ECE:5995 - GAIT
Brandon Cano
Meesam Amir Syed
Kasra Shahrivar
Rutger Schleyhahn

AI-Powered Personalized Travel Itinerary Generator

Travel planning can be a complex and time-consuming process, often involving extensive research across multiple platforms to find the best destinations, activities, and accommodations that align with personal preferences and budgets. Traditional methods of trip planning—such as reading blogs, consulting travel agencies, or relying on generic guides—can lack personalization and fail to capture the unique needs and interests of individual travelers. Moreover, visualizing a destination or activity before committing to a trip remains a challenge for many, leaving travelers uncertain about their choices. This gap in accessible, tailored, and immersive planning tools inspired the development of the AI-Powered Personalized Travel Itinerary Generator.

The AI-Powered Personalized Travel Itinerary Generator is a Flask-based web application that integrates multiple AI APIs to deliver immersive and personalized travel planning. OpenAI's GPT-4o-mini generates detailed travel itineraries, must-see attractions, and descriptive prompts, which are passed to OpenAI's DALL·E-3 for high-quality image generation. These images are processed by RunwayML Gen-3 Turbo to produce drone-like "hover" videos, simulating virtual tours for each attraction. Videos are then combined into a single cohesive clip using FFmpeg for backend video merging. The system dynamically handles user inputs (destination and travel dates), incorporates weather-based suggestions, and saves all outputs (images, videos, PDFs) locally for presentation. The frontend, built with HTML, CSS, and Jinja templating, offers an interactive UI where users can explore itineraries, preview videos, and download results.

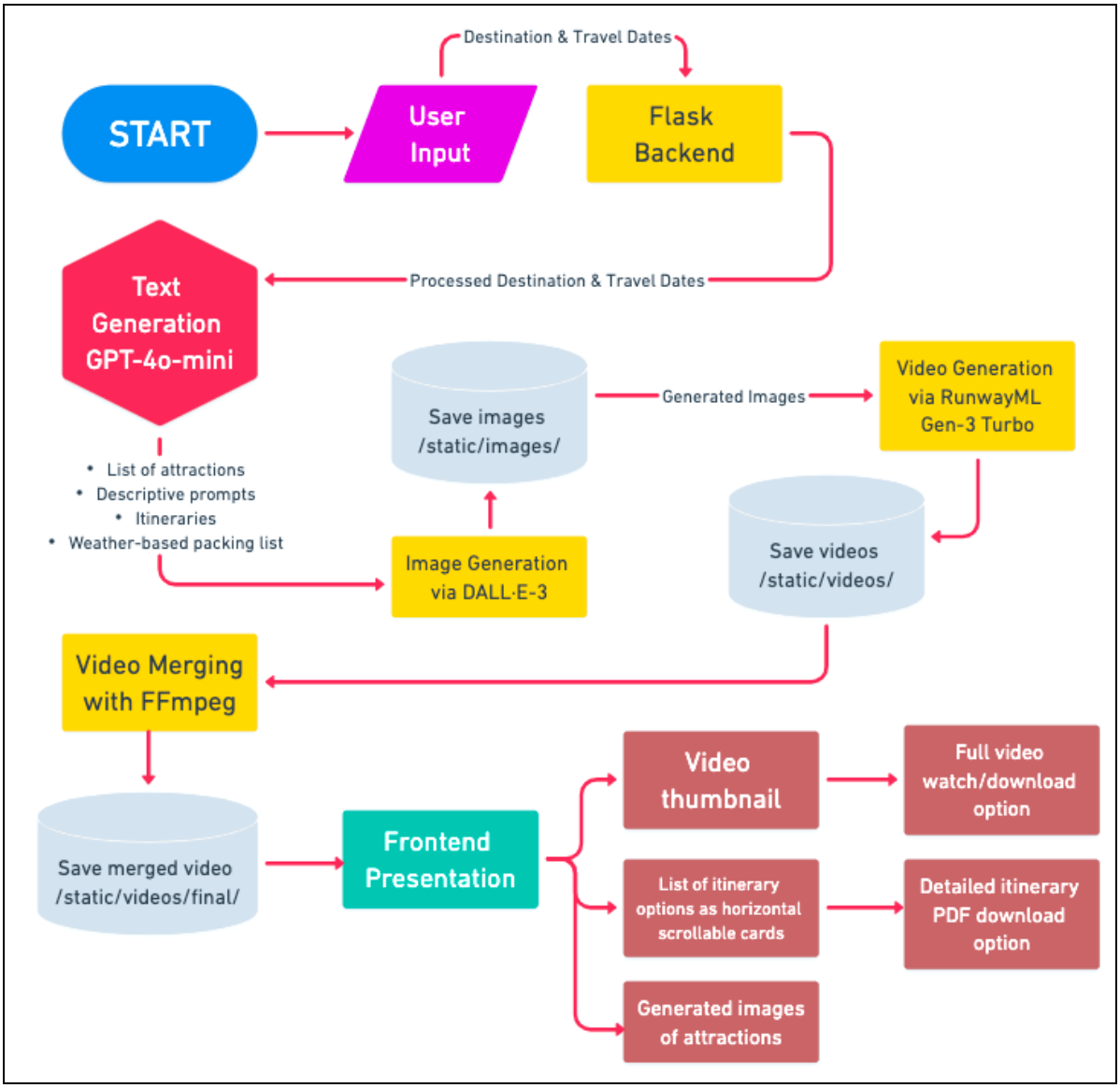

Figure 1: Workflow

We used two OpenAI API models for generation, ‘gpt-4o-mini’ for text generation and ‘dall-e-3’ for our image generation. We used both of these together since gpt-4o-mini was really helpful for our text generation when it came to considering a lot of different variables for suggestions, then the same idea was used for when we used dall-e-3 for the image generation for the different suggestions that we had. For video generation we used the RunwayML API, specifically using the ‘gen3a_turbo’ model. We used this because it was able to take both text and images as an input and able to take the image and make our ‘drone-like’ hovering movements for each image. Additionally, we used this because it was the fastest video generation model available (or that we could find) since we could have up to 5 separate videos generated back to back within the flow of the program.

To start the workflow of the project we used ChatGPT in order to create the outline based on our project description and how we decided we wanted to format it. This gave us our Flask outline with some basic HTML pages to start working off of. The Python file it gave us also listed out plenty of detailed TODOs for doing API calls or when mock data was being used instead. From there each of us went into our own tasks that were a bit isolated from the main code writing test scripts for the different elements of the project, like testing video and image generation, prompting, and then also using ffmpeg to combine videos together. ChatGPT was being used to help debug, create good prompts, and help get our code working. After the test scripts worked we then started adding them inside of the TODOs that we had initially. Then we had to go through and debug each section so that we could glue all of them together, and ChatGPT really helped with this. This was used to generate prompts for image generation in the midst of the project and to ensure that the images can be converted into videos.

For this project each of us did the following work. Brandon created the UI of the project and figured out how to do video generation API calls with images and text as the input. He also connected the pieces between the different screens and parameters. Kasra researched how to use the ffmpeg program in order for us to edit videos together and then wrote the function to be able to do that. He also helped get image generation working with OpenAIs API. Meesam got the initial project setup with Flask and created the general structure of the project. He also was working with OpenAIs API to get text generation for the attractions and suggestions for the initial input of the user. He also helped get image generation working. Rutger was working with OpenAIs API for text generation in order to get weather information as well as creating the itinerary and getting other suggested items for the trip details, like items to pack.

One big issue we ran into was the wait time with video generation since initially we tried using a model from Replicate which took 2-4 minutes for a 5 second video, but to address this we were recommended to use Runways API instead which was about 1 minute or so per 5 second video. We still had to wait a few minutes but it was a much faster solution than before.

An ethical consideration could have to do with the image/video generation since one it’s not entirely accurate and two when it's closer to an art style than realistic images than we have the issue of “stealing” art in some aspects especially if this were to ever be a commercial product.

If we were to create this project again in the next few years from now, we could likely make it a whole lot better. Video and image generation from the APIs will be faster, which can reduce the multi-minute wait times into mere seconds. As well as we can get longer videos for cheaper and likely more accurate representations of the

different attractions for a location we want to visit. This process could also likely all be simplified further and just have a text to speech explaining everything you want in detail and letting the generations come up with the proper itineraries, images, videos, suggestions, etc.