Project: Vehicle Parking Management System

1. Introduction

This document outlines the Low-Level Design (LLD) for a **Vehicle Parking Management System** that facilitates efficient parking slot management, vehicle entry/exit logging, reservations, and billing operations. The system aims to streamline both visitor and subscriber parking operations in real time.

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

2. Module Overview

2.1 User Management
Handles registration, login, and role-based access for admins, staff, and customers.

2.2 Parking Slot Management
Manages slot types, occupancy status, and availability.

2.3 Vehicle Entry & Exit Logging
Tracks vehicle movement and updates slot status.

2.4 Reservation System
Enables users to book slots in advance and manage them.

2.5 Billing and Payments
Generates bills based on usage duration and allows payment processing.

3. Architecture Overview

3.1 Architectural Style

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

3.2 Component Interaction

- Frontend communicates with backend APIs.
- Backend manages business logic and data operations.
- Database stores persistent data like user info, reservations, and invoices.

4. Module-Wise Design

4.1 User Management Module

Features

- Register and login
- Assign roles and permissions

Entities

- **User**: UserID, Name, Email, Phone, Role, Password (hashed)

4.2 Parking Slot Management Module

Features

- Add/remove/update parking slots
- Check real-time availability

Entities

- **ParkingSlot**: SlotID, Type (2W/4W), IsOccupied, Location

4.3 Vehicle Entry & Exit Logging Module

Features

- Log entry and exit
- Calculate duration and update occupancy

Entities

- **VehicleLog**: LogID, VehicleNumber, EntryTime, ExitTime, SlotID, UserID

4.4 Reservation System Module

Features

- View availability and reserve slots
- Modify or cancel reservations

Entities

- **Reservation**: ReservationID, UserID, SlotID, VehicleNumber, StartTime, EndTime, Status

4.5 Billing and Payments Module

Features

- Dynamic billing based on time and vehicle type
- Payment via multiple modes

Entities

- **Invoice**: InvoiceID, UserID, Amount, PaymentMethod, Status, Timestamp

5. Deployment Strategy

5.1 Local Development

- Frontend served via Angular CLI or React dev server
- Backend run with Spring Boot or .NET CLI
- DB setup using local MySQL/PostgreSQL/SQL Server instance

6. Database Design

Table Name	Primary Key	Foreign Keys
User	UserID	–
ParkingSlot	SlotID	–
VehicleLog	LogID	UserID, SlotID
Reservation	ReservationID	UserID, SlotID
Invoice	InvoiceID	UserID

7. User Interface Design

7.1 Wireframe Elements

- Login/Signup
- Dashboard for Admin/Staff/User
- Slot Availability Map
- Reservation Page
- Vehicle Entry/Exit Management
- Billing Summary and Payment

8. Non-Functional Requirements

8.1 Performance

- Support 200+ concurrent users in dev/test setups

8.2 Scalability

- Horizontal scalability using containerization (optional in future)

8.3 Security

- Role-based access

- Encrypted password storage
- HTTPS for all data exchange

8.4 Usability

- Responsive and mobile-friendly UI
- WCAG-compliant design

9. Assumptions and Constraints

Assumptions

- Each slot can accommodate only one vehicle
- Entry/exit operations are manual or via scanner

Constraints

- SMS/Email notification is out of scope
- Third-party integrations are not included in this phase