Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept

Prof. Dr. Hicham Elmongui
CSE371 Database System

Smart City Parking Management System

1. Introduction

The Smart City Parking Management System is designed to optimize parking in urban areas. The system connects parking lots, parking meters, and users (drivers) through a centralized database. The goal is to manage parking availability, reservations, pricing, and traffic flow effectively, with real-time updates. This system introduces IoT-style interaction for smart city applications, where each parking spot is monitored and reported in real-time to users.

System Objectives

- **Real-Time Availability**: Track available parking spots across multiple locations in real-time.
- **Dynamic Pricing**: Implement a dynamic pricing model based on time of day, demand, and location.
- **User Reservations**: Allow drivers to reserve parking spots ahead of time, with penalties for non-use.
- **Role-Based Access**: Differentiate between regular users (drivers), parking lot administrators, and system administrators.
- **Web Integration**: Enable a web-based interface for users to search, reserve, and navigate to available spots.

2. Project Requirements

The project requirements are outlined in the following section, detailing the features and functionalities to be implemented:

1. Parking Lot Management:

- **Parking Lot Profiles**: Each parking lot will have a profile with details like location, capacity, types of parking spots (e.g., regular, disabled, EV charging), and pricing structure.
- **Parking Spot Status**: Monitor and update the status of each parking spot in real-time (occupied, available, reserved).
- **Dynamic Pricing:** Introduce a pricing model that adjusts based on various factors, including demand and location. (e.g., surge pricing during peak hours)

2. User Management:

- **Driver Profiles**: Allow users to register with personal details, license plate number, and payment method.

- **Reservation System**: Users can search for parking spots by location, reserve a spot for a certain duration, and get real-time navigation directions to the parking lot.
- **Reservation Rules**: Set rules for reservations, such as time limits, penalties for not showing up, and automatic release of reserved spots after expiration.

3. IoT Parking Spot Monitoring (Simulated):

- **Real-Time Sensors (Simulated)**: Simulate the integration of IoT sensors that report parking spot status (occupied, available) to the central database.
- **Event-Driven Architecture**: Trigger updates to the database whenever a parking spot is occupied or becomes available. Use asynchronous notifications to update the front-end in real-time.

4. Reporting and Analytics:

- **Admin Reports**: Create dashboards for parking lot managers, showing real-time data on occupancy rates, revenue, and violations (e.g., overstay penalties).
- **Top Users and Lots**: Display reports for system administrators, like top users who reserved the most spots and top parking lots by revenue.

5. Notifications and Alerts:

- **User Notifications**: Notify users of reservation confirmations, parking time nearing expiry, and penalties.
- **Admin Alerts**: Notify parking lot managers of any issues such as over-occupied lots, unpaid reservations, or faulty spots.

6. Authentication and Authorization:

- **Role-Based Access:** Three roles with different permissions:
  - **Drivers**: Can search, reserve, and pay for parking spots.
  - **Parking Lot Managers**: Can manage parking lots, update spot status, and view reports.
  - **System Administrators**: Full control over system operations, user management, and global reporting.

7. Advanced Transactions and Concurrency Control:

- implement a transactional model for reservations and payments to ensure consistency, especially in high-concurrency environments (multiple users trying to book the same spot).
  - **Concurrency control**: Handle conflicts where multiple users attempt to reserve the same spot, ensuring that no double booking occurs.

8. Scalability and Performance Considerations:

- **Indexing and Query Optimization**: Optimize the database for high-volume read and write operations, especially for querying parking spot availability.

3. Technological Stack & Deliverables

1. Database:

- Design the ERD and develop the database generation scripts (SQL).
- Focus on triggers, stored procedures, and constraints for ensuring data consistency (e.g., no double booking of the same spot).

2. Application:

- Use **Java** or **SpringBoot** with **Angular** or **React** for the web interface or any appropriate technology stack.
- **Tables, queries, triggers, functions, and transactions** should be written directly in **native MySQL**, without the use of libraries or tools that generate SQL or manage transactions.
- Develop a REST API for web integration (e.g., a companion web application for users to search and reserve spots).

3**. Performance Testing:**

- Use **Apache JMeter** or **Gatling** to test the system's performance under different loads (e.g., 100, 500, 1000 users).

4**. Reporting:**

- Use **Jasper** or any reporting tool to generate reports on parking lot performance, occupancy rates, and top users.

5**. IoT Simulation:**

- Create a simple simulator (can be in Python) that periodically updates the status of parking spots (available or occupied), mimicking real-time IoT devices.

4. Evaluation Criteria:

- **Database Design**: Correctness and efficiency of the ERD and SQL scripts.
- **Functionality**: How well the system meets the described functional requirements.
- **Concurrency & Transactions**: Ability to handle multiple concurrent reservations correctly.
- **Performance**: Response time and scalability, especially under load.
- **Documentation**: Detailed report on database tuning, concurrency control, and performance optimization strategies.
- **Innovation**: Additional features or improvements introduced by students, such as creative ways to improve user experience or system performance.

5. Guidelines

- **Team Composition**: Work in groups of four students.
- **Report Deliverables: (note also there are application deliverables in "Technological Stack & Deliverables")**
  1. **Introduction:**
     - Overview of the Smart City Parking Management System, its goals, and the technology stack used.
  2. **Database Design:**
     - Include a readable ERD diagram (Simple MySQL generation needs to be adjusted to become user-readable) and describe the database schema.
     - You are required to provide a database generation script (SQL files), the source code of the application.
     - Discuss triggers, stored procedures and constraints to ensure data integrity (e.g., preventing double booking).
  3. **Performance Optimization:**
     - Describe database tuning (e.g., indexing).
  4. **Simulation:**
     - Detail the IoT parking spot simulation and event-driven architecture for real-time updates.
  5. **Performance Testing:**
     - Provide results from performance testing under different loads (e.g., using JMeter).
  6. **Reporting (Jasper):**
     - Describe the use of Jasper Reports to generate parking lot performance reports (e.g., occupancy rates, top users).
  7. **Screenshots and Functionality:**
     - Include screenshots demonstrating key system functionalities (real-time updates, dynamic pricing, reservations, etc.).
  8. **Conclusion:**
     - Summarize key findings and highlight any innovative features added.