AI-Powered Travel Planner - Documentation

Project Overview

The AI-Powered Travel Planner is a web-based application designed to provide users with detailed travel information using artificial intelligence.
The app allows users to enter a starting location (source) and a destination to receive structured travel details, including different modes of transport,
estimated travel time, costs, and company-specific recommendations. Additionally, it offers alternative travel routes and tips to enhance the travel experience.

Technologies & Libraries Used

1. Streamlit

Used to build the interactive web application, providing an easy-to-use interface where users can input their travel details and view AI-generated results.

2. LangChain & Google Gemini AI

LangChain is integrated with Google's Gemini AI model to process and generate travel recommendations. It structures queries and ensures that responses are formatted in a meaningful way.

3. Pillow (PIL)

Used for image processing, such as displaying travel-related images in the application interface.

4. OS Module

Used to fetch environment variables, particularly the API key required to authenticate with Google Gemini AI.

Step-by-Step Development Process

Step 1: Setting Up the Development Environment

- Install the necessary Python libraries: Streamlit, LangChain, LangChain-Google-GenAI, and Pillow.

- Set up the GOOGLE_API_KEY environment variable, which is essential for authenticating API requests to Google Gemini AI.
- Prepare a project directory with the necessary files, including an image for branding (if available).

Step 2: Designing the User Interface

- Create a web-based interface where users can input their source and destination.
- Display a header and relevant images to enhance the visual appeal.
- Use Streamlit components to take user input in a structured format.

Step 3: Integrating Google Gemini AI for Travel Data

- Configure the AI model to process travel-related queries.
- Define a structured prompt to ensure the AI provides useful and relevant travel information.
- Structure the response to include a table of transport options followed by a detailed travel guide.

Step 4: Handling User Inputs and AI Responses

- Validate the user input to ensure source and destination fields are filled correctly.
- If the API key is missing, display an error message prompting the user to set it up.
- Process the AI-generated response and display the structured travel details in the app.

Step 5: Deploying the Application on Hugging Face Spaces

- Create a `requirements.txt` file listing all dependencies needed for the project.
- Push the project files to a GitHub or Hugging Face repository.
- Deploy the application on Hugging Face Spaces and test it.

Step 6: Testing and Debugging

- Check for potential issues such as missing API keys or package installation errors.
- Ensure the AI returns structured and relevant travel information.
- Verify that the application interface is responsive and user-friendly.

Expected Functionality

- The user inputs a source and destination.
- The AI generates a structured table with available transport options, including costs and estimated travel times.
- A detailed travel guide follows, explaining the best routes, alternative travel options, and travel tips.

Future Enhancements

- Integrate real-time travel pricing from APIs such as Google Flights, IRCTC, or Uber.
- Add user preferences for budget-friendly, fastest, or luxury travel recommendations.
- Enhance the AI model to support multi-destination travel planning.

Conclusion

By following this structured approach, the AI-Powered Travel Planner ensures users receive accurate, efficient, and well-organized travel recommendations, making trip planning easier and more informed.