Python Itinerary Generator Checklist

Build a standalone AI-powered travel planner with Python, Cursor, and Gemini 2.5 Flash.

Based on YouTube Video cNigjD-GZfI by Casey Builds It

Watch the video: *https://www.youtube.com/watch?v=cNigjD-GZfI*

---

1. Project Setup and Core Features

Define the foundational elements and desired functionalities for your itinerary generator.

- ■ Pull reference material from an existing website (e.g., Airbnb feature).
- ■ Feed example data into Cursor for AI context.
- ■ Define core features: date selection, guest counts (adults, kids, dogs).
- ■ Incorporate budget preferences.
- ■ Add functionality for custom activity requests.
- ■ Implement PDF export functionality.

2. Location-Based Data Scraping

Configure the tool to dynamically pull location-specific information.

- ■ Test the tool with an initial location (e.g., Colorado Springs).
- ■ Test with different locations (e.g., Portland, Oregon; Portland, Maine).
- ■ Scrape real businesses, restaurants, and activities.
- ■ Utilize sources like OpenStreetMap, Wikipedia, and DuckDuckGo.
- ■ Debug issues with location scraping.
- ■ Consider pros and cons of free scraping vs. paid APIs (e.g., Google Places).

3. AI Integration and API Keys

Connect with AI models and manage necessary API credentials.

- ■ Use Cursor for AI-assisted coding.
- ■ Integrate Google Gemini 2.5 Flash.
- ■ Input your Google API key into the tool.
- ■ Understand API cost considerations (Gemini Flash vs. Claude).
- ■ Generate and save API keys securely.

4. Building the Python Application

Follow the step-by-step coding process to construct the itinerary generator.

- ■ Define the area for the build logic.
- ■ Implement area-based logic for data retrieval.
- ■ Start building the Python script in Cursor.

- [ ] Save the current state of the build.
- [ ] Explore Python GUI tool development.
- [ ] Implement conditional logic for itinerary generation.

5. Data Handling and Storage

Manage how data is extracted, stored, and retrieved within the application.

- [ ] Extract relevant data for itinerary planning.
- [ ] Store extracted data in JSON format.
- [ ] Implement logic to handle area and time inputs.
- [ ] Consider saving scraped data to a database.
- [ ] Implement search functionality for stored data.
- [ ] Cache previously scraped areas for faster retrieval.

6. Enhancing Itinerary Features

Add advanced functionalities to make the itinerary more comprehensive and useful.

- [ ] Integrate weather API for forecasts.
- [ ] Implement GPS-based distance filtering.
- [ ] Incorporate event APIs for concerts and local happenings.
- [ ] Display event listings dynamically.
- [ ] Add address information to generated plans.
- [ ] Calculate travel distance between locations.

7. Output and Refinement

Generate the final itinerary and review the results.

- [ ] Generate the itinerary based on user inputs.
- [ ] Save the itinerary as a PDF output.
- [ ] Review the generated itinerary for accuracy and completeness.
- [ ] Check arrival details and planned activities.
- [ ] Make necessary changes or refinements to the plan.
- [ ] Identify local activities and points of interest.

8. Scaling and Future Development

Consider how to expand the tool's capabilities and potential applications.

- [ ] Explore statewide itinerary generation potential.
- [ ] Consider using the tool for personal projects or business applications.
- [ ] Discuss options for pre-known data integration.
- [ ] Explore adding more options and features.
- [ ] Consider how the tool could scale into a full web application.
- [ ] Think about other API ideas for further integration.

Resources & Further Reading

Use these pages while working through the checklist:

Casey Builds It | -
- | caseybuildsit.com
Build practical AI tools with Python and Cursor for dynamic travel planning.