PARKING LOT MANAGEMENT SYSTEM

Elvis Ilor and Mason Hawkins

IUPUI System Design.

Jafrina Jabin.

Contents

Customer Problem Statement 1
Problem Statement 1
Glossary of Terms 2
System Requirements..3
User Interface Requirements 4
Plan of Work 5
Functional Requirements Specification..5
System Sequence Diagram and Activity Diagram 12

Customer Problem Statements & System Requirements

Customer Problem Statement

Problem Statement: In today's financial landscape, individuals and small businesses often face significant challenges in accessing credit through traditional banking systems. These obstacles include strict loan requirements, high interest rates, and prolonged approval processes, which particularly affect those with limited credit histories or unconventional income sources. As a result, underserved communities struggle to secure the financial support necessary for personal or business growth.

The P2P Lending Platform seeks to address this by providing a direct connection between borrowers and lenders, offering more flexible and inclusive credit options. Borrowers can specify their loan amount, purpose, and repayment terms, while lenders can select loan requests that meet their preferences or rely on an automatic matching algorithm based on their risk tolerance.

Additionally, the platform introduces alternative credit scoring, which uses non-traditional financial data like transaction history and utility payments to assess borrower creditworthiness. Through secure payment gateways like Stripe or PayPal, the platform ensures safe transactions, giving both parties confidence in the lending process.

Glossary of Terms

- **Borrower:** An individual or small business requesting a loan through the P2P Lending Platform.
- **Lender:** An individual or entity providing funds to borrowers through the platform, with the expectation of repayment plus interest.
- **Alternative Credit Scoring:** A system that evaluates borrower creditworthiness based on non-traditional financial data, such as transaction history, utility payments, and other indicators.
- **Loan Matching Algorithm:** A mechanism that automatically pairs borrowers and lenders based on loan amount, risk tolerance, and repayment terms.
- **Payment Gateway:** A service that processes and secures transactions between borrowers and lenders, ensuring safe money transfers.

System Requirements

Functional Requirements

No.	Priority Weight	Description
REQ-1	High	The platform should allow borrowers to submit loan requests specifying the amount, purpose, and repayment terms.
REQ-2	High	The platform should allow lenders to browse loan requests based on loan amount, risk profile, and repayment terms.
REQ-3	Medium	The system should recommend loans to lenders based on their risk tolerance through an automatic matching algorithm.
REQ-4	High	The platform should support secure transactions through payment gateways like Stripe or PayPal.
REQ-5	Medium	Borrowers without traditional credit scores should be assessed through alternative credit scoring using transaction history and utility payments.
REQ-6	High	The system should ensure that both borrowers and lenders have access to a user-friendly interface for managing transactions.

Nonfunctional Requirements

No.	**Priority Weight**	**Description**
NFR-1	High	**Functionality:** The platform should accurately match borrowers with lenders and handle secure transactions without errors.
NFR-2	Medium	**Usability:** The platform interface should be intuitive for both borrowers and lenders.
NFR-3	High	**Reliability:** The system should guarantee uptime during peak transaction periods to ensure continuous service availability.
NFR-4	Medium	**Performance:** The platform should handle multiple concurrent transactions without delays or timeouts.
NFR-5	Low	**Supportability:** The platform should allow for easy updates and maintenance to enhance features based on user feedback.

User Interface Requirements

1. **Borrower Loan Request Form:** A simple interface allowing borrowers to input the loan amount, purpose, and repayment terms. (High Priority)
   - **Sketch:** A form interface with fields for loan amount, purpose, repayment terms, and submission button.
2. **Lender Loan Selection Page:** A browse page where lenders can filter and select loans based on preferences. (High Priority)

  - **Sketch:** A filterable list of loan requests with details like loan amount, risk profile, and repayment terms.

3. **Loan Matching Dashboard:** A dashboard displaying recommended loans to lenders based on their risk profile. (Medium Priority)
   - **Sketch:** A dashboard view with recommended loan options and acceptance buttons.
4. **Transaction Summary Page:** A page summarizing ongoing and past transactions for borrowers and lenders. (High Priority)
   - **Sketch:** A table view listing all current loans, repayments, and statuses.

Plan of Work

- **Weeks 1-2:** Completed user research and created detailed wireframes for borrower and lender experiences.
- **Weeks 3-5:** Frontend and backend development is ongoing. We have started developing the borrower interface and lender loan matching functionalities using React.js and Node.js.
- **Week 6:** Currently setting up payment gateway integrations and preparing for secure user authentication.

Functional Requirements Specification

- **Drivers**: Individuals using the system to find and reserve parking spots.
- **Admin**: Responsible for overseeing parking operations, adding/removing spots, and generating reports.

- **Parking Assistants**: Facilitate check-ins, manage billing, and help customers on-site.
- **System Administrators**: Oversee the system's maintenance and ensure it operates smoothly.
- **Investors/Sponsors**: Interested in the project's success from a business and revenue perspective:
- **Drivers**: Use the system to find and reserve parking, receive notifications, and pay for parking.
- **Parking Agents**: Assist drivers, manage check-ins, and process payments.
- **Admin**: Manage parking spaces, generate reports, and oversee the system's operations.
- **System**: Provides real-time availability, manages reservations, and processes payments.

- **Add parking spot**: To add a new parking spot, including selecting the type (compact, handicapped, large, motorcycle) and location. (2 points)
- **Edit parking spot**: To modify an existing parking spot detail (type, location, availability).
- **Remove parking spot**: To delete an existing parking spot from the system. (3 points)
- **Generate reports**: To generate usage reports on parking spaces (revenue, peak hours). (3 points)
- **Login/Logout**: To log in or out of the admin account. (2 points)

- **View account**: To view account details like admin privileges or the system status. (2 points)
- **Manage staff accounts**: To create, update, or remove accounts for parking

Driver

- **View parking**: To see the real-time availability of parking spaces in the selected area. (2 points)
- **Reserve parking spot**: To reserve a parking space in advance. (2 points)
- **Receive notification**: To receive alerts when a parking spot becomes available or if a reservation is about to expire. (2 points)
- **Pay for parking**: To pay for parking through the system using credit/debit cards or digital wallets. (2 points)
- **Check out/Exit lot**: To scan the ticket at the exit and complete the parking session. (2 points)
- **Cancel reservation**: To cancel a parking reservation. (2 points)

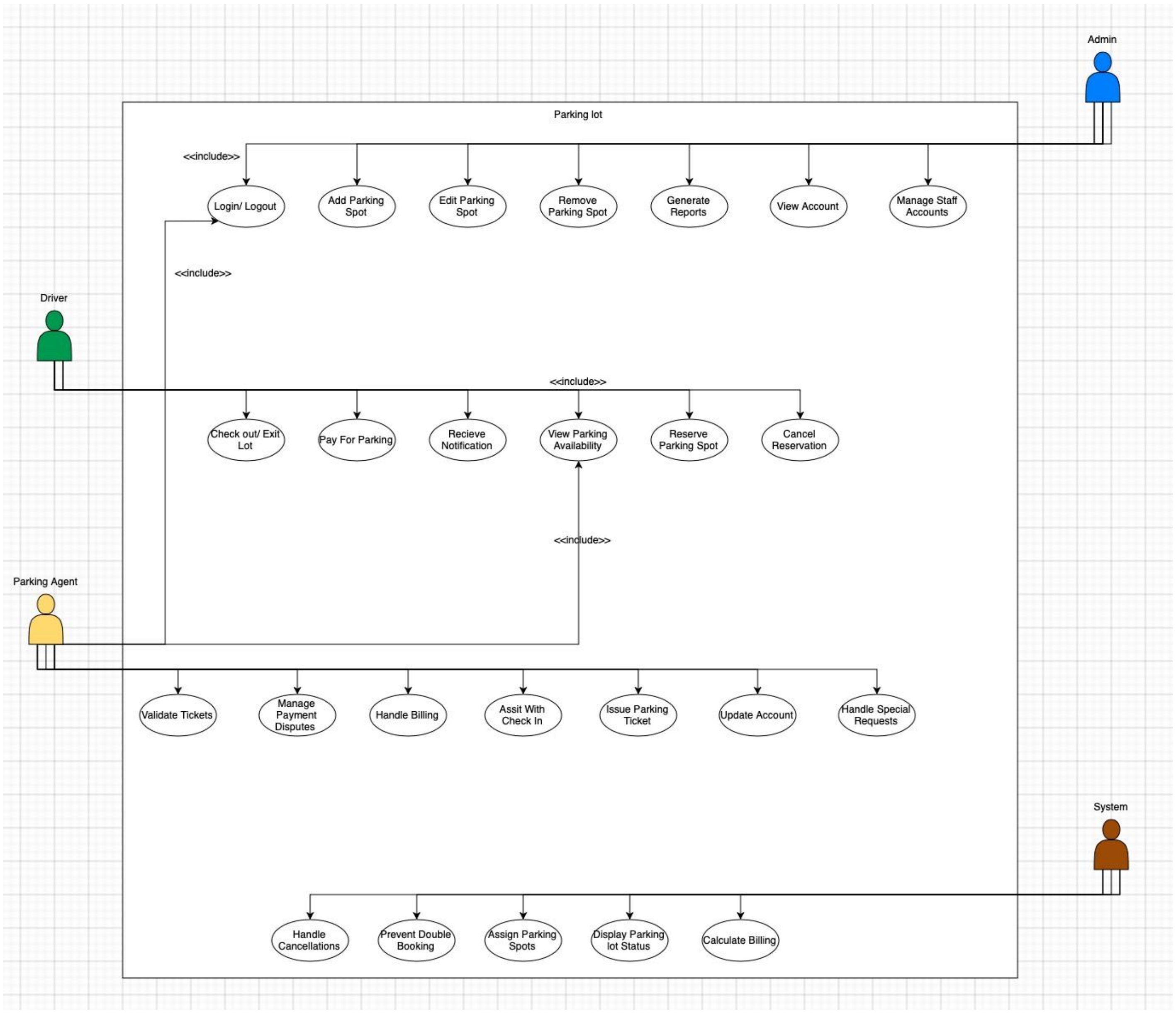
Parking Agent

- **Login/Logout**: To log in or out of the agent account. (2 points)
- **View parking availability**: To monitor parking availability across locations. (2 points)
- **Assist with check-ins**: To help drivers by scanning their reservation codes or manually assigning parking spots. (2 points)

- **Handle billing**: To assist drivers with payment processing, and accepting cash, card, or mobile payments. (2 points)
- **Issue parking ticket**: To issue parking tickets for drivers without reservations. (2 points)
- **Manage payment disputes**: To resolve payment disputes or generate invoices for later payment. (2 points)
- **Update account**: To update account details and payment information. (2 points)
- **Validate tickets**: To check the validity of parking tickets at the exit. (2 points)
- **Handle special requests**: To accommodate drivers with special requests (e.g., specific parking spots, assistance). (2 points)

System

- **Assign parking spots**: To automatically assign available parking spots based on vehicle type. (2 points)
- **Prevent double-booking**: To ensure no two drivers can reserve the same spot at the same time. (2 points)
- **Display parking lot status**: To show whether the parking lot is full or has available spots. (2 points)
- **Calculate billing**: To calculate the parking fee based on the time spent. (2 points)
- **Handle cancellations**: To process reservation cancellations and make the spot available again. (1 point)



1. **View Parking Availability**:
   - Shared by **Drivers** and **Parking Agents**. Both need to see the real-time availability of parking spaces.
2. **Reserve Parking Spot**:
   - Only **Drivers** can reserve parking spots in advance.

3. **Receive Notifications**:

   ○ **Drivers** receive alerts when a parking spot is available or when a lot is full.

4. **Pay for Parking**:

   ○ **Drivers** can complete payments using various methods like credit card, debit card, or mobile payment.

5. **Check out/Exit Lot**:

   ○ **Drivers** scan a ticket at the exit to calculate the parking fee.

6. **Cancel Reservation**:

   ○ **Drivers** can cancel reservations, and the spot is made available again.

7. **Manage Check-ins**:

   ○ **Parking Agents** assist **Drivers** with checking in by scanning reservation codes or manually assigning spots.

8. **Process Payments**:

   ○ **Parking Agents** handle payments through cash, card, or mobile payment options.

9. **Issue Parking Ticket**:

   ○ **Parking Agents** issue tickets to **Drivers** without reservations.

10. **Login/Logout**:

○ Shared by **Parking Agents** and **Admin** to access their respective accounts.

11. **Add/Edit/Remove Parking Spots**:

    ○ **Admin** can modify the availability and location of parking spots.

12. **Generate Reports**:

○ **Admin** generates reports on parking usage, revenue, and peak hours.

13. **Prevent Double-Booking**:

○ The **System** ensures that no two reservations are made for the same spot by blocking it once reserved.

14. **Show Parking Lot Status**:

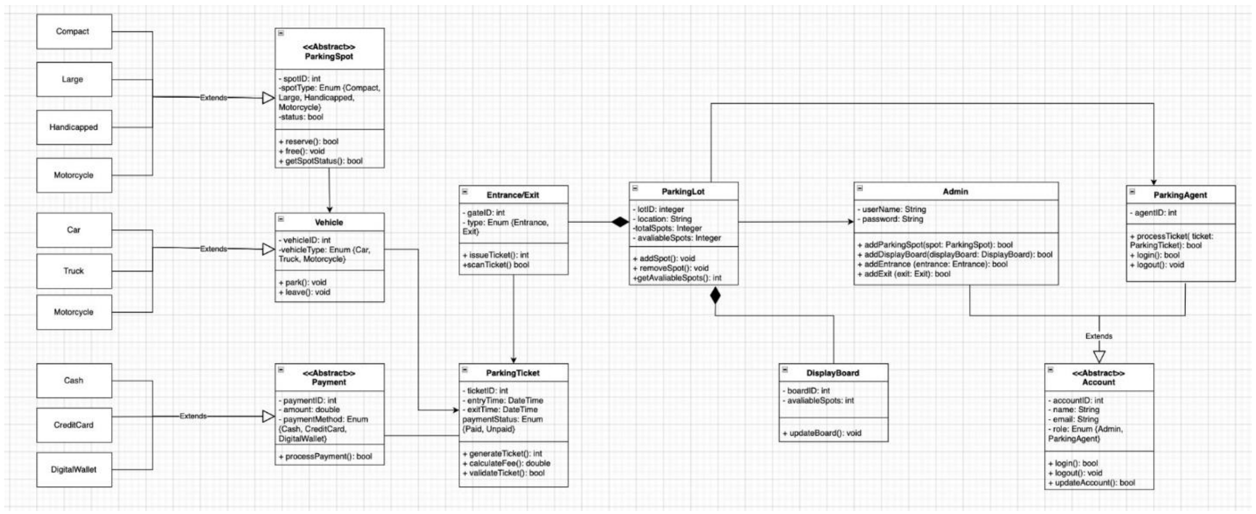
○ The **System** shows whether the parking lot is full or has available spaces for **Drivers** and **Parking Agents**.

15. **Calculate Parking Fees**:

○ The **System** calculates parking fees based on the duration of the driver's parking session.

16. **Handle Cancellations**:

○ The **System** processes reservation cancellations and updates the availability of parking spaces.



System Sequence Diagram and Activity Diagram

Overview:

We structured our System Sequence Diagrams (SSDs) and Activity Diagrams based on the key use cases outlined in the Functional Requirements Specification. Below, we detail our approach to completing the diagrams for two complex use cases, ensuring a clear representation of the system's interactions and flows.

Step 1: Identify Key Use Cases

As a team, we selected two key use cases from the specification document that are both complex and involve multiple interactions:

1. Reserve Parking Spot
    - Actor: Driver
    - Goal: Enable the driver to find and reserve a parking spot through the system.
2. Pay for Parking
    - Actor: Driver or Parking Agent

- Goal: Process parking payment either at the exit or through a payment kiosk.

Step 2: Create System Sequence Diagrams (SSDs)

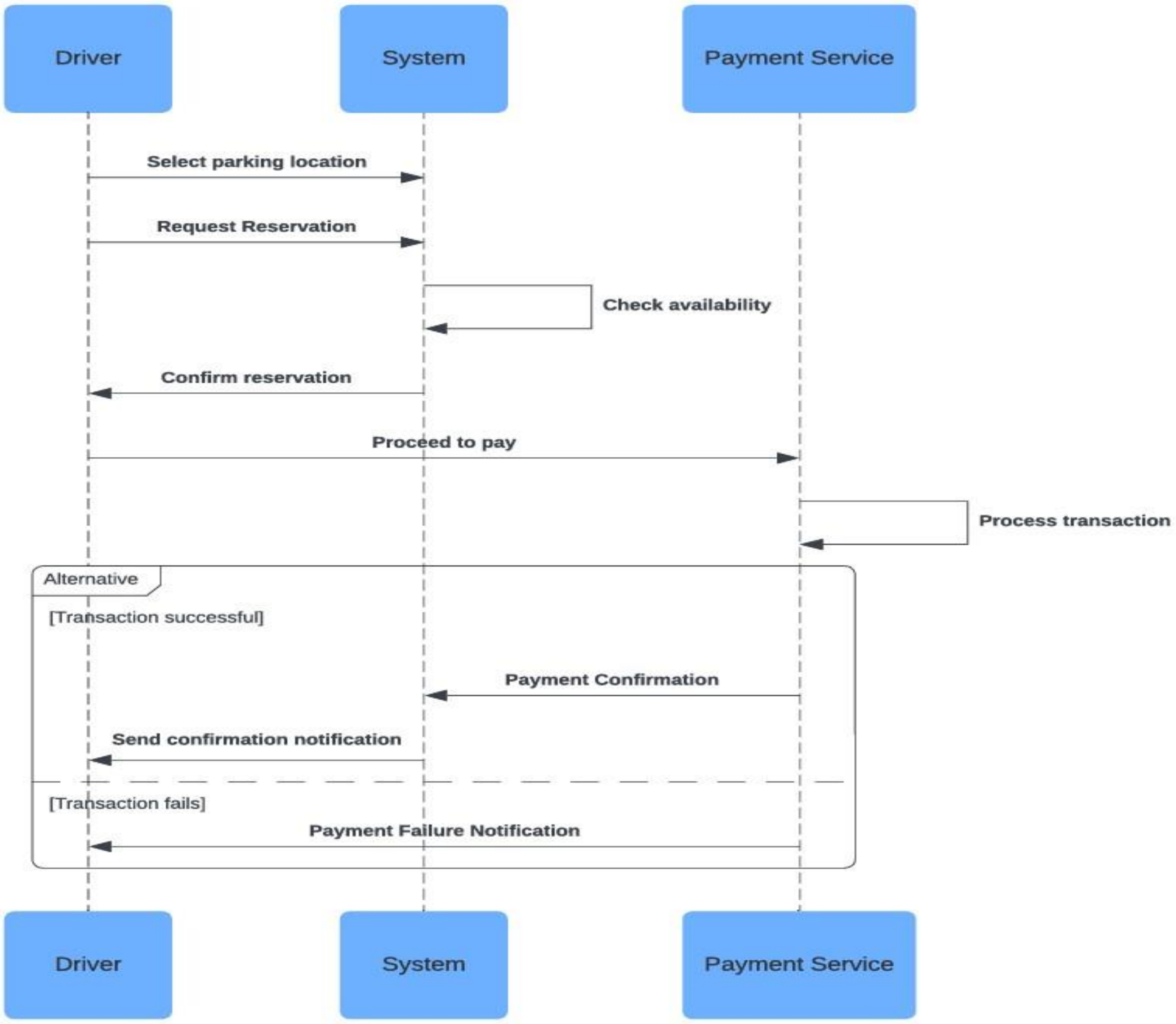
We broke down the sequence of interactions between the actor and the involved system components for each use case.

1. SSD for Reserve Parking Spot
    - Actor: Driver
    - Objects Involved: System, Payment Service, Notification System

Sequence Flow:

3. The driver selects a parking location and requests a reservation.
4. The system checks the availability of parking spaces and confirms the reservation.
5. The driver proceeds to payment.
6. The payment service processes the transaction.
7. If the transaction is successful, the system sends a confirmation notification to the driver.
8. If the transaction fails, a payment failure notification is sent to the driver.

Diagram:



1. SSD for Pay for Parking

- Actor: Driver
- Objects involved: System, Payment Service

Sequence Flow:

1. The driver inserts the parking ticket at the kiosk or exit.

2. The system calculates the parking fee.

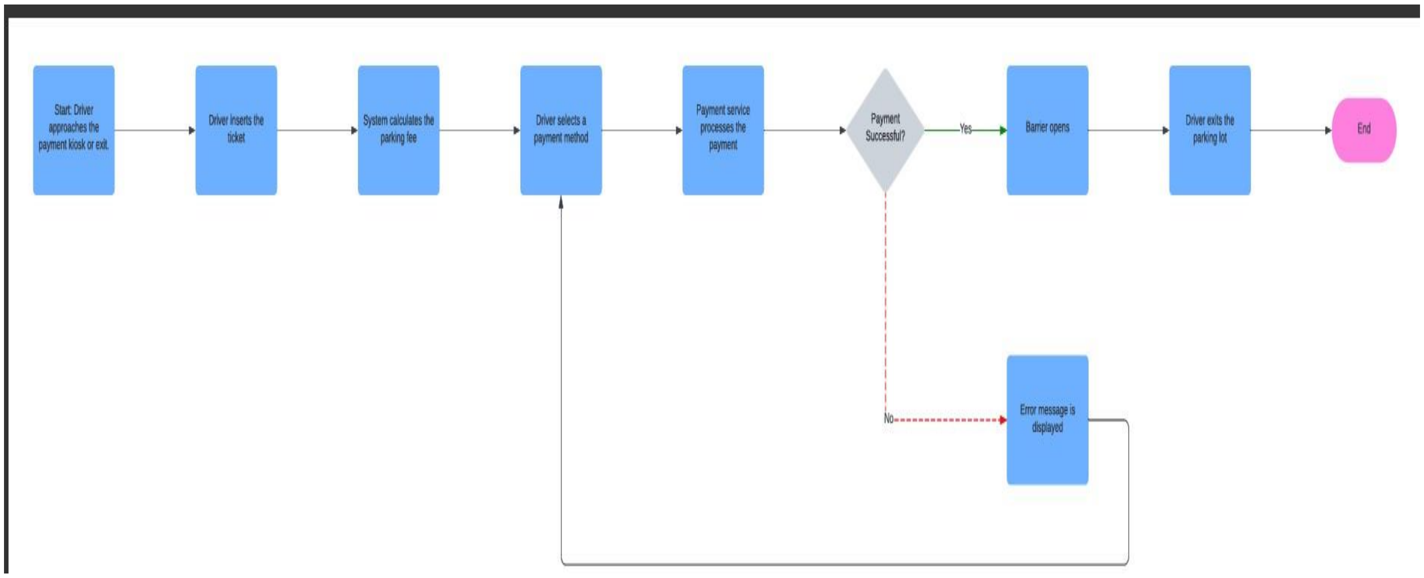
3. The driver selects a payment method (e.g., card, mobile payment).

4. The payment service processes the transaction.

5. If the payment is successful, the system opens the barrier, and the driver exits.

6. If the payment fails, an error message is displayed.

Diagram:


Start: Driver approaches the payment kiosk or exit.
Driver inserts the ticket
System calculates the parking fee
Driver selects a payment method
Payment service processes the payment
Payment Successful?
Yes
Barrier opens
Driver exits the parking lot
End
No
Error message is displayed

Step 3: Create Activity Diagrams

We used activity diagrams to illustrate the flow of events from start to finish for each use case.

Example: Activity Diagram for Pay for Parking

- Initial State: The driver approaches the payment kiosk or exit.

Actions:

1. The driver inserts their parking ticket.
2. The system calculates the parking fee.
3. The driver selects a payment method (e.g., card or mobile payment).
4. The payment service processes the transaction.
5. If the payment succeeds, the barrier opens. If not, an error message is displayed.

- Final State: The driver exits the parking lot.

Diagram: This activity diagram includes:

- Decision nodes to handle success or failure in payment (e.g., "Payment Successful? Yes/No").
- Swim lanes to separate the activities performed by the Driver, System, and Payment Service, making it easier to visualize their roles.