LankaRouteWatch Documentation

1. Project Overview

LankaRouteWatch is a real-time, community-driven web application designed to monitor road conditions across Sri Lanka. It helps travelers identify blocked routes (due to floods, landslides, etc.) and find alternative paths. The platform covers major highways and over 100 destinations connecting Colombo to the rest of the island.

Key Capabilities

- **Route Monitoring:** Visualizes status (Clear, Warning, Blocked) for major routes like A1 (Kandy), A2 (Galle), A4 (High Level), etc.
- **Community Verification:** Users can upvote "Road is Clear" or "Road Blocked" to provide real-time validation.
- **Alternative Routing:** Users can draw and submit detour paths on the map if a main segment is blocked.
- **Smart Search:** Allows users to find the best route by searching for starting points and destinations (e.g., "Colombo" to "Jaffna").

2. Technical Architecture

Tech Stack

- **Frontend Framework:** React (Functional Components with Hooks).
- **Styling:** Tailwind CSS (Utility-first CSS).
- **Map Engine:** Leaflet JS (via direct DOM manipulation in React).
- **Icons:** Lucide React.
- **Backend / Database:** Firebase Firestore (NoSQL).
- **Authentication:** Firebase Auth (Anonymous & Custom Tokens).

Data Structure (Firestore)

The application uses a specific data model to handle routes and user submissions.

1. Segments Collection (segments_v5)

Represents a physical stretch of road between two towns.

- id: Unique identifier (e.g., a1-01).
- name: Display name (e.g., "Colombo - Peliyagoda").
- start / end: Lat/Lng coordinates.
- status: 'clear' | 'warning' | 'blocked'.
- verified: Integer count of positive votes.
- blocked: Integer count of negative votes.
- routes: Array of route IDs this segment belongs to (e.g., ['colombo-kandy',

'colombo-kurunegala']). *This allows one road segment to appear in multiple route searches.*

2. Alternatives Collection (alternatives)

User-submitted detours.

- segmentId: The ID of the blocked/warned segment this alternative bypasses.
- points: Array of objects [{lat, lng}, ...] representing the path.
- description: Text description of the detour.
- userId: ID of the submitter.

3. Component Breakdown (App.jsx)

The application is contained within a single file for portability, but logically divided into these parts:

A. Configuration & State

- **ROUTES Constant:** Defines the metadata (ID, Name, Color) for every supported destination (e.g., Colombo-Kandy, Colombo-Jaffna).
- **SEED_SEGMENTS Constant:** Contains the hardcoded initial data for the road network. This is loaded into Firestore if the database is empty.
- **App Component State:** Manages user, segments, alternatives, selectedSegment, activeRouteId, search terms, and drawing modes.

B. Map Component (MapComponent)

A wrapper around the native Leaflet library.

- **Initialization:** Loads Leaflet CSS/JS from CDN if not present to ensure the map renders in the environment.
- **Rendering:**
  - **Polylines:** Draws colored lines for road segments based on status.
  - **Interactive Layers:** Adds invisible, thicker lines over segments to make clicking easier on mobile devices.
  - **Markers:** Draws start/end points for segments.
- **Drawing Mode:** Listens for clicks when isDrawingMode is active to plot coordinates for alternative routes.

C. Sidebar UI

The control center of the application.

1. **Header & Search:** Contains the "Find Route" inputs. The search logic scans segment names and route definitions to intelligently switch the activeRouteId.
2. **Segment List:** If no specific segment is selected, this shows all road sections for the current route.

3. **Detail View:** When a segment is clicked, this view appears. It allows users to:
   - Vote on status (Verify/Block).
   - View existing alternative routes.
   - Enter "Drawing Mode" to create a new detour.

4. Key Logic Flows

Route Filtering Logic

Because Sri Lanka's road network is interconnected (e.g., the road to Anuradhapura shares the first 70km with the road to Kandy), segments are not duplicated.

- **Storage:** A segment has a routes array: ['colombo-kandy', 'colombo-kurunegala'].
- **Display:** When the user selects "Colombo - Kandy", the app filters segments: segments.filter(s => s.routes.includes('colombo-kandy')).

Search Algorithm

1. **Direct Route Match:** Checks if the user typed "Colombo" and "Kandy". If yes, switches directly to that route ID.
2. **Segment Match:** If the user types "Nugegoda", the app finds the segment "Colombo - Nugegoda".
3. **Heuristic:** It calculates which Route ID appears most frequently among the matching segments and switches to that route.

Alternative Route Creation

1. User clicks "Add New Detour".
2. isDrawingMode becomes true.
3. User clicks points on the map.
4. Points are stored in drawnPoints state.
5. On "Save", points are formatted as objects and pushed to the alternatives Firestore collection.
6. The parent segment is automatically marked as "Warning" status to alert other drivers.

5. How to Run

1. Ensure you have a valid **Firebase Configuration** object available in the global scope as __firebase_config (or replace the parsing logic with your own config object).
2. Ensure lucide-react, firebase, and leaflet dependencies are available.
3. Run the App component within a React environment.