Project: Fleet Maintenance Scheduling System

1. Introduction

This document outlines the Low-Level Design for a **Fleet Maintenance Scheduling System** aimed at improving vehicle uptime, reducing breakdowns, and ensuring regulatory compliance.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks.

2. Functional Modules

1. **Vehicle Registration & Tracking** – Enables registration and tracking of fleet vehicles along with their mileage updates.
2. **Maintenance Scheduling** – Automates scheduling of maintenance tasks based on mileage or calendar intervals.
3. **Service History Management** – Maintains a detailed log of all completed maintenance services and associated costs.
4. **Technician Assignment** – Facilitates assigning available technicians to scheduled service jobs efficiently.
5. **Dashboard and Analytics** – Provides visual insights and reports on service schedules, costs, and technician performance.

3. Technology Stack

- **Frontend**: Angular or React
- **Backend**: REST API-based microservices
- **Database**: Relational DB (MySQL/SQL Server)

4. Module Details

4.1 Vehicle Registration & Tracking

Entities

- Vehicle: VehicleID, Type, Make, Model, Year, VIN, LastServiceDate
- OdometerReading: ReadingID, VehicleID, Timestamp, Mileage

APIs

- POST /api/vehicles – Add vehicle
- GET /api/vehicles – List all vehicles
- POST /api/vehicles/{id}/odometer – Add odometer reading
- GET /api/vehicles/{id}/odometer – View mileage history

4.2 Maintenance Scheduling

Entities

- MaintenancePlan: PlanID, VehicleType, FrequencyKM, FrequencyDays
- ScheduledService: ServiceID, VehicleID, DueDate, DueKM, Status

APIs

- POST /api/plans – Define maintenance plan
- POST /api/services/schedule – Schedule maintenance
- GET /api/services?vehicleId= – Get scheduled services
- PUT /api/services/{id}/complete – Mark service completed

4.3 Service History Management

Entities

- ServiceRecord: RecordID, VehicleID, ServiceDate, Description, Cost

APIs

- POST /api/services/record – Log completed service
- GET /api/services/history?vehicleId= – View service history

4.4 Technician Assignment

Entities

- Technician: TechnicianID, Name, Skills, Availability
- ServiceAssignment: AssignmentID, ServiceID, TechnicianID, Status

APIs

- POST /api/technicians – Register technician
- POST /api/assignments – Assign technician to service
- GET /api/assignments?technicianId= – Technician schedule

4.5 Dashboard and Analytics

Features

- Upcoming service due reports
- Cost analysis by vehicle
- Technician workload summary

APIs

- GET /api/reports/upcoming-services
- GET /api/reports/cost-summary?vehicleId=
- GET /api/reports/technician-summary

5. Simplified Database Schema

Table Name	Primary Key	Foreign Key
Vehicle	VehicleID	–
OdometerReading	ReadingID	VehicleID
MaintenancePlan	PlanID	–
ScheduledService	ServiceID	VehicleID
ServiceRecord	RecordID	VehicleID
Technician	TechnicianID	–
ServiceAssignment	AssignmentID	ServiceID, TechnicianID

6. UI Overview

- **Fleet Overview**: All registered vehicles and their statuses
- **Maintenance Calendar**: Scheduled services view
- **Technician Panel**: Assignments and availability
- **Reports**: Downloadable PDFs/Excel

7. Security

- JWT-based user authentication
- Roles: Fleet Manager, Technician, Viewer
- Role-based access control for endpoints

8. Assumptions & Constraints

- Odometer updates are manual or from external integration
- Real-time alerts and notifications are out of scope
- No integration with OEM telematics systems