Bahria University, Islamabad

Department of Software Engineering

Object Oriented Programming Lab

(Fall-2024)

Teacher: Engr. Raheela Ambrin

Student: Hasan Zahid & Waleed Ahmed Razzaqi

Enrollment: 01-131232-028 & 093

Project Report: Parking Management System

Date: January 1, 2025

GitHub Repository Link:

https://github.com/hasanzafzal/OOP-Semester-Project

Comments:
Signature

LAB

Abstract:

- ### Objectives

  The Parking Management System aims to provide an efficient solution for managing parking spaces by automating vehicle registration, slot allocation, and ticketing processes.

- ### Methodology

  The system is developed using Java and JavaFX for the front end, providing a user-friendly graphical interface. An SQLite database serves as the backend for storing and managing data related to vehicles, parking slots, and tickets. The application adheres to an MVC-like architecture, ensuring modular and scalable design.

- ### Results

- Successfully manages parking operations through features like vehicle registration, tracking entry/exit times, and monitoring slot availability.
- Ensures data integrity with a structured database comprising Vehicles, ParkingSlots, and Tickets tables.
- Provides an interactive GUI for easy user interaction and system control.

- ### Conclusion

  The Parking Management System effectively streamlines parking operations, reducing manual effort and increasing accuracy in slot management and record-keeping. It is a scalable and practical solution for small to medium-sized parking facilities.

Table of Contents:

Introduction:

Problem Statement

Modern urban environments face significant challenges in managing parking spaces due to increasing vehicle density and limited parking infrastructure. Inefficient parking management results in wasted time, congestion, and a negative user experience. Traditional manual systems are error-prone, lack scalability, and fail to provide real-time tracking or analytics.

Objectives of the Project

The Parking Management System aims to address these challenges by providing:

1. An automated solution for vehicle registration, slot allocation, and ticketing.
2. A user-friendly interface for administrators and users to interact with the system.
3. Real-time monitoring of parking slots to optimize space utilization.
4. Reliable record-keeping and reporting for operational transparency.

Scope and Limitations

- Scope: The system is designed for small to medium-sized parking facilities and supports two-wheeler and four-wheeler parking. It manages vehicle details, parking slot status, and generates parking tickets with entry/exit time tracking.
- Limitations: The system is standalone and relies on SQLite for database management, making it less suitable for larger facilities requiring a distributed or cloud-based solution. It also lacks advanced features like integration with payment systems or mobile applications.
- Significance of the Project: The Parking Management System simplifies parking operations by replacing manual processes with an automated, efficient, and reliable system. It minimizes human errors, reduces administrative workload, and enhances user experience by providing a streamlined and transparent parking management solution. This system can significantly benefit parking facility operators and users, particularly in urban areas with high vehicle density.

System Analysis

Requirement Analysis

The Parking Management System is designed to streamline parking operations by automating the process of slot allocation, vehicle tracking, and data management. Below are the identified requirements:

- Functional Requirements:
    - Record vehicle details (vehicle number, type, owner's name, contact information).
    - Allocate parking slots dynamically based on availability.
    - Display the status of parking slots (occupied/free).

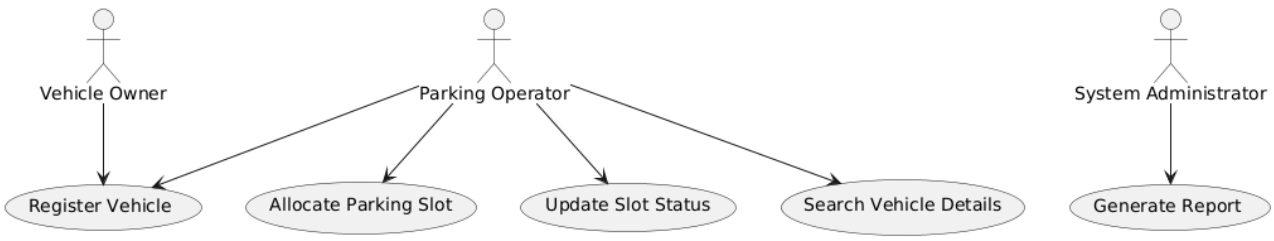
- Generate reports for parked vehicles.
- Provide a user-friendly interface for operators.

- Non-Functional Requirements:
  - Scalability to handle increased vehicle volumes.
  - System reliability and fault tolerance.
  - Quick response time for slot allocation and queries.
  - Data integrity and security, especially for payment and vehicle data.

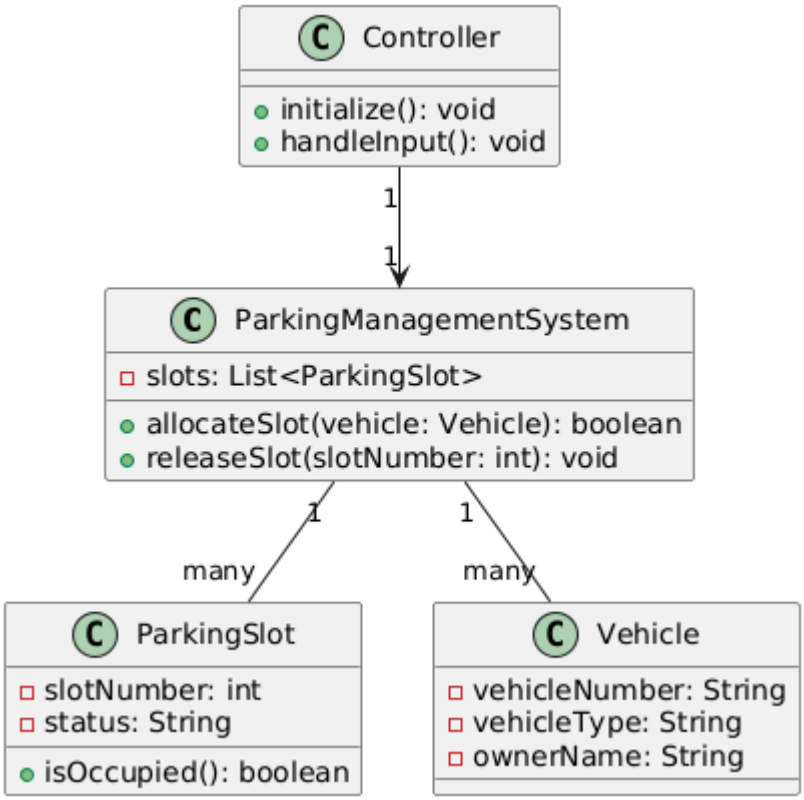
➢ Feasibility Study

1. **Technical Feasibility**:
   - The system is developed using Java and JavaFX, ensuring robust performance for desktop applications.
   - Database management is achieved using SQLite, which offers lightweight and efficient data storage.
   - The system can integrate easily with other hardware like barcode scanners or payment terminals if required.
2. **Operational Feasibility**:
   - The system is intuitive and requires minimal training for operators.
   - The use of JavaFX ensures that the UI is modern and interactive, making it easy for users to navigate.
   - Operational cost savings are significant as the system automates routine tasks like manual slot allocation.
3. **Financial Feasibility**:
   - Minimal development costs due to the use of open-source tools (Java, SQLite).
   - The system can reduce overhead costs by eliminating manual labor for tracking and slot allocation.
   - The potential for increased revenue through better space utilization and faster customer turnover.

➢ Use Case Diagram:



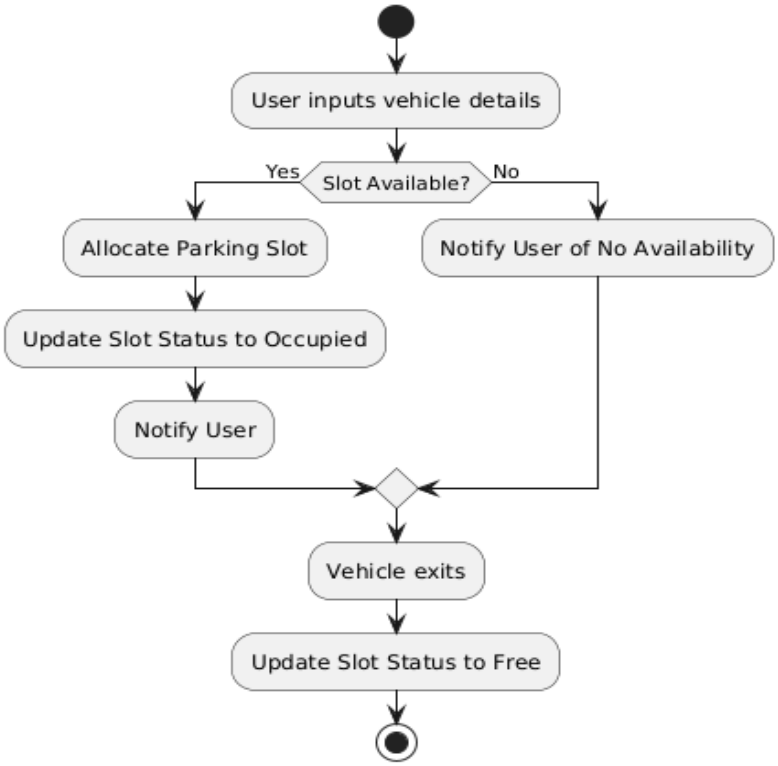
➢ Class Diagram:



System Design

➢ Architectural Design:

A flowchart that outlines the parking system's operational flow, from vehicle entry to exit.



➢ Database Design:

The database for the Parking Management System includes the following tables:

1. Vehicle
2. Parking Slot
3. Transaction

➢ User Interface Design

1. Login Page: Allows operators or administrators to log in.
2. Dashboard: Displays parking slot statuses and quick actions (e.g., assign slot, release slot).
3. Vehicle Entry Form: Collects details like vehicle number, type, and owner details.
4. Parking Slot View: Visual representation of available and occupied slots.
5. Reports Page: Displays and exports data on parked vehicles and system utilization.

Implementation

➢ Programming Languages and Tools Used

1. Programming Language:
    - Java: Core programming language for application logic and object-oriented design.
    - SQL: Used for database interactions with SQLite.
2. Frameworks and Libraries:
    - JavaFX: For building the user interface.
    - Maven: Dependency management and project building.
    - SQLite: Lightweight database for storing parking data.
3. Tools:
    - IntelliJ IDEA 2024.3.1
    - SceneBuilder
    - MySQL

➢ Modules or Components Description

1. Parking Slot Management:
    - Tracks and updates the status of parking slots (occupied or free).
    - Allocates slots dynamically based on availability.
2. Vehicle Registration:
    - Records details like vehicle number, type, and owner information.
    - Ensures that data is stored securely in the SQLite database.

3. Transaction Handling:
    - Logs entry and exit times of vehicles.
    - Manages transactions for generating reports and tracking history.
4. User Interface:
    - Provides a dashboard with options for vehicle registration, parking slot allocation, and status viewing.
    - Ensures usability and intuitive navigation using JavaFX components.
5. Database Integration:
    - Handles communication between the application and the SQLite database.
    - Supports CRUD operations (Create, Read, Update, Delete) for parking data.

➢ Code Snippets:

Complete Code is Available on GitHub.

Results and Discussion

➢ Outcomes of the Project

1. **Improved Parking Management**:
    - The system automates parking slot allocation, reducing manual errors.
    - Real-time tracking of parking slot availability ensures efficient use of space.
2. **User-Friendly Interface**:
    - JavaFX-based UI provides an intuitive and responsive experience for operators and administrators.
    - Features like dropdown menus, status tables, and notification dialogs simplify interactions.
3. **Data Integrity and Reporting**:
    - Integration with SQLite ensures secure and efficient data storage.
    - The system generates comprehensive reports, making it easier to analyze parking usage trends.
4. **Scalability**:
    - Designed to handle an increasing number of parking slots and vehicles as the business grows.

➢ Challenges Faced

1. Database Integration:
    - Ensuring smooth interaction between the application and the SQLite database, particularly with concurrent user operations, was challenging.
    - Parking Slots in database was difficult to implement hence it is hardcoded in the database.
2. User Interface Design (GUI):
    - Balancing a feature-rich interface with simplicity and practical usability for operators with minimal training was also challenging.
    - Integrating databases with Gui was also difficult and wasn't implemented fully as planned.
3. JavaFX:
    - Use of JavaFX was necessary for Gui of parking management system in java and it was especially challenging to maintain consistency between main.java, controller.java, maincontroller.java and hello-view.fxml files.

➢ Comparison with Existing Solutions

1. Real-time Slot Tracking
    - Our Project: Provides real-time updates on parking slot availability, ensuring efficient space utilization.
    - Existing Solutions: Often require manual updates or periodic synchronization, leading to delays in tracking.
2. Ease of Use
    - Our Project: Features an intuitive user interface built with JavaFX, suitable for operators with minimal technical expertise.
    - Existing Solutions: Usability varies; some systems have complex interfaces that require extensive training.
3. Cost
    - Our Project: Low cost, as it leverages open-source technologies like Java and SQLite.
    - Existing Solutions: Proprietary solutions can be expensive, with licensing fees and hardware requirements.
4. Scalability
    - Our Project: Designed to accommodate an increasing number of parking slots and vehicles, ensuring long-term usability.
    - Existing Solutions: Some legacy systems have limited scalability, requiring significant upgrades for expansion.

5. Reporting and Analytics
    - Our Project: Generates detailed reports, enabling better analysis of parking trends and usage patterns.
    - Existing Solutions: Reporting features may be basic or absent, limiting data-driven decision-making.

Advantages: The proposed system is affordable, user-friendly, and designed for scalability, making it suitable for small to medium-sized parking facilities.

Limitations: Existing systems might offer advanced features like mobile app integration or IoT-based parking sensors, which are beyond the scope of the proposed system.

Conclusion and Future Work

➢ Summary of Achievements

1. Automation of Parking Management:
   The system successfully automates critical tasks like slot allocation, vehicle registration, and reporting, reducing manual effort and improving efficiency.
2. User-Friendly Interface:
   The JavaFX-based UI is intuitive and responsive, ensuring that operators can easily navigate and utilize the system.
3. Reliable Data Management:
   Integration with SQLite ensures secure and efficient storage of vehicle and parking slot data, with the capability to generate detailed reports.
4. Scalability and Affordability:
   The use of open-source tools (java, JavaFX, etc.) and modular design ensures that the system is cost-effective and scalable for future expansion.

➢ Suggestions for Improvements or Future Enhancements

1. Mobile Application Integration:
   Develop a mobile app to allow users to check slot availability, reserve slots, and receive notifications about parking status.
2. IoT Integration:
   Incorporate IoT devices like sensors to automatically detect vehicle presence and update the system in real time.
3. Payment Gateway:
   Add an online payment module to handle parking fees, offering multiple payment options (e.g., credit cards, digital wallets).
4. Advanced Reporting and Analytics:
   Implement machine learning algorithms to analyze parking trends and provide predictive insights for demand management.

5. Cloud Integration:
   Shift to a cloud-based architecture for enhanced scalability, enabling multi-location management and remote access.
6. Enhanced Security Features:
   Introduce advanced authentication mechanisms, such as biometric or two-factor authentication, to secure access.
7. Vehicle Image Capture and Recognition:
   Implement a module to capture vehicle images and recognize license plates for automated registration and verification.

References:

- JavaFX
- SQLite Documentation
- Inspiration - https://github.com/mhamzap10/Parking-Management-System