Senior Front-End Developer Assessment: Real-Time Logistics Tracker

Type: Take-Home Assignment
Technology Requirement: TypeScript, React

Overview

The objective of this project is to create a web interface that allows dispatchers to track delivery drivers, manage delivery statuses, and handle real-time updates effectively.

This application should feature key elements such as **real-time map integration**, **optimistic UI**, **WebSocket communication**. The goal is to assess your ability to design scalable, maintainable front-end solutions, work with complex state management, and handle asynchronous data flow.

Core Features

1. Live Map Visualization

- Display a dynamic map showing the real-time locations of delivery drivers.
- Each driver should be represented by a unique identifier and their current location coordinates (latitude and longitude).
- When a driver is selected from the list, their details should be displayed, and their location should be highlighted on the map.

2. Driver Dashboard Panel

- Provide a list of drivers showing the following details:
    - Driver name or ID
    - Current location

    - Delivery status (e.g., **Delivering**, **Paused**, **Idle**)
    - Estimated time of arrival (ETA) — this can be mocked for simplicity.
- Allow filtering or sorting of drivers based on their delivery status.

3. Real-Time Updates

- Implement functionality to receive and display live updates on driver locations and delivery statuses via **WebSocket**.
- These updates should trigger changes in the application state, and the UI should reflect these updates in real-time.

4. Dispatch Actions with Optimistic UI

- Allow dispatchers to perform the following actions:
    - Reassign a delivery to another driver.
    - Mark a delivery as completed.
    - Pause or resume a driver.
- These updates should be immediately reflected in the UI (optimistic updates), even while waiting for server confirmation.
- In case of failure, ensure proper error handling and rollback of the UI state.

Notes

- You are free to use any state management solution that fits the task, but we recommend using **Redux** for managing global state such as driver details, assignments, and UI states.
- The application must be implemented in **TypeScript**.

- You will need to handle or **mock the WebSocket server** to simulate real-time updates.
- Use **Mapbox** or **Leaflet** to integrate the map. The map should display real-time driver positions and allow user interaction.

Deliverables

- A **Git repository** containing the source code for the project.
- A **README.md** file with:
    - Clear instructions for setting up and running the application.
    - An explanation of your architectural decisions and design trade-offs.
    - Known limitations or areas that could be further optimized.

WebSocket Server Implementation

As part of this assignment, you will need to implement or mock the WebSocket server to simulate real-time updates. The server should emit messages that simulate driver updates, such as location and status.

A sample message might look like this:

```
{
  "driverId": "123",
  "latitude": 40.7128,
  "longitude": -74.0060,
  "status": "Delivering",
  "eta": "15 min" //optional
}
```

You are free to choose your preferred WebSocket server implementation (e.g., using **Node.js** with the **ws** library or another framework). If you choose to mock the server, ensure it behaves like a real WebSocket service, sending periodic updates for each driver.

Environment Requirements

- **TypeScript** is mandatory for the project.

- Use a **map library** such as **Mapbox** or **Leaflet** to render the driver locations on a map.
- Optional (support offline mode)

Good luck, and we look forward to seeing your solution! Backend can be mocked or fully done up to you.