APP DESIGN: JUST TRAVEL

Box 1: Initial User Questionnaire

- ❑ **Trip Logistics:** Define the destination(s), duration of stay, and whether multiple destinations are planned.
- ❑ **Financials:** Input the total budget for the trip.
- ❑ **Dietary Profiles:** Selection of diet types such as Vegan, Carnivore, or Halal (Mandatory field).
- ❑ **Meal Preferences:** Choose which specific meals to eat out: Breakfast, Lunch, Dinner, Snacks, or None (Mandatory field).
- ❑ **Religious Preferences:** Optional field for any religious requirements.
- ❑ **Flight Preferences:** Preference for layovers (escales) and the maximum number of hours acceptable for a layover.

Box 2: Smart Transportation Routing

- ❑ **Arrival Options:** Identify all available means of transport to reach the requested destination.
- ❑ **Cost Efficiency:** Filter for the cheapest transportation options.
- ❑ **Time Efficiency:** Calculate and provide the shortest path to the destination.

Box 3: Budget-Aligned Accommodations

- ❑ **Hotel Recommendations:** Suggest lodging options that fit strictly within the user's budget.
- ❑ **Real-Time Status:** Verify and display the current availability of these hotels.

Box 4 :Personalized Food & Dining

- ❑ **Local Cuisine:** Provide food recommendations focusing on authentic local dishes.
- ❑ **Availability & Cost:** Cross-reference all dining options with the user's budget and check current availability.

Box 5: Exploration & AI "Hidden Gems"

- ❑ **Tourist Landmarks:** Identify standard touristy locations.
- ❑ **Social AI Agent:** Use an AI agent to scan Instagram and TikTok videos from locals to find "hidden gems

Box 6: Dynamic Daily Planner

- ❑ **Unified Itinerary:** Organize all selected components into a comprehensive plan for each day of the stay.
- ❑ **User Flexibility:** Allow the customer to manually change or adjust the planner as they wish.

Like wa7sh rinconi's doc

1. Core System Architecture

To achieve the dynamic and personalized experience outlined in your plan, the app will utilize a **Microservices Architecture** to ensure each "box" operates independently yet stays synchronized.

- **Frontend Layer:** Developed in **Next.js** or **Flutter** to support a multi-step conversational questionnaire and dynamic itinerary rendering.
- **Orchestration Layer:** A **Node.js or Python (FastAPI)** backend to manage user state, coordinate API calls, and handle the logic for "Mandatory vs. Optional" fields.
- **Database Schema:**
    - **PostgreSQL:** For structured user profiles (budgets, dietary/religious restrictions, and booking history).
    - **Neo4j (Graph Database):** Essential for "Smart Transportation Routing" to calculate shortest/cheapest paths across multi-modal transport networks.

2. Technical Execution of boxes:

A. Smart Questionnaire & Data Ingestion (Box 1)

- **Logic:** Implement a state-managed form (using Redux or React Context) where certain fields (Budget, Diet, Meals) are flagged as **`isRequired: true`** while Religion is **`false`**.
- **Flight Preferences Module:** Logic to filter flight API results based on user-defined "Maximum Layover Hours".

B. Routing & Logistics Engine (Box 2)

- **Multi-Modal Integration:** Utilize APIs like **Amadeus** or **Skyscanner** for flights and **Google Directions API** for local transit.
- **Optimization Algorithm:** Implement a **Dijkstra-based algorithm** or A* search on the graph database to find the intersection of "Cheapest" and "Shortest" routes.

C. Budget-Aligned Accommodations & Dining (Box 3 & 4)

- **Filtering Engine:** A backend service that takes the total budget and divides it per day. It queries the **Booking.com API, airbnb API** or **Expedia API** with a `price_max` parameter.
- **Dietary Compliance:** Use **Yelp or Google Places API** to fetch restaurant data. An AI filter (using a Large Language Model) will cross-reference the `dietary_preference` (e.g., Vegan) with the restaurant's menu tags.

D. AI "Hidden Gems" Scraper (Box 5)

- **Social AI Agent:** Use an AI agent (built with **LangChain**) to scrape and analyze data from social platforms.
    - **Tech Stack:** Use tools like **Apify** for Instagram/TikTok scraping and **OpenAI's GPT-4** to perform "Sentiment Analysis" and "Popularity Scoring" to identify trending spots that aren't yet "tourist traps".
    - **POI Database:** Store these results in a vector database (like **Pinecone**) for fast semantic retrieval based on user vibe.

E. Dynamic Planner & User Overrides (Box 6)

- **Itinerary Generator:** Use **LangGraph** to construct a sequence of activities (Breakfast -> Transit -> Attraction -> Lunch) while ensuring geographical proximity to minimize travel time.
- **Manual Overrides:** Implement a "drag-and-drop" interface (e.g., **React DnD**) that allows users to swap tasks. Each manual change triggers a "Re-Optimization" call to adjust the rest of the day's schedule based on the new time/location.

3. Success Metrics & Deployment

- **Technical Goal:** Itinerary generation in **<5 seconds**.
- **Personalization Accuracy:** 90% match rate between user dietary preferences and suggested dining.
- **Infrastructure:** Deploy on **AWS (Lambda & ECS)** for automatic scaling during peak travel seasons.