Software Requirement Specification

Application Overview

Application Concept

A web application that provides real-time neighborhood safety ratings and local safety news alerts. Users can see unsafe "red zones," view detailed safety information, check the history of past incidents, and contribute local news updates.

Application Specification

Frontend (Next.js)

- Framework: Next.js 14 (App Router)
- Styling: Tailwind CSS (for UI + responsive design)
- Map: MapLibre GL JS (supports overlays for Red Zones, markers, clustering)
- State management: React Query / TanStack Query → data fetching + caching
- Deployment: Vercel (free tier is enough for MVP)

Backend & Database (Supabase)

- Supabase Postgres → store users, red zones, news, and history data
- PostGIS extension → geospatial queries (red zone polygons, event locations)
- Supabase Realtime → live updates for news posts and red zones
- Supabase Auth → Google OAuth 2.0 for login & profile management

Authentication

- Users sign in with Google → handled by Supabase
- Profile data (name, email, picture) pulled via Supabase Auth API
- Session tokens are handled automatically by the Supabase client SDK

Specification

- Web Application developed using -
  - Typescripts
- Compatible with browsers (Chrome, Safari)
- 3 key features
  - Viewing Red Zone Details
  - Viewing News
  - Displaying Past Events

System Requirement

Functional Requirement

1. Authentication and User Profile

Authentication

- **Create a New Account with a Google Account**
    1. The system must automatically create a new user account when there is no user's Google email address registered in the system.
    2. The system must retrieve information: Full name, Email and profile picture from Google API.
    3. The system must display the "Terms and Conditions" for the user to accept before creating the new account.
- **New User onboarding**
    1. The system must automatically create a new user account when there is no user's Google email address registered in the system.
    2. The system must retrieve information: Full name, Email, and profile picture from the Google API.
    3. The system must display the "Terms and Conditions" for the user to accept before creating the new account.
- **Google Sign-in for Existing User**
    1. The system must display a sign-in/login button on the register/login page
    2. The system must display a pop-up window or redirect to Google's account verification page when the user clicks the button.
    3. The system will use the token that was sent by Google after the user successfully verifies the account. Then, a session will be created with the given token.
- **Account Linking**
    1. The system must allow the existing user to link their Google account to their current profile through the "User Profile" page.

User Profile

- **Accessing the User Profile page**
    1. The system must display a link or button that leads to the "User Profile" page after the user logs in.
    2. Users must be able to access the User Profile page easily from all system pages.

3. Users must be logged in before accessing the User Profile page. The system must redirect users to the login page if the user is not logged in.

- **Display Profile Information**
  1. User Profile page must display the user's current information:
     a. Full Name
        i. The system must display the name initially retrieved from the Google account or the user's recently edited name.
     b. Email Address
        i. The system must display an email for login (cannot be edited)
     c. Profile Picture:
        i. The system must display the profile picture in a 1:1 ratio using either the image initially retrieved from the user's Google account or their most recently updated profile picture.
- **Editing the user's profile information**
  1. Users must be able to edit the Full Name and Profile Picture.
  2. The system must update the changes when the user saves the edited full name of the profile picture.
     a. The changes will not affect the Google account.
  3. Users cannot edit the Email Address.

2. MVP Feature 2 – Scouting Mode (Red Zone Safety Monitoring)

Selecting Scouting Mode

- The system must allow users to select "Scouting Mode" from the main menu.
- The system must highlight unsafe areas as red zones on the map.

Viewing Red Zone Details

- The system must display red zone information when tapped:
  1. Risk Type (crime, accident, protest, etc.)
  2. Severity Level
  3. Time of Report

Decision Support

- The user can review the details and decide whether to proceed or avoid the area.
- The system must update the map in real-time when safety data changes.

3. MVP Feature 3 – News Announcements on Map

Viewing News

- The system must retrieve real-time/local safety news and display it as markers on the map.

- The system must allow users to tap a marker to read details, including:
    1. Title
    2. Description
    3. Time of Report
    4. News Source
    5. Recommended Action (if any)

Posting News (Community Contribution)

- The system must allow users to post news by filling in: Title, Description, Location, and optional media.
- The system must validate submitted news before publishing.
- The system must display successfully posted news as a marker visible to all users.

4. MVP Feature 4 – History Mode (Past Event Analysis)

Selecting History Mode

- The system must allow users to view archived safety/news data by choosing a time range (1 Month, 6 Months, 1 Year, 5 Years).

Displaying Past Events

- The system must show past events as pins, color-coded by event type or severity.
- The system must provide filtering options (e.g., accidents only, protests only).

Decision Making

- Users must be able to compare trends across time ranges for travel planning.

Non-Functional Requirement

1. Authentication and User Profile

Authentication

- **Security**
    1. The system must use the OAuth 2.0 Protocol to connect Google API.
    2. The received token from Google must be validated before each use.
- **Performance**
    1. The entire login process (latency) must not take more than 3 seconds. (from user clicking the login button to accessing the main page)
- **Usability**
    1. The sign-in/log-in button must be easy for the user and clearly visible on all devices
    2. When there is an unsuccessful or failure case during the authentication process, the error message must be easy to understand and provide clear instructions.

User Profile

- **Data Validation**
    1. The system must validate that the edited name
        a. The name field must not be empty.
        b. The name must not contain any emojis.
        c. The name must not exceed 50 characters.
    2. The system must check the uploaded image for appropriate file types and size.
        a. Only allow .jpg, .jpeg, .png
        b. File size must not exceed 5 MB
- **State Management**
    1. User's profile changes information must be updated in all parts of the application where it is displayed.

2. MVP Feature 2 – Scouting Mode (Non-Functional)

- **Performance:** Map and red zone overlays must load within 3 seconds.
- **Usability:** Red zones must be color-blind accessible (patterns or symbols in addition to color).
- **Security:** Location data must be encrypted and stored no longer than 24 hours.

3. MVP Feature 3 – News Announcements (Non-Functional)

- **Performance:** Map and red zone overlays must load within 3 seconds.
- **Usability:** Red zones must be color-blind accessible (patterns or symbols in addition to color)
- **Security:** Location data must be encrypted and stored no longer than 24 hours.

4. MVP Feature 4 – History Mode (Non-Functional)

- **Performance:** Historical data retrieval must not exceed 4 seconds.
- **Usability:** Filtering options must update the display instantly (<1 second).
- **Portability:** Works consistently across Android 10+ and iOS 15+.