



Full-stack Home Assignment Vehicle Management App

Overview

Welcome to the Optibus Fullstack Home Assignment!

This exercise is designed to help us evaluate your technical skills, creativity, and approach to problem-solving.

You'll be building a small **Vehicle Management App** that allows users to manage a fleet of vehicles — create, edit, delete, and track their current status.

We recommend you to approach this assignment as you would a real-world task: structure your code cleanly, write readable and maintainable logic, and don't hesitate to add improvements or small features that make the app more user-friendly.

Goal

Implement a Vehicle Management View that includes:

1. Full **CRUD** functionality for vehicles (Create, Read, Update, Delete).
2. A **vehicle status table**, allowing users to view and manage vehicles by their status:
 - a. Available (default)
 - b. InUse
 - c. Maintenance

Functional Requirements

Your application should enable the following:

- **List View**
 - Display all vehicles in a table with the following columns:
 - `id`
 - `licensePlate`
 - `status`
 - `createdAt`
 - **Create / Edit / Delete Vehicle**
 - Add a new vehicle.
 - Edit an existing vehicle's details.
 - Delete a vehicle with confirmation.
 - **Status Management**
 - Each vehicle has a `status` (Available, InUse, Maintenance).
 - Default status for new vehicles: **Available**.
 - Ability to change status via dropdown or other intuitive control.
 - **Validations**
 - A vehicle in **maintenance** status can **only** move to status Available
 - A vehicle that is InUse or Maintenance can not be deleted
 - Only **up to 5%** of the vehicles can be in **maintenance** at the same time
 - **Seed Data**
 - Provide example data (e.g., `vehicles.json`) with at least 3–5 vehicles, one in each status.
-

Requirements

✓ Required

- TypeScript throughout the app (frontend and backend).
- CRUD Operations
- Status Management
- Clean Code
- Tests

💡 Nice to have

- Persistence
- Sorting / Filtering / Search
- Responsive / Polished UI
- Error handling and loading states.

Deliverables

Please submit your solution as a **zip file** containing:

- The full project source code.
- **README.md** with setup and run instructions
- **vehicles.json** (seed data)
- Optional: simple API documentation (list of routes or Swagger/OpenAPI file)

Timeline

🕒 Deadline: Submit within 3 days of receiving the assignment..

Good luck and have fun building!

We're looking forward to seeing how you approach the challenge.
