Version 01: Add Places, Routes and Create Shareable Itineraries

A user who wants to plan their travel on an interactive map interface - opens this application, adds places by their names or address. Further the user confirms routes between each pair of places to create an itinerary. The user can add personal and optional notes to places and routes as required. Post the places and routes are confirmed, the user can generate a shareable link to make the itinerary viewable to all that can access it through the link.

This feature as a standalone product allows users an option other than Google Maps to plan their travel itinerary. Users can add notes to places and plan and confirm each route individually, unlike Google Maps. This feature as a standalone would be targeting a niche user persona of travel planners - unlike google maps which is a generic app.

Flow

1. a user visiting the website, on loading, can see a vector map loaded - the user can play around by zooming and panning
2. the user is able to click on a bar to add places on this map interface. On clicking Add, the places are added as markers onto the map.
3. the user should be able to click on these places to remove them or add personal notes to them.
4. the user should be able to access a 'Plan Route' option from the UI and select any two place markers to plan a route between them. The user is displayed multiple routes from which they can confirm one.
5. the user can click on any added route to add personal notes or remove those routes
7. the user should be able to save a map post completely adding all places and routes or midway. A saved map can be accessed later for completion.
8. the user can access the saved maps through a CTA on UI over the vector map interface. At any point of time a user loads the website they can add new places and routes to create a new map or open a saved map.
9. on clicking saved map icon, the user can select any of their saved maps to view the already added places and routes
10. the user should be able to generate a shareable link for the saved maps, this link can be shared to anyone who can open this map with all places and routes visible in view-only mode
12. post a map is saved, the user should receive a popup to share their feedback. Once a user has already given a feedback, then the same popup should not be visible to them again

Requirements

Rules

1. Use Next JS + Tailwind for frontend
2. This version of application is serverless - as in IndexedDB to be used for storage and retrieval
3. Develop a PWA so that it can run on web and can be installed in mobile devices
4. Ensure that the UI/UX experience is good on desktop as well as mobile screens

5. Use Ola Maps for Mapping use cases - use Ola Maps Website SDK for rendering vector maps, managing markers, and events. Separate Ola Maps APIs are available for Directions, Routes as specified in detail in the below table (feature description and tech stack).
6. **Security**: Use Next.js API routes as a proxy to hide API keys. Process geocoding/routing requests server-side.
7. **Storage Method**
    a. Use IndexedDB via LocalForage to store maps, routes, and places without a backend.
    b. IndexedDB allows structured storage, making it faster than LocalStorage for large datasets.
8. **User Session Management (No Authentication)**
    a. Use localStorage or a persistent browser fingerprint (e.g., uuid stored in localStorage) to retain maps across sessions.
    b. Generate UUID on first visit using crypto.randomUUID() and store in localStorage
    c. Store uuid in `maps` and `feedback` tables to differentiate users.
9. **Map Saving & Loading**
    a. Save maps as JSON in IndexedDB.
    b. Retrieve and display them using React state
10. **Shareable Links**
    a. Store `map_id` in the URL, encode the entire map data as a compressed JSON string in the URL hash (e.g., yourdomain.com/map/#<encoded_data>
    b. Fetch map details using `map_id` when the link is opened.
11. **Routing Optimization**
    a. Use Ola Maps Routing API to fetch route data and store it in JSON format inside the `route_data` column.
12. **Error Handling**
    a. Display Toast Message if an API fails, return the message from the API
    b. Warn users when approaching IndexedDB storage quotas

Feature Description and Tech Stack

Feature	Feature Description and Technical Documents	Tech Stack
1. Home Page - Render Vector Map: Zooming & panning functionalities	As soon as the website is loaded, render a map with Pune as the default location. User should be able to zoom and pan this map. Define map style: Ola Vector Map Tiles Setup and initialize Ola Maps SDK to render vector styles: Ola Maps Web SDK	Use Ola Maps WebSDK to render Vector Maps

2. Add Places	Display a bar on the top right for users to add places. Users enter the name or address of the place and click "Add Place" - use geocoding based on place name: Ola Geocoding API. On clicking "Add Place," the place should be added as a marker in the map, and the map should route to this location. Add a default marker: Ola Maps Marker API. If multiple places exist with the same name or address, allow the user to confirm one by clicking the place marker to see a confirm CTA. If only one place is found, confirm by default. Save places in IndexedDB only when confirmed.	Ola Geocoding API + IndexedDB for storing confirmed places
3. Click on markers to add user notes	Click event on a place marker within the map: Ola Maps Events. Users can remove the place marker or add a personal note to the place.	Ola Maps Events + React State + IndexedDB for storing notes and managing remove operations
4. Add route between two points	Use Ola Directions API with waypoints. Users should be able to select a "Route Tool" on the map overlay, then select any two place markers to create a route.	Ola Maps Routing API + Next.js + Tailwind for overlay CTAs
5. Identify and confirm routes	Use event to identify route click: Ola WebSDK Events. If there is only one route, confirm it by default; otherwise, the user can select one route and confirm via a CTA. Once a route is confirmed, the user can click on any route to get a remove option. On clicking "Remove," the route should not be visible on the map. Save route details in IndexedDB only post-confirmation.	IndexedDB (localForage) + React State + Ola WebSDK Events

6. Add Notes or Remove routes	Allow users to enter personal notes. Click on a route to reveal a "Remove" CTA in the description panel. On clicking, the route is removed. Users can recreate a new route between those two place markers.	IndexedDB (localForage) + React State
7. Save the map with all routes added	A saved map includes all added place markers and confirmed routes, along with any user-added notes. Save maps against the user session (avoid authentication flow).	IndexedDB (localForage) + React State
8. Prompt user for feedback after saving	Identify users based on browser sessions. Only ask for feedback when a map is successfully saved. Users can close the modal without sharing. If a user has already shared feedback, do not ask again during future saves.	React Modal + Simple Form. Store feedback via IndexedDB
9. Share map via link and View shared maps	Users should be able to one-click copy a link to share the map. On link click, anyone can view the map but not edit it.	Next.js API Route + TinyURL for shortening
10. Option to View Saved Maps	A "View Saved Maps" CTA should be available on the map interface. Clicking on it should route users to a list of saved maps. Only maps from the same browser session should be accessible. Identify the user via UUID in `localStorage` or a persistent browser ID. Clicking a saved map allows users to view places and routes and make modifications.	Next.js + Tailwind for UI

11. Show PWA install popup	Allow user to install the PWA through a popup once the webpage is ready to be downloaded. - Listen for the beforeinstallprompt event (fired when PWA is installable). - Show a CTA/button (e.g., "Install App"). - Trigger the installation prompt when the user clicks.	beforeinstallprompt

IndexedDB Schema

```
-- Table to store maps created by the user

CREATE TABLE maps (
id TEXT PRIMARY KEY, -- Unique identifier (UUID or NanoID)
name TEXT NOT NULL, -- User-defined map name created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP, -- Timestamp of creation
uuid TEXT NOT NULL -- user-based identifier (no login required) );

-- Table to store places within a map

CREATE TABLE places (
id INTEGER PRIMARY KEY AUTOINCREMENT,
map_id TEXT NOT NULL, -- Foreign key to maps table
name TEXT NOT NULL, -- Name of the place
latitude REAL NOT NULL, -- Latitude coordinate
longitude REAL NOT NULL, -- Longitude coordinate
user_notes TEXT, -- Optional user-added notes
FOREIGN KEY (map_id) REFERENCES maps(id) ON DELETE CASCADE );

-- Table to store routes created within a map

CREATE TABLE routes (
id INTEGER PRIMARY KEY AUTOINCREMENT,
map_id TEXT NOT NULL, -- Foreign key to maps table
start_place_id INTEGER NOT NULL,-- Foreign key to places table (starting point)
end_place_id INTEGER NOT NULL, -- Foreign key to places table (ending point) route_data
TEXT NOT NULL, -- Serialized JSON data for route details (path, distance, time)
user_notes TEXT, -- Optional user-added notes
FOREIGN KEY (map_id) REFERENCES maps(id) ON DELETE CASCADE,
FOREIGN KEY (start_place_id) REFERENCES places(id) ON DELETE CASCADE,
FOREIGN KEY (end_place_id) REFERENCES places(id) ON DELETE CASCADE );

-- Table to store user feedback

CREATE TABLE feedback (
id INTEGER PRIMARY KEY AUTOINCREMENT,
uuid TEXT NOT NULL, -- Identify user without login
map_id TEXT NOT NULL, -- Map associated with feedback
feedback_text TEXT, -- User feedback
submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (map_id) REFERENCES maps(id) ON DELETE CASCADE );

```