Parking Lot Management System

Name: Eric Reyes

Course: INFO-C450

Date: 05-07-25

Table of Contents

Customer Problem Statement

Finding parking in busy zones like shopping malls, sports stadiums, and office complexes is a common frustration for many drivers. The lack of visibility into available parking spaces often results in wasted time searching for a spot. In addition, parking lots frequently reach full capacity, making it difficult to find a space. Furthermore, the payment process can be slow and inefficient, causing delays during exit. The **Parking Lot Management System** is designed to address these issues by providing real-time information on parking availability and automating ticketing and payment processes.

Brief Description of Functionality

The **Parking Lot Management System** will offer the following features:

1. **Real-time Parking Availability**: A system to show the real-time availability of parking spaces, categorized by vehicle type (e.g., compact, handicapped, motorcycle).
2. **Ticketing System**: A ticket is issued upon entry, tracking the parking time for each vehicle.
3. **Automated Payment Options**: At the exit, users can pay automatically through a payment terminal or manually via a parking agent.
4. **Display Board**: A real-time display board showing the number of available parking spots in each category.
5. **Capacity Control**: The system will prevent new entries when the parking lot reaches full capacity.
6. **Multiple Payment Methods**: Users will be able to pay with credit/debit cards or cash.

Objectives of the System

1. **Efficient Parking Management**: Automate the allocation of parking spaces based on vehicle type and availability to minimize manual management and maximize space usage.
2. **User Convenience**: Provide drivers with real-time updates on available parking spots, reducing time spent searching for a parking space.
3. **Streamlined Payment Process**: Offer both automated and manual payment options for faster and more efficient transactions during exit.
4. **Scalability**: Ensure that the system can scale to accommodate different sizes of parking lots, from small venues to large-scale events.
5. **Cost Efficiency**: Minimize operational costs through automation of space allocation, ticketing, and payment processes.

Typical Customers for the Proposed System

1. **Commuters**: Drivers seeking efficient parking solutions for shopping, work, or events. They will benefit from real-time space updates and seamless ticketing and payment.

2. **Business Owners/Managers of Parking Facilities**: Owners and managers of parking lots at commercial locations, such as shopping malls or office buildings, who need to streamline parking operations and reduce manual labor.
3. **Event Organizers**: Organizers of large events (e.g., concerts, sports games) who need a scalable parking management solution to handle high traffic and ensure smooth parking operations for attendees.
4. **Parking Lot Operators**: Companies managing parking lots who need to improve the efficiency of their operations, track parking usage, and generate insights into parking patterns.

Glossary of Terms

1. **Parking Spot Types**: Different categories of parking spaces, including:
   - **Handicapped**: Reserved spaces for vehicles with special permits.
   - **Compact**: Smaller spaces designed for compact cars.
   - **Large**: Larger spaces for bigger vehicles like SUVs and vans.
   - **Motorcycle**: Smaller spaces designed for motorcycles.
2. **Parking Ticket**: A ticket issued at the entrance that tracks the vehicle's parking time and provides a reference for payment.

3. **Automated Exit Panel**: A payment terminal at the exit where customers can pay their parking fees before leaving.

System Requirements

☐ Functional Requirements

No.	Priority Weight	Description
REQ-1	High	The parking system should allow the parking of up to 30,000 vehicles in total.
REQ-2	High	The system should include different parking spot types: handicapped, compact, large, and motorcycle spots.
REQ-3	High	There must be a real-time display of available parking spots for each parking spot type (handicapped, compact, large, and motorcycle).
REQ-4	High	The system should prevent any vehicle from entering the parking lot if it is at full capacity (30,000 vehicles).

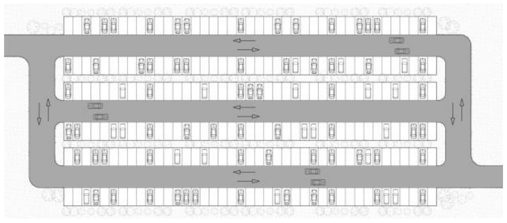
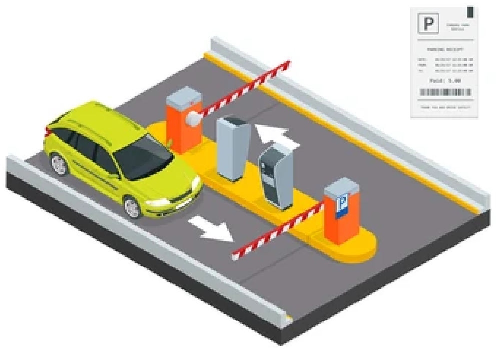
REQ-5	High	Customers should be able to receive a parking ticket upon entry, which records the time spent in the parking lot.
REQ-6	High	The system should allow customers to pay for their parking fee at the exit either through an automated exit panel or via a parking agent.
REQ-7	Medium	Payment for parking should be based on an hourly rate, with the total fee calculated based on time spent in the parking lot.
REQ-8	Medium	The parking system should support both credit/debit card payments and cash payments at the exit.
REQ-9	Medium	If the parking lot is at capacity, the entrance and the parking lot display should show a message indicating that no more vehicles can enter.
REQ-10	Low	The system should be able to automatically calculate parking fees and issue receipts upon payment.
REQ-11	Low	The system should allow the user to select a vehicle type when entering the parking lot for better spot allocation (e.g., car, truck, motorcycle).

Nonfunctional Requirements (FURPS)

No.	Priority Weight	Description
FURPS-1	High	**Functionality**: The system should accurately track the duration of parking and calculate fees based on time.
FURPS-2	High	**Usability**: The parking lot system should be intuitive and easy to navigate, with clear signage and real-time updates.
FURPS-3	Medium	**Reliability**: The system should have a high uptime and be able to handle up to 30,000 vehicles simultaneously.
FURPS-4	High	**Performance**: The system should provide real-time updates on available spots with a refresh rate of 1 second.
FURPS-5	Medium	**Supportability**: The system should be easy to maintain, with an intuitive dashboard for monitoring and troubleshooting.

User Interface Requirements

UI Requirements

1. **Parking Spot Display Board**:
   - **Priority**: High
   - **Description**: A large digital display board showing the number of available parking spots in real-time for each type of parking spot.
   - **Sketch**: 
2. **Ticket Collection Area**:
   - **Priority**: High
   - **Description**: A user-friendly interface at the entrance where customers can easily collect their parking ticket.
   - **Sketch**: 

shutterstock.com · 557834458

3. **Automated Payment Panel**:
   - **Priority**: High
   - **Description**: A streamlined payment terminal at the exit where customers can pay their parking fee via card or cash.
   - **Sketch**:

4. **Entrance and Exit Signage**:
    - **Priority**: Medium
    - **Description**: Clear signs at the entrance and exit indicating the status of the parking lot
    - **Sketch**:

Use Case 1: Vehicle Entry and Ticket Generation

This use case involves the customer entering the parking lot and the system generating a ticket for the vehicle.

State Actors and Objects:

- **Actor:** Customer
- **Objects:** Entry Panel, Ticket, Camera, Parking Database

Steps for vehicle entry and ticket generation:

1. The customer approaches the entry panel.
2. The entry panel checks for available parking spaces in the parking lot by querying the parking database.
3. The camera takes a picture of the vehicle's license plate for tracking purposes.
4. If spaces are available, the entry panel generates a ticket for the vehicle and displays it.
5. The customer receives the ticket, and the parking lot database records the entry time and vehicle ID.

System Sequence Diagram for Vehicle Entry and Ticket Generation:

```
Customer      Entry Panel       Camera       Parking Database       Ticket
   |              |              |                  |       |
   |   1. Approaches the panel     |                  |       |
   | ------------> |              |                  |       |
   |              |   2. Check for available spaces     |       |
   |              | <----------- |                  |       |
   |              |   3. Capture license plate          |       |
   |              | ------------> |                  |       |
   |            |            |      4. Store vehicle data  |
   |            |            | ------------->      |       |
   |            |   5. Generate ticket and display    |       |
   |            | <------------ |                  |       |
   |            |            |                  |       |
   |  6. Receive ticket             |                  |       |
```

Use Case 2: Parking Space Exit and Fee Payment

This use case involves a customer exiting the parking lot after parking and making the payment for their parking.

State Actors and Objects:

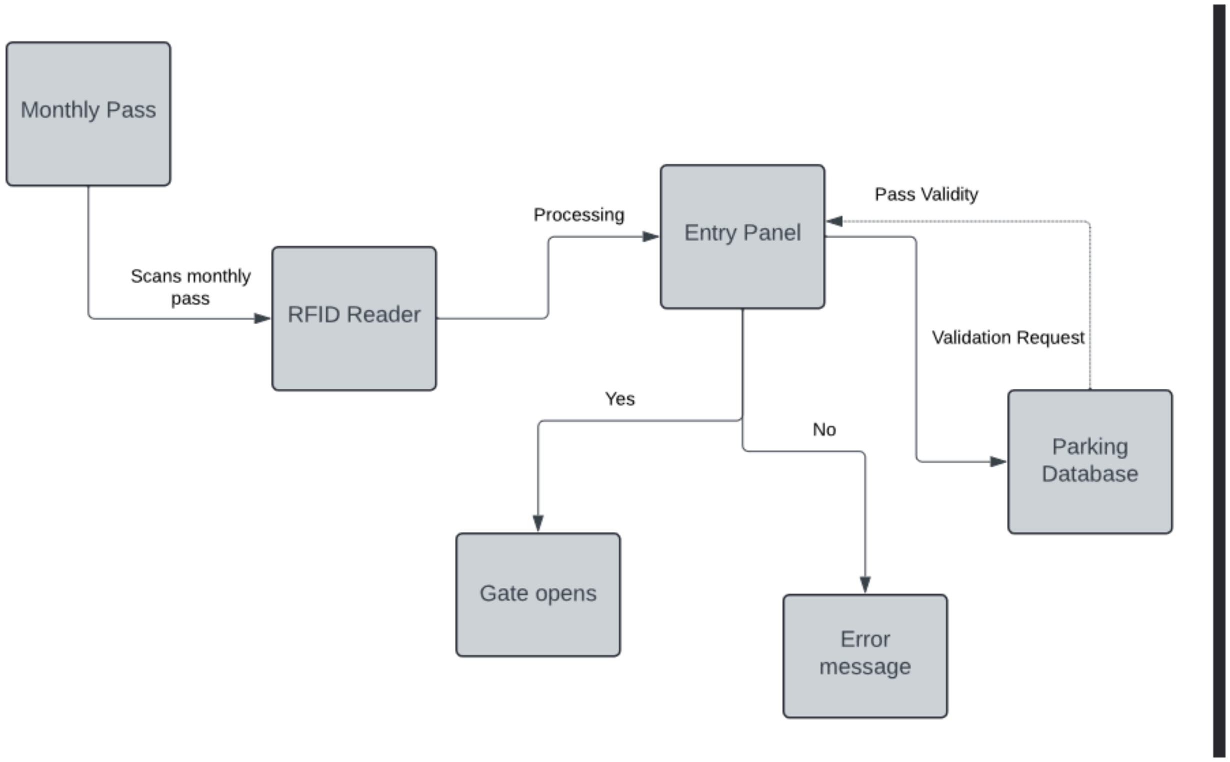
- **Actor:** Customer
- **Objects:** Exit Panel, Payment System, Ticket, Parking Database

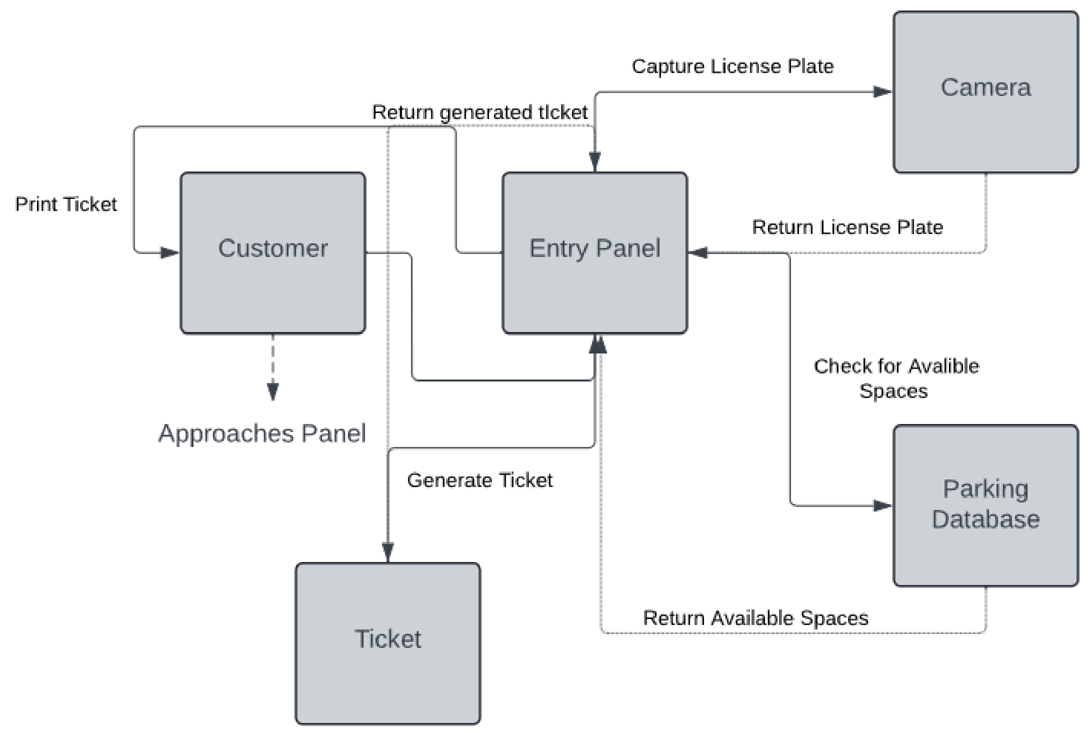
Steps for parking space exit and fee payment:

1. The customer approaches the exit panel with the parking ticket.
2. The exit panel reads the ticket and calculates the parking fee based on entry and exit times.
3. The customer is prompted to make the payment (cash or card).
4. The payment system processes the payment.
5. If the payment is successful, the exit panel opens the gate.
6. The parking database records the exit time and updates the available parking spaces.
7. If the payment is unsuccessful, the customer is notified and asked to try again or use another payment method.

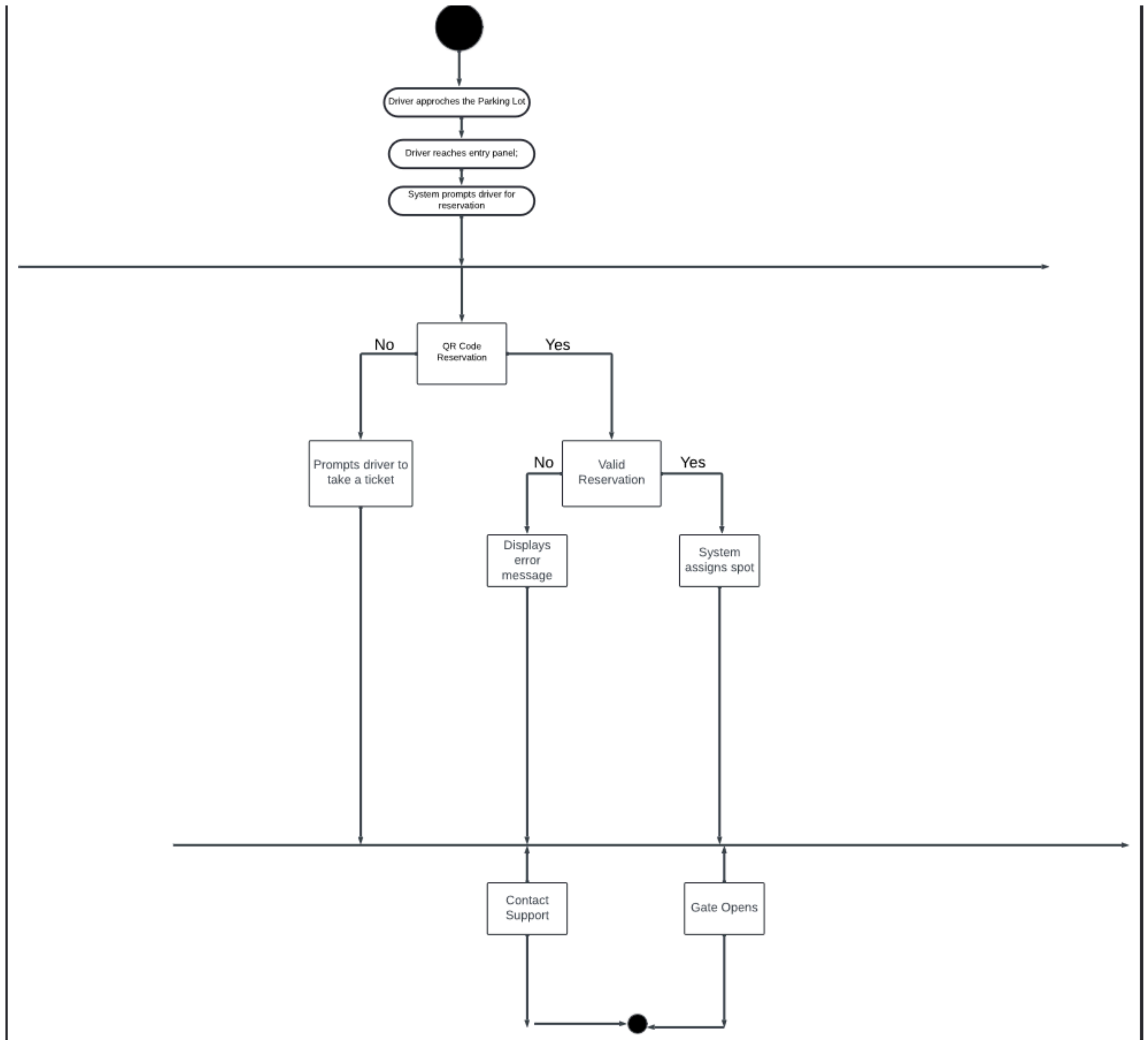
System Sequence Diagram for Parking Space Exit and Fee Payment:

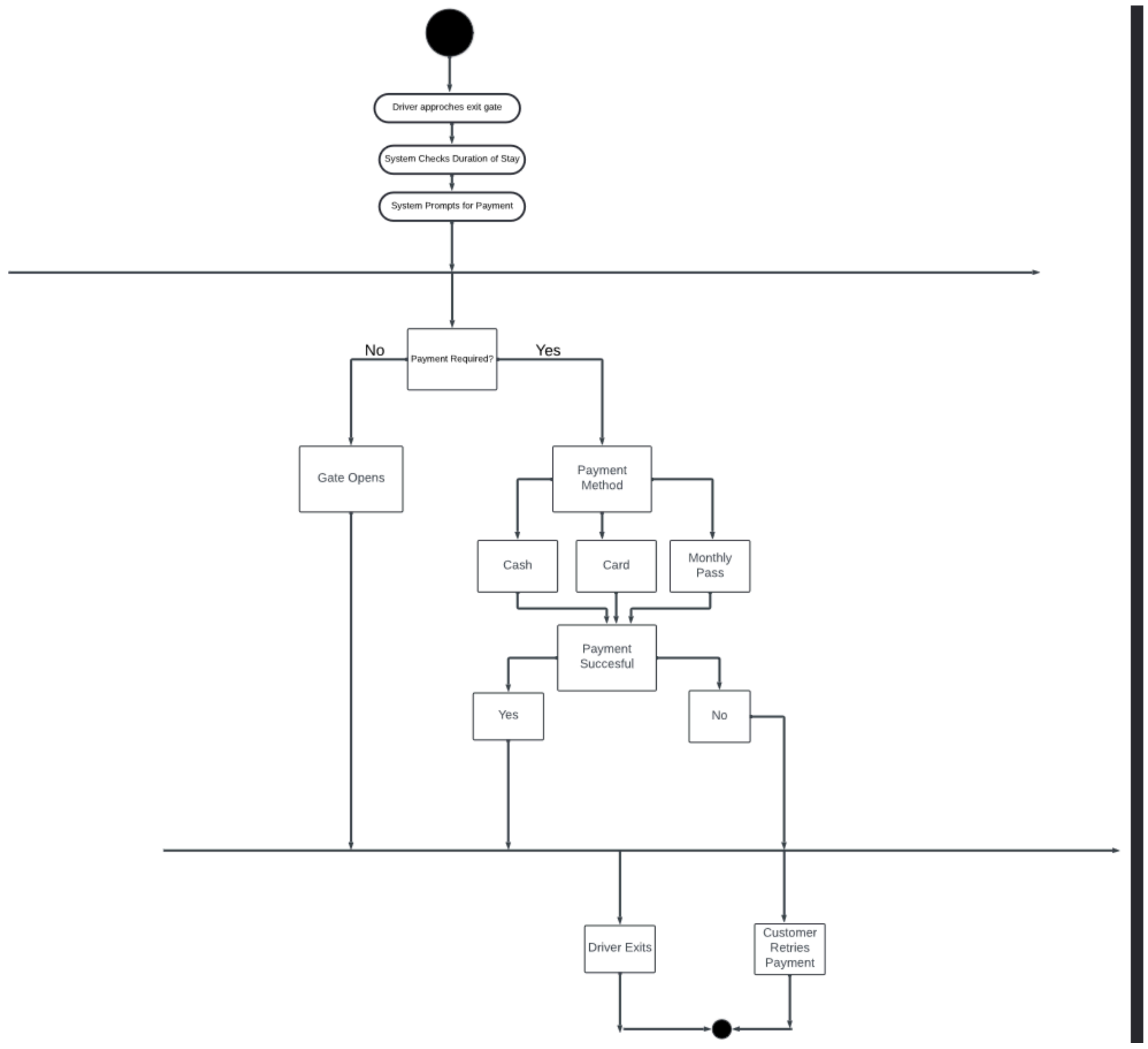
```
Customer      Exit Panel      Payment System     Parking Database       Ticket
    |            |               |                 |       |
    |  1. Approaches exit panel       |                 |       |
    | ------------> |               |                 |       |
    |            |   2. Read ticket and calculate fee    |       |
    |            | <------------    |                 |       |
    |            |               |      3. Prompt payment |       |
    |            | ------------>     |                 |       |
    |            |               |   4. Process payment  |       |
    |            |               | <------------        |       |
    |            |               |   5. Payment successful?|      |
    |            |               | ------------>        |       |
    |            |               |                 |       |
|            |   6. Open gate and allow exit      |       |
|            | <------------    |                  |       |
|            |              |   7. Update parking data |    |
|            | <------------    | ------------>       |       |
|            |              |                 |
```





Activity Diagrams


Driver approches the Parking Lot
Driver reaches entry panel;
System prompts driver for reservation
QR Code Reservation
No
Yes
Prompts driver to take a ticket
Valid Reservation
No
Yes
Displays error message
System assigns spot
Contact Support
Gate Opens



Project Planning

- **Software Requirements**:
  - **Frontend**: HTML, CSS, JavaScript.
  - **Backend**: Java (for handling server-side logic).
  - **Database**: MySQL or PostgreSQL (can be integrated using Java).
  - **Version Control**: Git and GitHub for code collaboration.
- **Hardware Requirements**:

    - Servers to host the backend and database.
    - Digital display boards for parking spot availability.
    - Automated payment terminals at exits.
- **Network Requirements**:
    - Secure internet connection for real-time updates and communication with the database.
    - Cloud hosting or local server for system deployment.

Development Approach

- **Frontend**:
    - **CSS** will be used to style the user interface and create a clean, responsive layout.
    - **JavaScript** will be used to handle interactions on the frontend, such as showing available parking spots in real-time and managing the ticketing process.
- **Backend**:
    - **Java** will be used to build the backend logic for handling parking space allocation, ticketing, and payment processing. It will also manage communication with the database.
    - **Java's Spring Boot** framework can be used to help with backend development, especially for API creation and data management.
- **Database**:
    - **MySQL** or **PostgreSQL** can be used for storing parking records and transaction data, with Java handling interactions via JDBC (Java Database Connectivity).
- **Version Control**:
    - Git and GitHub will be used to manage the project, enabling seamless collaboration with version tracking and code reviews.
- **Testing**:
    - Unit tests for both the frontend (using JavaScript testing frameworks) and backend (using JUnit for Java) to ensure each component works as expected.

Development Plan

- **Week 1-2: System Design and Setup**
    - Create wireframes for the UI and design the database schema.
    - Set up the development environment (backend in Java, frontend with HTML/CSS/JavaScript, database setup).

- **Week 3-4: Frontend and Backend Development**
  - Implement the user interface using CSS and JavaScript (parking availability dashboard, user ticketing interface).
  - Begin backend development in Java (handling parking allocation, issuing tickets, and payment processing).
- **Week 5-6: Integrating Payment System and Real-time Updates**
  - Integrate a payment system into the backend and implement real-time updates for parking spot availability.
  - Test the functionality of ticket issuance and payment processing.
- **Week 7-8: Testing and Debugging**
  - Conduct unit tests and integration tests for both the frontend and backend.
  - Perform user acceptance testing (UAT) to ensure everything works smoothly.
- **Week 9-10: Deployment and Final Testing**
  - Finalize system deployment on a secure hosting platform.
  - Conduct live environment testing to ensure real-time performance with multiple users.
- **Week 11-12: Documentation and Maintenance Plan**
  - Complete project report, including user manuals and technical documentation.
  - Develop a maintenance plan for handling future updates and user feedback.

Traceability Matrix

Requirement ID	**Requirement Description**	**Use Case**	**Design Component**	**Test Case**
REQ-1	Allow up to 30,000 vehicles	Vehicle Entry	Database schema: vehicle count check	TC-01: Max capacity entry test
REQ-2	Real-time display of spots	Parking Status Display	Frontend Dashboard,	TC-02: Display update interval test

			Backend Spot Monitor	
REQ-3	Ticket issued on entry	Vehicle Entry & Ticket Generation	Entry Panel Module	TC-03: Ticket content verification
REQ-4	Automated/manual payment	Exit & Fee Payment	Payment Gateway, Agent Terminal	TC-04: Card/Cash payment flow test
FURPS-4	1-second refresh rate	Availability Display	Frontend polling & DB update logic	TC-05: Refresh interval consistency

---

System Architecture and System Design

Architecture Diagram:

Layers:

- **Presentation Layer (Frontend):**
    - HTML/CSS/JS
    - Dynamic availability display
    - Ticket generation and payment UI
- **Application Layer (Backend):**
    - Java
    - Business logic: space allocation, payment processing

- **Database Layer:**
    - MySQL/PostgreSQL
    - Tables: `vehicles`, `tickets`, `parking_spots`, `transactions`

System Components:

- **Entry Controller:** Detects vehicles, issues tickets, checks space.
- **Exit Controller:** Handles ticket scan, calculates fee, processes payment.
- **Spot Tracker Service:** Updates spot availability in real-time.
- **Display Board Controller:** Pushes updates to digital boards.
- **Payment Processor:** Integrates with card and cash systems.

User Interface Design and Implementation

UI Components:

1. **Entry Panel UI:**
    - Button: "Enter Parking"
    - Label: “Spaces Available: [type]”
    - Output: Ticket with entry time and vehicle type
2. **Availability Dashboard:**
    - Live display (e.g., tables or cards) showing spot counts by type
    - Color-coded (e.g., red = full, green = available)
3. **Exit Panel UI:**
    - Input: Ticket ID

    - Display: Fee to pay
    - Payment options: Buttons (Card, Cash)
    - Output: Receipt + “Thank you” message

Implementation Tools:

- **HTML/CSS**: Structure and style
- **JavaScript**: Real-time updates via AJAX
- **MySQL**: Store data like tickets, vehicle types, and transactions

---

Design of Tests

Unit Tests:

Test Case ID	Component	Description	Expected Result
TC-01	EntryController	Test ticket generation when spots are available	Ticket created and stored
TC-02	ExitController	Test fee calculation for 2 hours	Correct fee returned
TC-03	SpotTrackerService	Update spot count after vehicle exit	Spot count incremented
TC-04	PaymentProcessor	Test card payment flow	Success message & receipt

TC-05	Database Connection	Test DB connection pooling	Connection established

Integration Tests:

- Simulate full vehicle entry and exit flow
- Test full capacity handling (attempt to enter 30,001st vehicle)
- Cross-check ticket issue vs database record

Conclusion

The **Parking Lot Management System** will simplify the parking experience for users while automating the management process for parking lot operators. By providing real-time parking spot availability, efficient ticketing, and fast payment options, the system aims to reduce the time and stress associated with parking. With scalability and cost-effectiveness in mind, the system is designed to meet the needs of parking lots of varying sizes, from small venues to large event spaces.