Project: Vehicle Service Booking System

1. Introduction

This document details the Low-Level Design (LLD) for a **Vehicle Service Booking System**, which enables users to book vehicle service appointments, track service history, and manage service center operations efficiently.

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

2. Module Overview

2.1 User Management
Handles customer registration, profile management, and service history.

2.2 Vehicle Management
Manages customer vehicles including model, make, and registration details.

2.3 Service Booking
Allows users to schedule service appointments and track their status.

2.4 Service Center Management
Manages service center availability, mechanics, and service types.

2.5 Invoice and Billing
Generates invoices for completed services and manages payment records.

3. Architecture Overview

3.1 Architectural Style

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

3.2 Component Interaction

- The frontend communicates with the backend REST APIs.
- The backend processes requests, interacts with the database, and sends responses.
- All user interactions and bookings are performed through the web interface.

4. Module-Wise Design

4.1 User Management Module

Features

- User registration, login, and profile updates
- View service booking history

Entities

- **User**: UserID, Name, Email, Phone, Address, PasswordHash

4.2 Vehicle Management Module

Features

- Register and manage vehicles
- Link vehicles to user profiles

Entities

- **Vehicle**: VehicleID, UserID, Make, Model, Year, RegistrationNumber

4.3 Service Booking Module

Features

- Book service appointments
- Cancel or reschedule bookings
- Track service status

Entities

- **Booking**: BookingID, UserID, VehicleID, ServiceCenterID, Date, TimeSlot, Status

4.4 Service Center Management Module

Features

- Manage service centers and mechanics
- Define available service types

Entities

- **ServiceCenter**: ServiceCenterID, Name, Location, Contact
- **Mechanic**: MechanicID, ServiceCenterID, Name, Expertise
- **ServiceType**: ServiceTypeID, Description, Price

4.5 Invoice and Billing Module

Features

- Generate invoice upon service completion
- Manage and record payments

Entities

- **Invoice**: InvoiceID, BookingID, ServiceTypeID, TotalAmount, PaymentStatus

5. Deployment Strategy

5.1 Local Deployment

- Angular/React for frontend (via local dev servers)
- Spring Boot/ASP.NET Core for backend (via localhost server)
- Local relational DB setup for development (MySQL/PostgreSQL/SQL Server)

6. Database Design

Table Name	Primary Key	Foreign Keys
User	UserID	–
Vehicle	VehicleID	UserID
Booking	BookingID	UserID, VehicleID, ServiceCenterID
ServiceCenter	ServiceCenterID	–
Mechanic	MechanicID	ServiceCenterID
ServiceType	ServiceTypeID	–
Invoice	InvoiceID	BookingID, ServiceTypeID

7. User Interface Design

Wireframes

- User Dashboard
- Vehicle Management Page
- Service Booking Form
- Service Center Schedule Viewer

- Invoice and Payment History

8. Non-Functional Requirements

8.1 Performance

- Must support 200 concurrent users in a local test environment

8.2 Scalability

- Easily deployable to cloud if needed in future

8.3 Security

- User data secured via encryption and role-based access control

8.4 Usability

- Optimized for mobile and desktop browsers
- Easy appointment navigation with calendar view

9. Assumptions and Constraints

Assumptions

- Each user owns at least one vehicle
- Bookings are made at least one day in advance

Constraints

- Real-time mechanic assignment is not included in this version
- Payment gateway integration is out of scope for local deployment