**Background**

The Travel Planner AI chatbot is designed to enhance the travel planning experience for users by providing personalized assistance. It offers recommendations for top hotels and suggests tailored itineraries based on the user's specified destinations. By leveraging advanced algorithms and user preferences, the chatbot aims to simplify the trip planning process, making it more efficient and enjoyable for travelers. (Note:The chatbot will be further enhanced for different search types)

**Objectives**

1. **User Assistance**:
   * Provide users with a conversational interface to assist them in planning their travel, including finding hotels, activities, and flights.
2. **Integration with APIs**:
   * Integrate with the OpenAI API for natural language processing and the Amadeus API for accessing travel-related data.
3. **User-Friendly Interface**:
   * Develop a simple and intuitive web interface that allows users to interact with the chatbot seamlessly.
4. **Dynamic Responses**:
   * Ensure that the chatbot can generate dynamic responses based on user input and preferences.
5. **Profile Management**:
   * Implement a system to manage user profiles and preferences, allowing for personalized interactions.

**Approach to Building the Travel Planner AI Chatbot**

**1. Define Requirements**

* Identify the main functionalities of the chatbot:
  + Searching for hotels.
  + Finding activities in a destination.
  + Searching for flights.
* Determine how the chatbot will interact with users and respond to various queries.

**2. Set Up the Development Environment**

* Install necessary libraries:
  + Flask for the web application.
  + OpenAI SDK for integrating ChatGPT.
  + Amadeus SDK for accessing travel-related APIs.
* Create a virtual environment to manage dependencies.

**3. Design the Architecture**

* Components:
  + User Interface (UI): Built using HTML and JavaScript, allowing users to interact with the chatbot.
  + Flask Web Server (app.py): Handles incoming requests and responses.
  + OpenAI API: Provides natural language processing capabilities for generating responses.
  + Amadeus API (travel\_api.py): Fetches travel-related data (hotels, flights, activities).
  + User Profiles (user\_profiles.py): Manages user preferences and past interactions.
  + Business Logic (functions.py): Contains functions for processing user input and interacting with APIs.

1. **Implement the Backend Logic**

app.py:

* 1. Set up Flask routes to serve the HTML page and handle user queries.
  2. Define a route for the main page (/) and a route for processing user input (/ask).

user\_profiles.py:

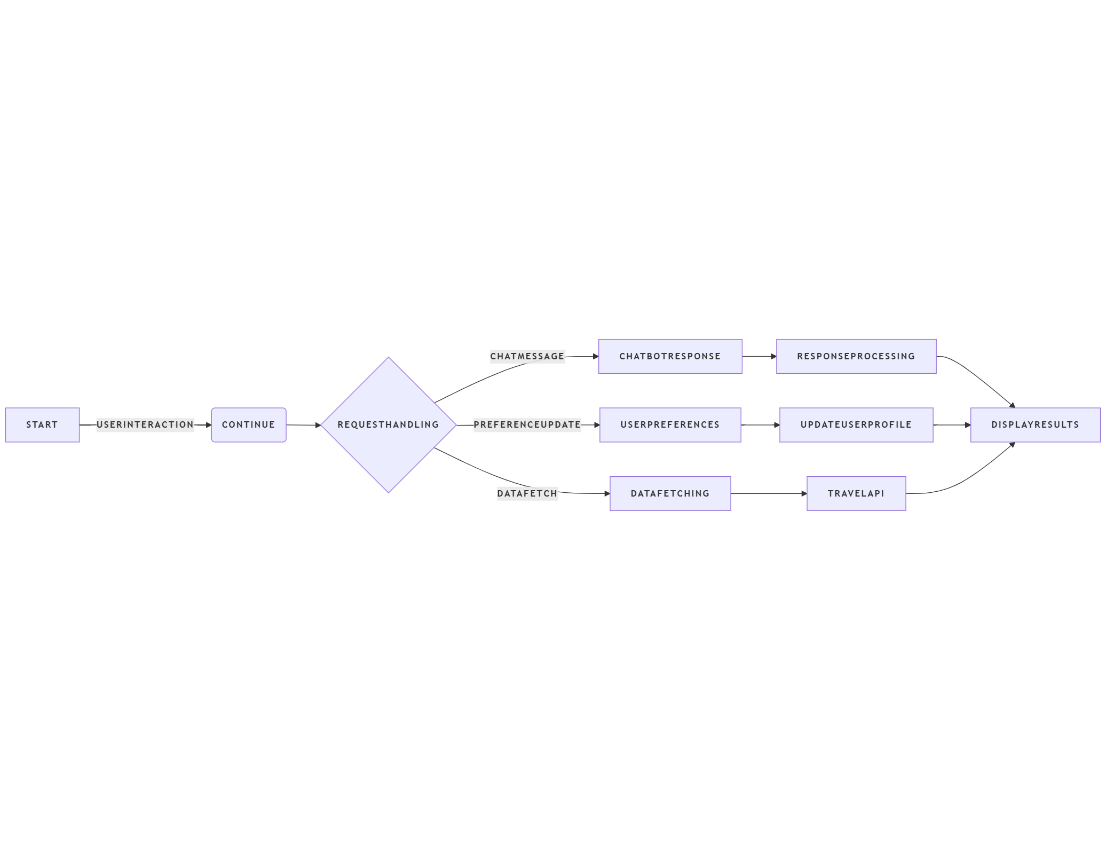
* 1. Implement functions to manage user profiles, preferences, and interactions.

functions.py:

* 1. Implement the main logic for processing user input and interacting with the APIs.
  2. Use the OpenAI API to generate activity suggestions.

1. **Create the User Interface**

* conversation\_bot.html:
  + Develop a simple web interface using HTML and JavaScript to allow users to interact with the chatbot.

**7. System Architecture**

**8.Prerequisites**

**Tech Stack for Travel Planner AI Chatbot**

**1**. Frontend Technologies

* HTML: Used for structuring the web pages and creating the user interface elements.
* CSS: Used for styling the web pages, making the chatbot visually appealing and user-friendly.
* JavaScript: Used for adding interactivity to the web pages, handling user input, and making asynchronous requests to the backend.
* jQuery: A JavaScript library that simplifies DOM manipulation and AJAX requests, making it easier to handle user interactions.

2. Backend Technologies

* Python: The primary programming language used for developing the backend logic of the chatbot.
* Flask: A lightweight web framework for Python that is used to create the web server, handle routing, and manage requests and responses.
* OpenAI API: Used to integrate ChatGPT for natural language processing, allowing the chatbot to understand user queries and generate responses.
* Amadeus API: A travel API used to fetch travel-related data, such as hotels, flights, and activities. The Amadeus SDK for Python simplifies the process of making API calls.

**9.Challenges**

1. **API Integration**:
   * Integrating multiple APIs (OpenAI and Amadeus) posed challenges in terms of authentication, handling API responses, and managing rate limits.
2. **Data Handling**:
   * Ensuring that the data returned from the APIs was correctly parsed and formatted for user-friendly output required careful handling of JSON responses.
3. **User Input Parsing**:
   * Developing robust functions to extract relevant information (like origin and destination) from user input was challenging, especially with varied phrasing.
4. **Error Handling**:
   * Implementing effective error handling for API calls to manage failures gracefully and provide meaningful feedback to users.

**10. Lessons Learned**

1. **Importance of API Documentation**:
   * Thoroughly reviewing API documentation is crucial for understanding the available endpoints, required parameters, and response formats.
2. **User-Centric Design**:
   * Focusing on user experience during the design phase leads to a more intuitive interface and better user satisfaction.

**10. Appendix - B**

User Output example screenshot

