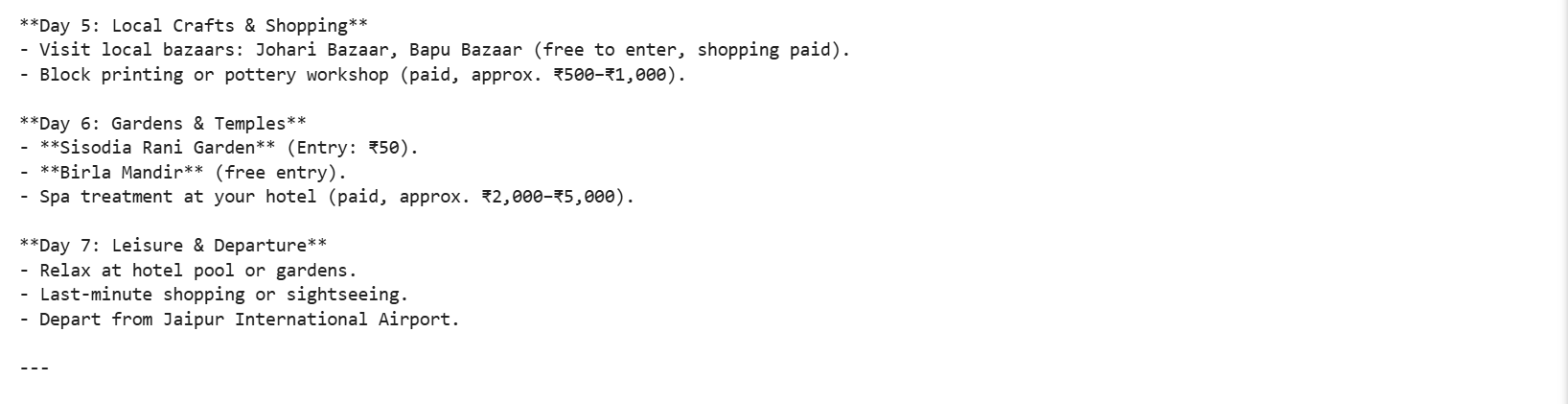
* Working Conversational Agent that:
  + Greets user.
  + Collects travel details.
  + Suggests itineraries, hotels, and transport.
  + Handles user inputs in real-time via console.
* Sample Interaction:

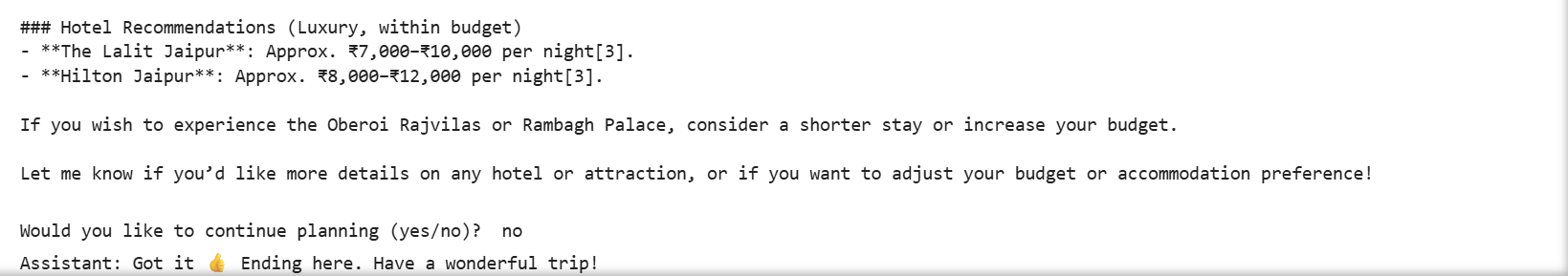












**8. Future Enhancements**

#### Real-time Data Integration

* + Flight & Train APIs: Integrate with Skyscanner, Amadeus, or IRCTC APIs to fetch live prices, timings, and booking links.
  + Hotel APIs: Connect to Booking.com for real-time hotel availability and dynamic pricing.
  + Weather APIs: Suggest itineraries based on real-time weather forecasts (e.g., avoid outdoor attractions on rainy days).

#### Safety & Guidance

* + Visa & Travel Rules: Provide entry requirements, visa info, and COVID-related advisories.
  + Local Emergency Info: Share nearest hospitals, police stations, or embassy contacts.
  + Safety Tips: Suggest safe neighborhoods, common scams to avoid, etc.

#### Advanced Planning

* + Day-wise Budget Tracking: Break down expenses (transport, accommodation, attractions) per day.
  + Multi-City Planning: Allow users to plan multi-stop itineraries (e.g., Delhi → Jaipur → Udaipur).
  + Group Planning: Handle group trips with shared budgets and preferences.