Group members: Vishvum Sriram & Tarun Valluri

Project Title: Sustainable Transportation and Commuting Planner

Problem to be solved: Develop a system that helps individuals plan and track their eco-friendly transportation choices, such as walking, cycling, and carpooling.

Web framework: Spring Boot

Goals:

- User Profile Creation: Allow users to create profiles to save their preferences and history.
    - Implement a feature allowing users to register with the system and create a personal profile where they can store their preferences and history.
        - Java for the backend to manage user data
        - Use HashMap to store data
        - HTML/CSS/JavaScript to create forms for user input (front end)
    - Spring Security: Utilize Spring Security for authentication and secure user registration
    - Spring Data JPA: Manage user data with Spring Data JPA repositories to abstract the database layer
- Transportation Mode Selection: Enable users to select from various eco-friendly transportation options (walking, cycling, public transport, carpooling).
    - Provide an interface where users can choose their preferred transportation mode
        - Use Java Enums (constants) to represent different transportation modes
        - Store user preferences in the database and map it with the user profile
        - Use HashMap to store preferences
    - Spring MVC: Leverage Spring MVC controllers to handle HTTP requests for selecting transportation modes
    - Java Enum: Define transportation modes as an Enum and use Spring's data binding features to bind them to user input
- Route Planning: Integrate a basic route planner that suggests the most eco-friendly route based on the selected mode of transport.
    - Offer a route planner that recommends eco-friendly routes based on user-selected transportation modes
        - Use an API (mapbox or openstreetmap)
        - Java
        - Use graphs?
    - Integration with APIs: Use Spring's RestTemplate or WebClient to consume third-party mapping and routing services
    - Domain Model: Create a domain model to represent routes and waypoints, persisting them using Spring Data JPA
- Carbon Footprint Calculator: Calculate and display the carbon footprint savings of choosing eco-friendly transportation over traditional methods.

    - Show alternative, cheaper options
    - Calculate and show the carbon savings of using eco-friendly transport options
        - Java class to calculate carbon emissions
            - Use information from route planning and transportation mode
- Local Weather Integration: Provide weather information to help users plan their trips better.
    - Incorporate weather forecasts to help users plan trips
        - Integrate weather API (OpenWeatherMap)
    - External API Consumption: Use Spring Boot's RestTemplate or the newer WebClient to fetch weather information from external APIs
    - Caching: Implement caching with Spring's caching abstraction to store frequently accessed weather data and reduce API calls
- Gamification & Rewards: Introduce points or badges for completing eco-friendly trips to motivate users.
    - Implement a system of points and badges to reward users for eco-friendly travel
        - Create a reward system for users
        - Store in database HashMap
        - Use HTML/CSS/JavaScript for front-end design (react?)
    - Database Relationships: Use Spring Data JPA to model the relationships between users and their rewards
- Real-time Traffic Data: Use real-time traffic information to suggest the fastest eco-friendly routes.
    - Find a real-time traffic data service API
    - Use Java to parse and integrate traffic data with the route planning feature
    - Implement HashMap data structure to where the key could be the route ID
    - WebSocket: For real-time features, such as traffic updates, use Spring's support for WebSocket to push data to the client
- Public Transport Schedules: Integrate public transport timetables to assist in planning.
    - Use a public transport API **(GTFS google**) to get the scheduled data
    - Use Java to create a scheduler that updates the transport schedules in your database at regular intervals.
    - Implement HashMaps **and lists** to quickly access the schedules.