Software Requirements and Design Document

For

Group 10

Version 1.0

Authors:
Kirra Orndorff
Alexander Jubran
Lauren Chang
Riley Galpin
Michael Greenberg

1. Overview (5 points)

We plan to build an app that helps runners decide where they should run, and what they should wear for their run. It would then recommend what you should wear (long sleeves, gloves, etc), and would take into account temperature, wind speed, and humidity. The second part is the route recommender, which would have a list of user saved routes with different parameters such as terrain (concrete, asphalt, dirt) and elevation change. A weekly workout planner would also be included, which could provide workout schedules (for certain goals). We also hope to implement shoe mileage tracking.

2. Functional Requirements (10 points)

1.Weather & Recommendations

- National Weather Service Integration: System shall fetch real-time data via the NWS API. The system shall connect to the National Weather Service API to retrieve the current temperature (Fahrenheit/Celsius), wind speed (mph), and humidity (%) based on the user's GPS coordinates - **HIGH**
- Manual Weather Override: Users must be able to manually input weather conditions or a location (temp, humidity, precipitation) if they choose not to share location data. - **HIGH**

2. Clothing Recommendation Engine

- An algorithm shall recommend specific gear based on weather data and a user-defined sensitivity setting (e.g., "I run cold" vs. "I run hot"). - **HIGH**
- The system shall compare current weather data against a predefined clothing logic table to output a specific recommendation (e.g., "Wear a windbreaker and gloves") - **HIGH**

3. Route & Workout Planning

- Algorithmic Route Suggestions: System shall recommend routes based on the selected workout type, current weather, and terrain preferences. - **MEDIUM**
- User Route Management: Users shall have the ability to input and save their own custom routes. The system shall allow users to manually save, edit, and delete a "Route" by inputting a title, total distance, terrain type (Concrete, Asphalt, Dirt, or Trail), and estimated elevation gain (optional). - **MEDIUM**
- Training Plans: System shall generate a daily calendar of runs over a 4-week to 16-week period based on a user-selected goal: Stay in Shape, 5k, 15k, or Marathon. - **LOW**
- (Difficulty Selection): The system shall allow users to modify a training plan's intensity using a difficulty slider that adjusts target pace and weekly mileage volume. - **LOW**.
- Pace & Goal Setting: Users can set and track specific pace goals within their training plans. - **LOW**
- (Mapping Interface): (Proposed) The system shall display the user's saved routes and current location on an interactive map using the Google Maps API. **- MEDIUM**

4. Activity & Gear Tracking

- Activity Logging: System shall record mileage, elevation gain, and terrain type for every run. - **MEDIUM**

- Shoe Mileage Tracker: Users can add multiple pairs of shoes, categorize them by terrain type (e.g., trail vs. road), and track cumulative mileage on each. - **LOW**
- The system shall include a "Shoe Closet" feature where users can input the make/model of a shoe and set a maximum mileage alert (default 400 miles). - **LOW**
- Race Calendar: Users can input upcoming race dates to align with their training schedule. - **LOW**

5. User Profile & Data Visualization

- Weekly Planner View: A dashboard displaying the upcoming week's workouts, predicted weather, and scheduled routes. - **MEDIUM**
- Performance History: A profile section to view past workouts, Personal Records (PRs), and total mileage statistics. - **MEDIUM**

3. Non-functional Requirements (10 points)

Privacy-First Architecture: All user data and activity logs shall be stored locally on the device by default to maximize user privacy. - **HIGH**

Offline Functionality: The core features (workout logging, manual weather input, and local route viewing) must remain functional without an internet connection. - **HIGH**

Data Security: If data is synced, it must be encrypted; however, the primary goal is a "no-service" capability for remote runs. - **HIGH**

Algorithm Performance: Route and clothing recommendations should be calculated in under 2 seconds to ensure a smooth user experience. - **MEDIUM**

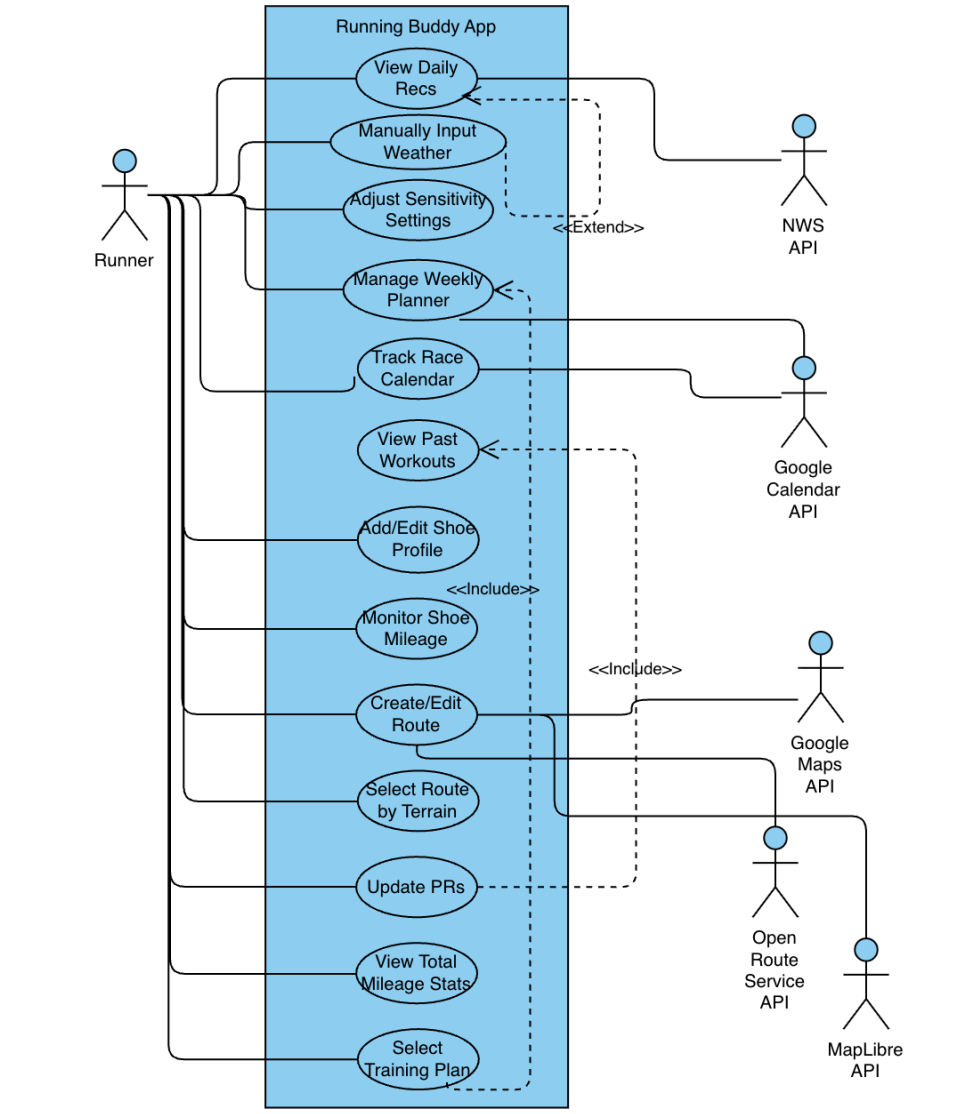
4. Use Case Diagram (10 points)

Home screen: opens with weather, clothing recommendations, and route recommendations. If offline, open asking to input weather first.
Calendar tab
- weekly planner
- race calendar
- Past workouts

Shoe Closet tab
- Create / edit shoe screen
- View all current shoes screen

Route creation tab
- Create / edit route screen
- View all routes screen

Profile tab + performance history
- Prs (manual input)
- Total mileage stats
- Goals



5. Class Diagram and/or Sequence Diagrams (15 points)

Class Definitions:

User Profile:

- Attributes: UserID (int), Name (str), ColdSensitivityScore (int), TotalLifetimeMileage (float).
- Methods: UpdatePR(), GetWeeklySummary().

Route:

- Attributes: RouteID (int), TerrainType (Enum), ElevationChange (float), Distance (float), IsUserSaved (bool).

WeatherService:

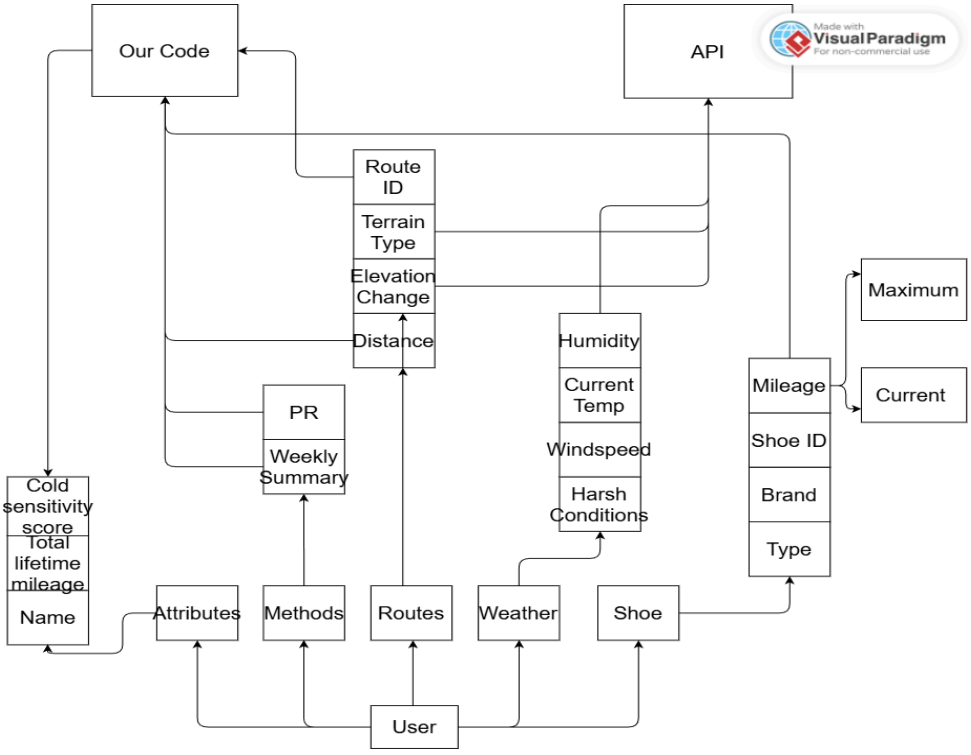
- Attributes: CurrentTemp (float), Humidity (float), WindSpeed (float).

- Methods: FetchWeatherData(), CalculateApparelScore(User).

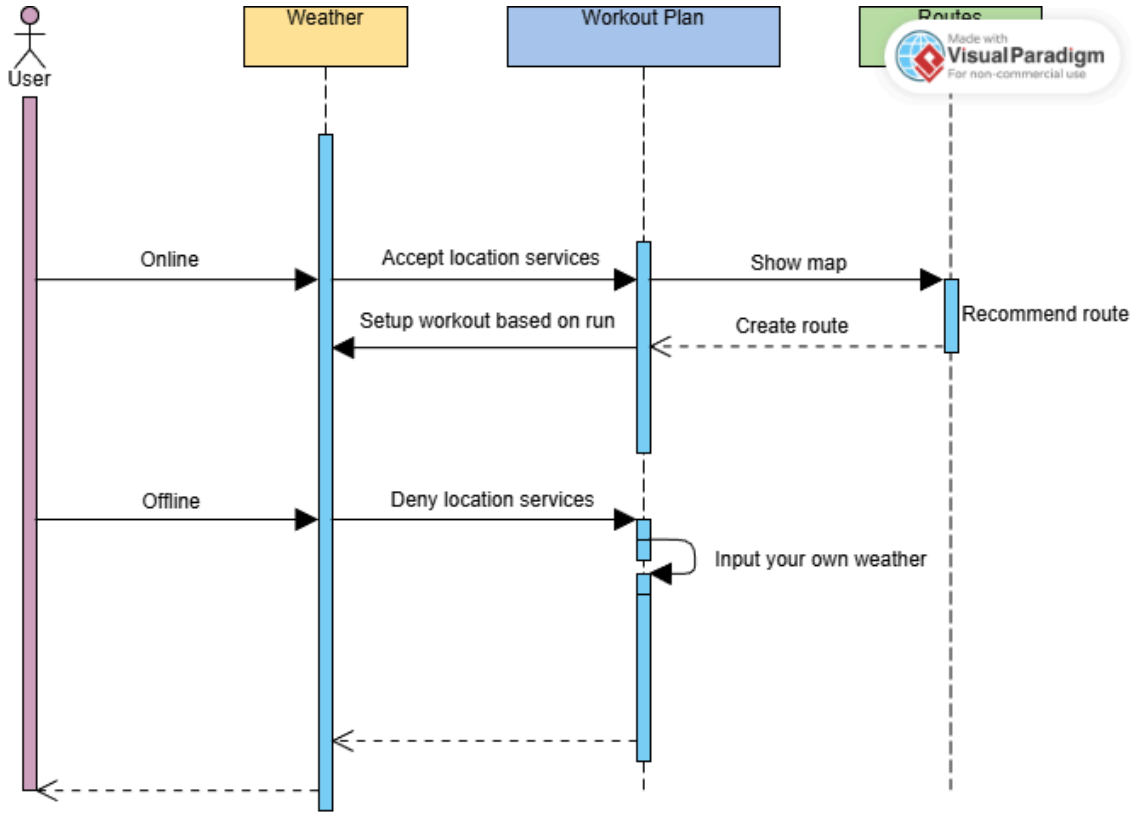
Shoe:
- Attributes: ShoeID (int), Brand (str), CurrentMileage (float), MaxMileage (float), ShoeType (Enum).
- Methods: AddRunMileage(float), CheckReplacementStatus().

WorkoutPlan:
- Attributes: PlanID (int), GoalType (str), TargetPace (float), StartDate (date).

Class Diagram:



Sequence Diagram:



6. Operating Environment (5 points)

Hardware Platform: The application is designed to be accessed via standard computing hardware including desktop PCs, laptops, and mobile devices (smartphones and tablets) that support modern web browsers.

Operating Systems: Because .NET MAUI is cross-platform, the application will operate on Windows 10/11, macOS, Android (API 21 or higher), and iOS (15.0 or higher) environments.

Software Dependencies/Coexistence:

- Web Browsers: The system must peacefully coexist and function within current versions of Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

- External APIs: The environment requires an active internet connection to communicate with the National Weather Service/OpenWeatherMap API, Google Maps API, OpenRouteService for routing API, OpenStreetMap contributors for map data, MapLibre for the mapping library, Turf.js for geospatial calculation, Google Calendar API for planner/calendar.

- Local Storage: The application will utilize built-in storage via the .NET MAUI framework for offline data persistence and user privacy.

7. Assumptions and Dependencies (5 points)

Assumptions:

- API Stability: We assume all APIs will remain free to use and provide consistent data formats throughout the development cycle.

- User Honesty: We assume users will provide accurate terrain and elevation data for their custom-saved routes.

- Hardware Compatibility: We assume that the target mobile and desktop devices have sufficient local storage to handle the .NET MAUI built-in database for offline functionality.

Dependencies:

- Google Maps API/OpenRouteService/OpenStreetMap/MapLibre/Turf.js: The route recommendation and mapping features are dependent on the successful integration and licensing of the Google Maps API/OpenRouteService/OpenStreetMap/MapLibre/Turf.js

- Google Calendar API: The planner and calendar features are dependent on the successful integration and licensing of the Google Calendar API.

- Framework Support: The project depends on the .NET MAUI framework and the UraniumUI library for cross-platform UI rendering

- Third-Party Libraries: Our MVVM architecture relies on the Community Toolkit.Mvvm for data binding and application logic.