Parking Garage Project Phase 1

Software Requirements Specification

Revision History

Date	Revision	Description	Author
2/22/2025	1.0	Started SRS document, finished Purpose and half of overall description.	Kurt, Vishal, and Raymond
3/1/2025	2.0	Finished SRS Document and UML Class Sequence diagrams	Kurt, Vishal, and Raymond

Table of Contents

1. Purpose

This document outlines the requirements for the Parking Garage Group Project.

1.1. Scope

This document will catalog the user, system, and hardware requirements for the Parking Garage System. It will not, however, document how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

Important terms:

1.2.1. **Parking Garage Management System (PGMS):** The software and hardware system used to manage parking spaces, transactions, and customer entry/exit in a parking facility.

1.2.2. **Entry/Exit Log:** A record of vehicles entering and leaving the parking garage, including timestamps and vehicle details.

1.2.3. **Multithreading:** A technique used to allow multiple tasks to run concurrently within the same program, improving performance and efficiency.

1.2.4. **Payment Gateway:** A system that processes financial transactions, allowing customers to pay for parking services.

1.2.5. **Server:** The central computing unit that manages and coordinates the operations of the parking garage system, including handling multiple garages and transactions.

1.2.6. **Thread:** A lightweight process that performs tasks concurrently with other threads in the system, used to handle multiple customer actions simultaneously.

1.2.7. **Employee:** A worker who oversees the system and its responsibilities in managing the network of parking garages.

1.2.8. **Unified Modeling Language (UML)**: Modeling used to visualize system design.

1.3. References

Use Case Specification Document

UML Use Case Diagrams Document

Class Diagrams

Sequence Diagrams

1.4. Overview

The Parking Garage Management System (PGMS) is designed to efficiently manage parking operations across multiple parking garages. The system monitors available parking spaces, processes and logs payments, and ensures proper management of the garage's capacity. It supports various payment methods and handles concurrent customer interactions through multithreading to ensure smooth operation even during peak times. Additionally, the system provides employees with real-time data on garage occupancy, payment status, and vehicle movement. The goal is to enhance the customer experience while maintaining efficient use of parking spaces and generating revenue for the facility.

2. Overall Description

2.1. Product Perspective

2.1.1. The Parking Garage Management System (PGMS) operates as a standalone, networked software application designed to facilitate the management of a network of self-parking garages. It consists of a client-server architecture running over TCP/IP and provides an interactive graphical user interface (GUI) for customers and employees.

2.2. Product Architecture

The system will be organized into 3 key major modules:

1. **The Ticketing System:** simultaneously reserves a spot for multiple customers and checks them out
2. **The Parking Garage:** an independent facility on the server side allowing independent tracking of parking availability , payments, and vehicle logs for each location.
3. **Parking Logs:** The history or logs of all tickets, including timestamps.

These modules will follow standard Object-Oriented design principles by sending and receiving data between each other and use that to control the parking garages. For example, the ticketing module will use the parking availability information for the corresponding garage within the system to see whether a new car should be allowed in.

2.3. Product Functionality/Features

The **Parking Garage Management System** (PGMS) includes the following high-level features to ensure efficient operation, security, and a smooth customer experience:

1. **User Authentication and Access Control**
   - Secure login functionality for employees with unique usernames and passwords.
   - Only authorized employees can process payments, access usage reports, and manage parking garage operations.
   - Account locking after multiple failed login attempts to enhance security.
2. **Parking Space Monitoring and Availability**
   - Real-time tracking of available parking spaces in the garage.
   - Continuous updates to parking space availability after customer entry/exit.
   - System prevents overcapacity by ensuring that only available spaces are assigned to customers.
3. **Payment Processing**
   - Support for various payment methods, including credit/debit cards and cash.
   - Automated payment based on the duration of stay, with an option for manual payment processing by employees for customer assistance.
   - Real-time payment processing and immediate space updates after each transaction.
4. **Graphical User Interface (GUI)**
   - A user-friendly GUI for employees to manage parking garage operations.
   - Displays parking space availability, payment options, and fees.
   - Real-time updates to parking availability and payment status over TCP/IP for both server and client applications.
5. **Concurrent Client Handling**

- The system must handle multiple customers simultaneously, processing transactions in parallel using multithreading.
- Ensures that each customer's payment and space update is processed independently without delays or data conflicts.
- Proper synchronization mechanisms are in place to avoid data corruption in shared resources.

6. **Employee Login and Access Control**
   - Employees must log in with a unique ID and password for secure access to the system.
   - Only authorized employees can process payments and access usage reports.
7. **Usage Reporting for Management**
   - The system must generate daily, weekly, and monthly usage reports, including total revenue, peak times, and space utilization.
   - Tracks performance metrics, efficiency, and cost management to assist in parking garage administration and decision-making.
8. **Multi-Garage Management**
   - The server supports managing multiple parking garages at once, allowing independent tracking of each garage's operations while synchronizing data across all locations.
   - Employees can manage multiple garages and view real-time data from all locations in a unified interface.
9. **Error Handling and System Diagnostics**
   - **Error Detection**: The system will automatically detect and log errors related to payment processing, server-client communication, or system operations.
   - **Error Logging**: Errors are logged with details (e.g., type, time, transaction ID) for troubleshooting and future reference.
   - **Alerts**: Alerts are sent to employees when critical errors (e.g., payment issues or communication failure) are detected, enabling quick action.
   - **System Monitoring**: Continuous system health checks to identify and address issues with server operations, TCP/IP connections, and more.
10. **Response Time and System Performance**
    - The system must provide real-time processing including parking entry, exit, fee calculation, and payment processing.
    - Updates to parking availability and transaction data must be presented to clients with minimal delay.
    - The system must maintain responsive performance under normal and peak load conditions.

2.4. Constraints

2.4.1. Since multiple employees might be logged in at the same time, the system should ensure that transactions are processed accurately without conflicts.

2.4.2. The system must keep track of all parking transactions, including entry time, exit time, and payments, storing them in a log file for quick reference.

2.4.3. The system must handle TCP/IP errors and prevent unauthorized modifications to transaction data.

2.4.4. The system should operate continuously during parking garage business hours, and any failures should be logged with an error message for debugging.

2.5. Assumptions and Dependencies

2.5.1. The system will support a maximum of 200 concurrent users (including customers and employees) per parking garage location, based on expected usage patterns.

2.5.2. Parking entry and exit terminals will be properly configured and calibrated before deployment to ensure accurate ticket generation and tracking.

2.5.3. The system will have a stable power supply to maintain continuous operation of the parking terminals, payment stations, and server.

2.5.4. All sensors and automated barriers will be installed in strategic locations to ensure proper functionality without false detections or errors.

2.5.5. The system relies on local TCP/IP networking for communication between client terminals and the central server.

2.5.6. The accuracy of parking space availability tracking depends on proper usage by customers and enforcement by employees.

2.5.7. The system requires regular maintenance to ensure all hardware (kiosks, ticket scanners, payment terminals) remains functional.

2.5.8. Employees must be properly trained to operate the system and assist customers with payments and troubleshooting when necessary.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

3.1.1.1. Payment Method
3.1.1.2. Parking Duration
3.1.1.3. Parking Space Tracking
3.1.1.4. Concurrent Client Handling
3.1.1.5. Graphical User Interface
3.1.1.6. Employee Login
3.1.1.7. Usage Report
3.1.1.8. Multi Garage Management
3.1.1.9. Error Handling
3.1.1.10. Response Time

3.1.2. Ticketing System Requirements:

3.1.2.1. The system must generate a unique **ticket ID** for each vehicle entering the parking garage.
3.1.2.2. The ticket must store the **entry time, ticket ID, and associated parking garage ID**.
3.1.2.3. The system must automatically **reserve a parking spot** upon issuing a ticket, ensuring available space before allowing entry.
3.1.2.4. Customers must provide their ticket at exit for **fee calculation based on duration**.
3.1.2.5. The system must allow employees to manually override a ticket in case of lost or damaged tickets.
3.1.2.6. The system must prevent **duplicate ticket issuance** for a single vehicle within the same session.
3.1.2.7. The ticketing module must communicate with the **parking garage module** to verify space availability before generating a new ticket.
3.1.2.8. The system must remove the reservation upon checkout to free up the parking space.

3.1.3. Parking Garage Requirements:

3.1.3.1. Each parking garage must operate **independently** while being managed under a unified system.
3.1.3.2. The system must track the **total number of available and occupied parking spaces** for each garage.
3.1.3.3. The system must **update parking space status** in real time when vehicles enter or exit.
3.1.3.4. Each parking garage must maintain a **separate vehicle log** to track parked cars, including their ticket ID and entry time.
3.1.3.5. The system must allow for **multiple parking garages to be managed simultaneously**, keeping data separate for each.
3.1.3.6. The system must **verify ticket validity** before allowing a car to exit.
3.1.3.7. Payments must be handled per garage, ensuring that transactions are properly recorded under the correct facility.

3.1.4. Parking Logs Requirements:

3.1.4.1. The system must maintain a **persistent log of all parking transactions**, including timestamps for entry, exit, and payments.
3.1.4.2. The system must store **ticket history**, including ticket ID, entry time, exit time, and amount paid.
3.1.4.3. The log system must support **daily, weekly, and monthly reports** for garage usage statistics.
3.1.4.4. Employees must be able to retrieve **past transaction logs** based on ticket ID, date, or vehicle information.
3.1.4.5. The system must prevent unauthorized **modifications or deletions** of existing logs.
3.1.4.6. Logs must be stored in a structured format to enable **efficient searching and filtering**.
3.1.4.7. The system must provide alerts for **unresolved tickets** where a vehicle has entered but has not yet checked out.

3.2. External Interface Requirements

3.2.1. The system must be able to communicate with parking sensors to detect vehicle entry, exit, and individual parking space occupancy in real time.
3.2.2. The system should provide a graphical user interface (GUI) accessible on a standard computer for employees to monitor parking availability and process payments.
3.2.3. The system must generate parking logs that can be exported in a structured text format for external review and reporting.
3.2.4. The ticketing system must interface with automated ticket dispensers to issue tickets upon vehicle entry.
3.2.5. The payment system must support integration with external payment processing hardware (e.g., card readers, cash terminals) to accept customer payments.

3.3. Internal Interface Requirements

3.3.1. The ticketing module must communicate with the parking garage module to check space availability before issuing a ticket.
3.3.2. The parking garage module must update the parking logs module whenever a vehicle enters or exits the facility.
3.3.3. The payment processing system must retrieve ticket data from the ticketing module to calculate the correct fee.
3.3.4. The employee interface must allow authorized personnel to monitor parking availability, process transactions, and review logs with role-based access.
3.3.5. The system must allow multiple parking garages to function independently while sharing a common management interface.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1. **Access Control:** Only authorized employees can access administrative functions, while customers are limited to self-service features.

4.1.2. **Data Protection:** All transaction records, parking logs, and employee credentials must be securely stored.

4.1.3. **Secure Communication:** Client-server communication must be protected against interception or tampering using secure protocols.

4.1.4. **Audit and Logging:** All system activities must be logged, with access restricted based on user roles for employees and auditing.

4.2. Environmental Requirements

4.2.1. **Operational Reliability:** The system must function reliably withstanding varying weather conditions such as heat, cold, and humidity.

4.2.2. **Hardware Compatibility:** The user interface should be accessible on standard computers and kiosks with minimal hardware requirements.

4.2.3. **Network Dependency:** The system must operate efficiently over local TCP/IP networks, ensuring that critical functions like ticket validation and payment processing remain uninterrupted during temporary connectivity issues.

4.3. Performance Requirements

4.3.1. **Transaction Processing:** The system must process parking entry, exit, and payment transactions with minimal time to ensure a smooth user experience.

4.3.2. **Real-Time Updates:** Parking space availability and fee calculations must be updated immediately when a vehicle enters or exits.

4.3.3. **Logging Efficiency:** All transactions, login attempts, and system events must be recorded and stored without delay for accurate tracking and auditing.

4.3.4. **User Interface Responsiveness:** The system's GUI must load and display critical parking and payment information with minimal delay of user interaction.

Use Case Specification Document

1. Use Case: Login and Logout

Use Case ID: UC-01
Relevant Requirements: Employee Login, Graphical User Interface
Primary Actor: Employee (Attendant, Manager)
Pre-conditions: The system must be powered on; the user must have valid login credentials.
Post-conditions: The user is successfully logged in or out of the system.

Basic Flow:

1. The user navigates to the login page.
2. The user enters their username and password.
3. The system verifies the credentials.
4. If valid, the system grants access based on the user's role.
5. The user is redirected to the dashboard.
6. When finished, the user selects the logout option, and the session ends.

Extensions/Alternate Flows:

- If credentials are incorrect, the system displays an error message.
- If three failed attempts occur, the account is locked for security.

Exceptions:

- System error preventing login.
- Unexpected session timeout during login.

Related Use Cases: Ticket Handling, Payment Processing

---

2. Use Case: Ticket Handling

Use Case ID: UC-02
Relevant Requirements: Ticketing System, Parking Space Tracking
Primary Actor: System, Customer
Pre-conditions: The parking garage has available space; the system is active.
Post-conditions: The customer receives a ticket, and a parking spot is reserved.

Basic Flow:

1. The customer arrives at the parking garage entry.
2. The system checks for available parking spaces.
3. If a spot is available, the system generates a ticket.
4. The system assigns a parking space and updates the total capacity.
5. The ticket is printed and given to the customer.

Extensions/Alternate Flows:

- If the garage is full, the system displays a **"Garage Full"** message and does not issue a ticket.

- If the ticket printer fails, an employee must manually generate a ticket.

Exceptions:

- System error preventing ticket issuance.

Related Use Cases: Handling Parking Data, Payment Processing

---

3. Use Case: Handling Parking Data

Use Case ID: UC-03
Relevant Requirements: Parking Space Tracking, Multi-Garage Management
Primary Actor: System, Employee
Pre-conditions: The system is running and actively tracking vehicles.
Post-conditions: The parking availability data is updated accurately.

Basic Flow:

1. A vehicle enters or exits the parking garage.
2. The system updates the number of occupied and available spaces.
3. The system assigns or frees up a parking space based on the transaction.
4. The system logs the event for tracking purposes.

Extensions/Alternate Flows:

- If a sensor fails to detect a vehicle, the system logs an error.
- Employees can manually adjust parking counts in case of a discrepancy.

Exceptions:

- System communication failure with parking sensors.

Related Use Cases: Ticket Handling, Payment Processing

---

4. Use Case: Payment Processing

Use Case ID: UC-04
Relevant Requirements: Payment Method, Parking Duration
Primary Actor: Customer, Employee, System
Pre-conditions: The customer has a valid ticket and is ready to leave.
Post-conditions: The payment is processed, and the exit gate opens.

Basic Flow:

1. The customer drives to the exit gate.
2. The system scans the ticket and calculates the fee based on duration.
3. The system prompts for payment.
4. The customer completes the payment using an accepted method (cash, card).
5. Upon successful payment, the system opens the exit gate.

Extensions/Alternate Flows:

- If the customer lost their ticket, an employee manually retrieves parking data and processes the payment.
- If the payment terminal fails, the customer must use another terminal or pay manually.

Exceptions:

- System error preventing payment processing.

Related Use Cases: Automatic vs. Manual Payment

---

5. Use Case: Automatic vs. Manual Payment Handling

Use Case ID: UC-05
Relevant Requirements: Employee Charging, Payment Processing
Primary Actor: Customer, Employee, System
Pre-conditions: The system has ticket data stored for each customer.
Post-conditions: The customer is charged correctly based on the method used.

Basic Flow (Automatic Payment):

1. A customer with an auto-payment account is detected at the exit.
2. The system verifies the linked payment method and calculates the fee.
3. The system processes the payment automatically.
4. If successful, the exit gate opens.

Basic Flow (Manual Payment):

1. A customer without auto-payment drives to an employee checkout booth.
2. The employee scans the ticket and retrieves the parking fee.
3. The customer makes the payment.
4. The employee confirms payment and allows exit.

Extensions/Alternate Flows:

- If automatic payment fails, the system prompts for manual payment.
- If the employee is unable to process a payment, the customer must use another payment method.

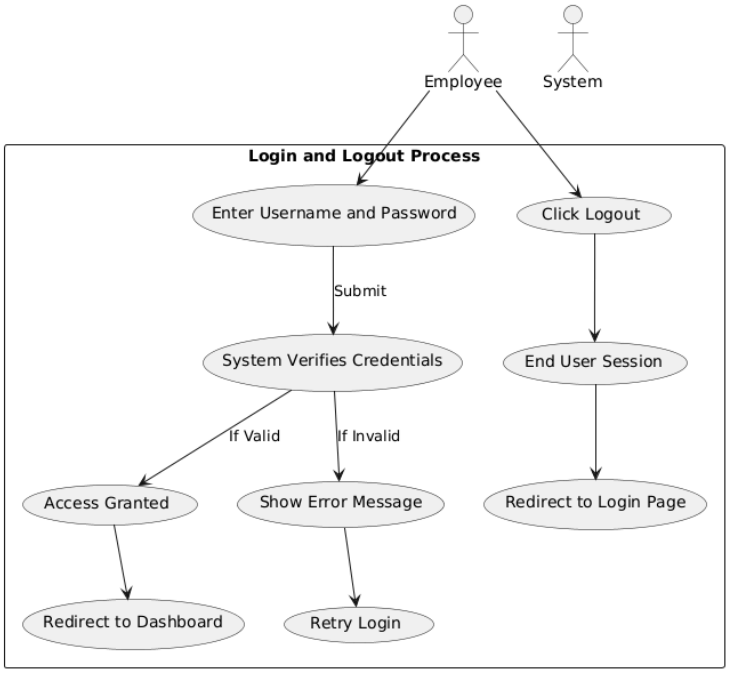
Exceptions:

- System failure preventing payment authorization.

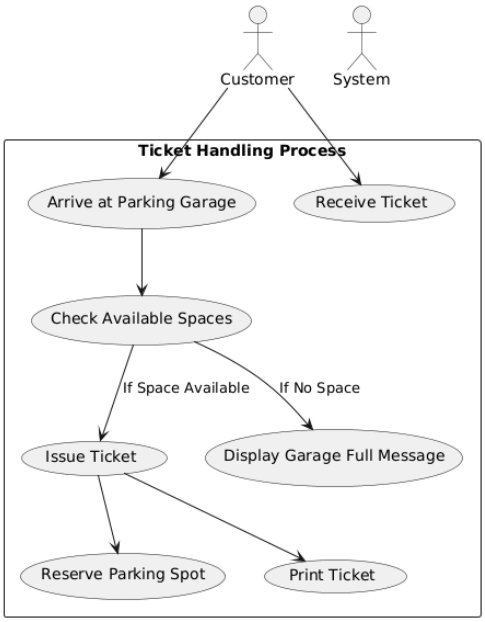
Related Use Cases: Payment Processing, Ticket Handling

UML Use Case Diagrams Document

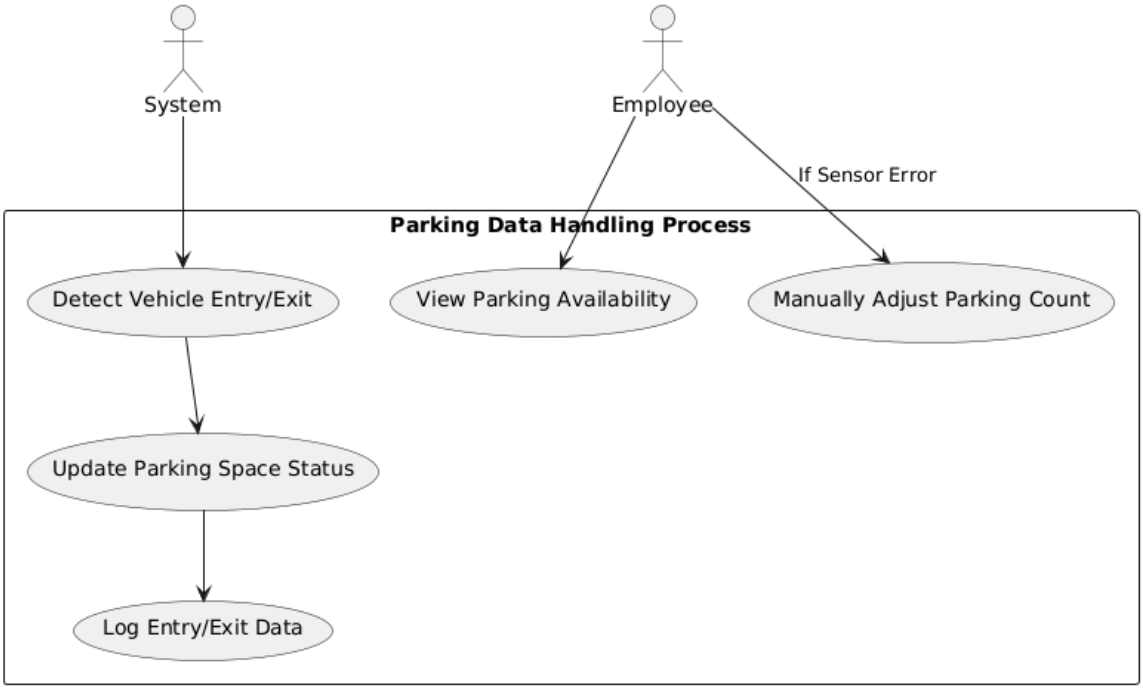
1. Use Case: Login and Logout



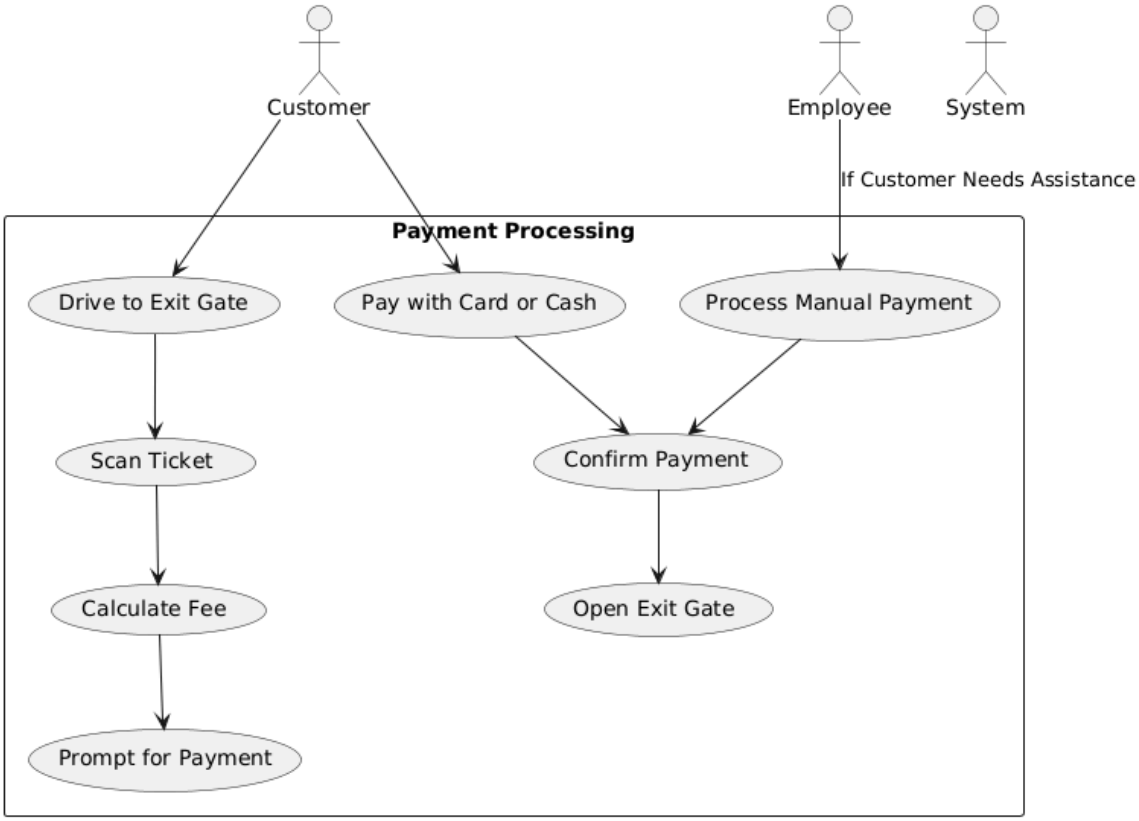
2. Use Case: Ticket Handling



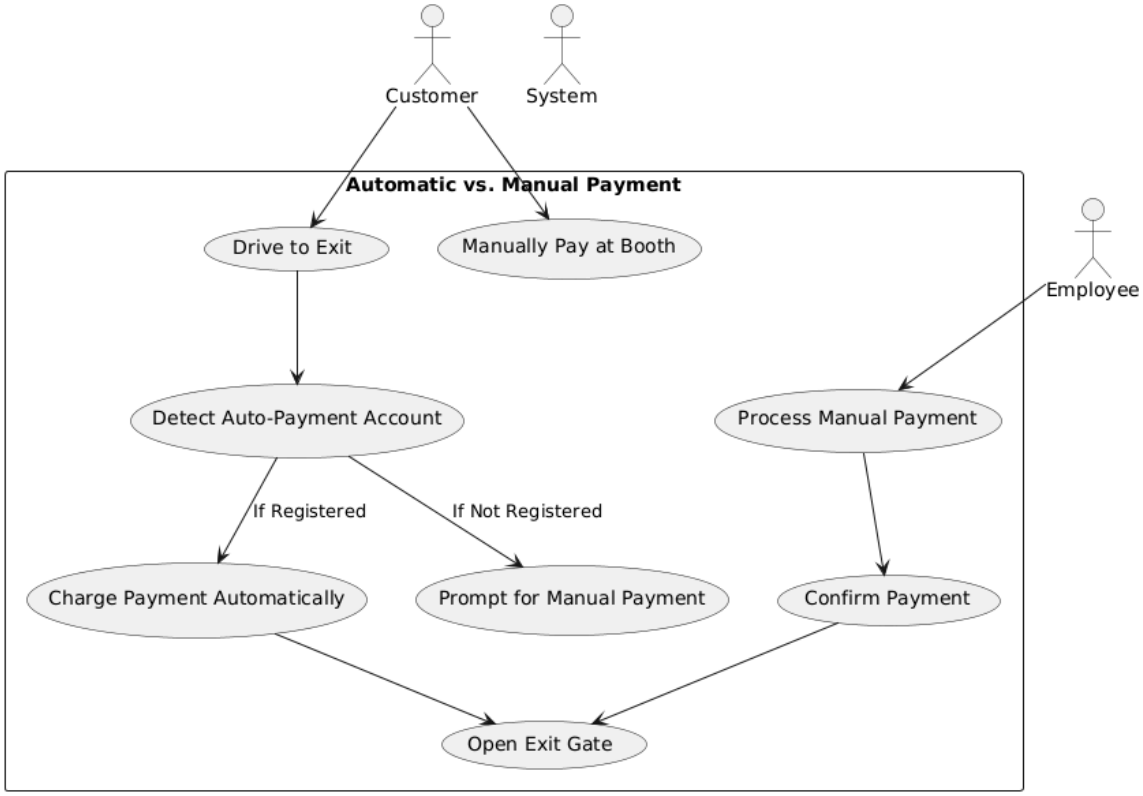
3. Use Case: Handling Parking Data



4. Use Case: Payment Processing



5. Use Case: Automatic vs. Manual Payment Handling



Class Diagrams

4. Brainstorming:

ParkingGarage

What do I know?

- Data encapsulated: garageID, location, totalSpaces, availableSpaces, spaces (List<ParkingSpace>)

What do I do?

- Methods:
  - isFull(): Returns whether the garage is at full capacity.
  - updateAvailability(): Updates the count of available spaces.

ParkingSpace

What do I know?

- Data encapsulated: spaceID, isOccupied

What do I do?

- Methods:
  - assignVehicle(Vehicle v): Marks the space as occupied by a vehicle.
  - removeVehicle(): Frees up the space.

Vehicle

What do I know?

- Data encapsulated: licensePlate, vehicleType

What do I do?

- No methods (Vehicle only stores data).

Ticket

What do I know?

- Data encapsulated: ticketID, entryTime, exitTime, fee

What do I do?

- Methods:
    - calculateFee(): Calculates the total parking fee based on duration.

Payment

What do I know?

- Data encapsulated: paymentID, amount, paymentMethod, isPaid

What do I do?

- Methods:
    - processPayment(): Marks the payment as completed.

Employee

What do I know?

- Data encapsulated: employeeID, name, role

What do I do?

- Methods:
    - processTicket(): Handles ticket verification and processing.
    - handlePayment(): Manages customer payments.

Customer

What do I know?

- Data encapsulated: name, contactInfo

What do I do?

- Methods:
    - enterGarage(): Allows the customer to enter the parking garage.

- exitGarage(): Allows the customer to exit after payment.

SystemLog

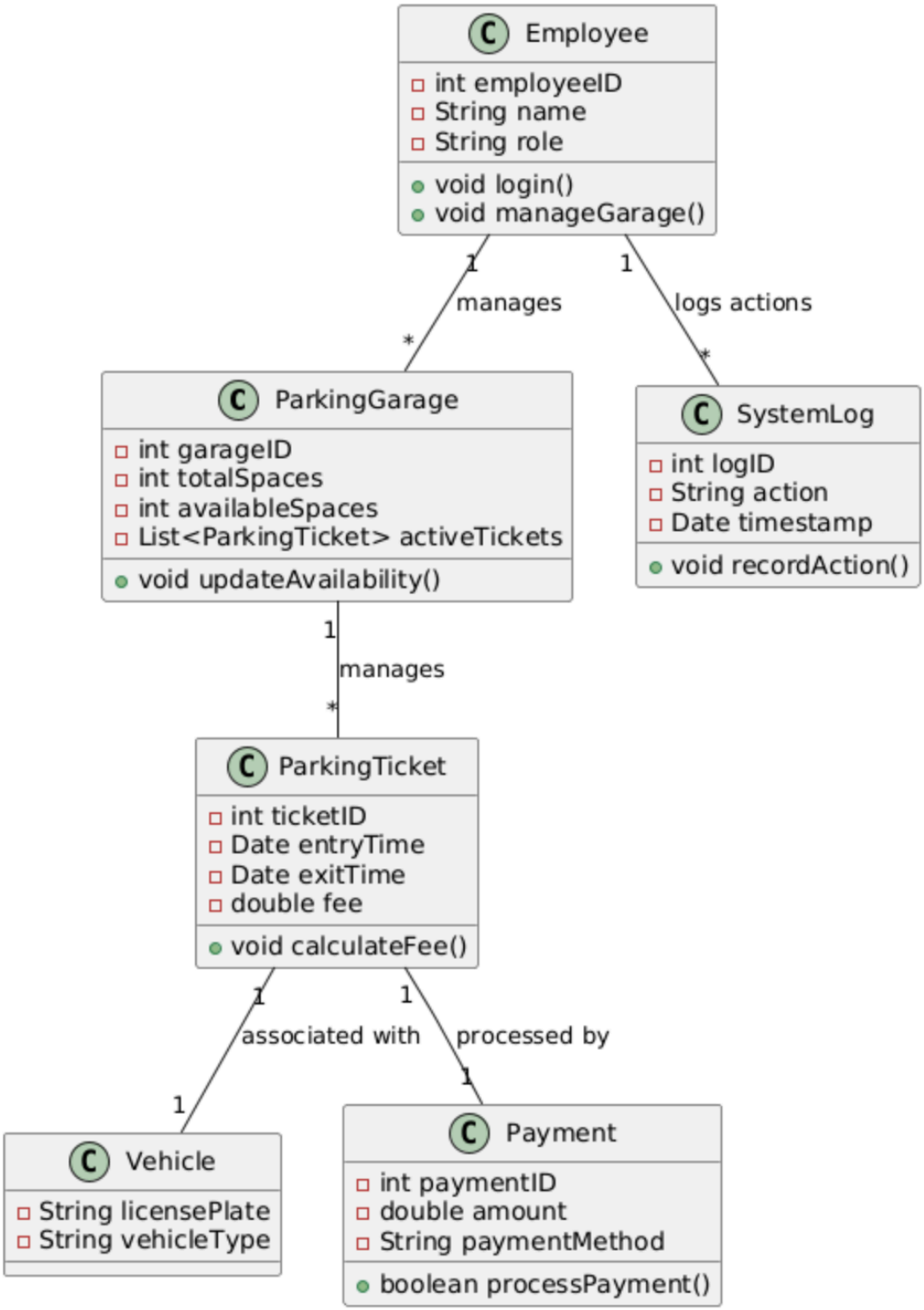
What do I know?

- Data encapsulated: logID, eventDetails, timestamp

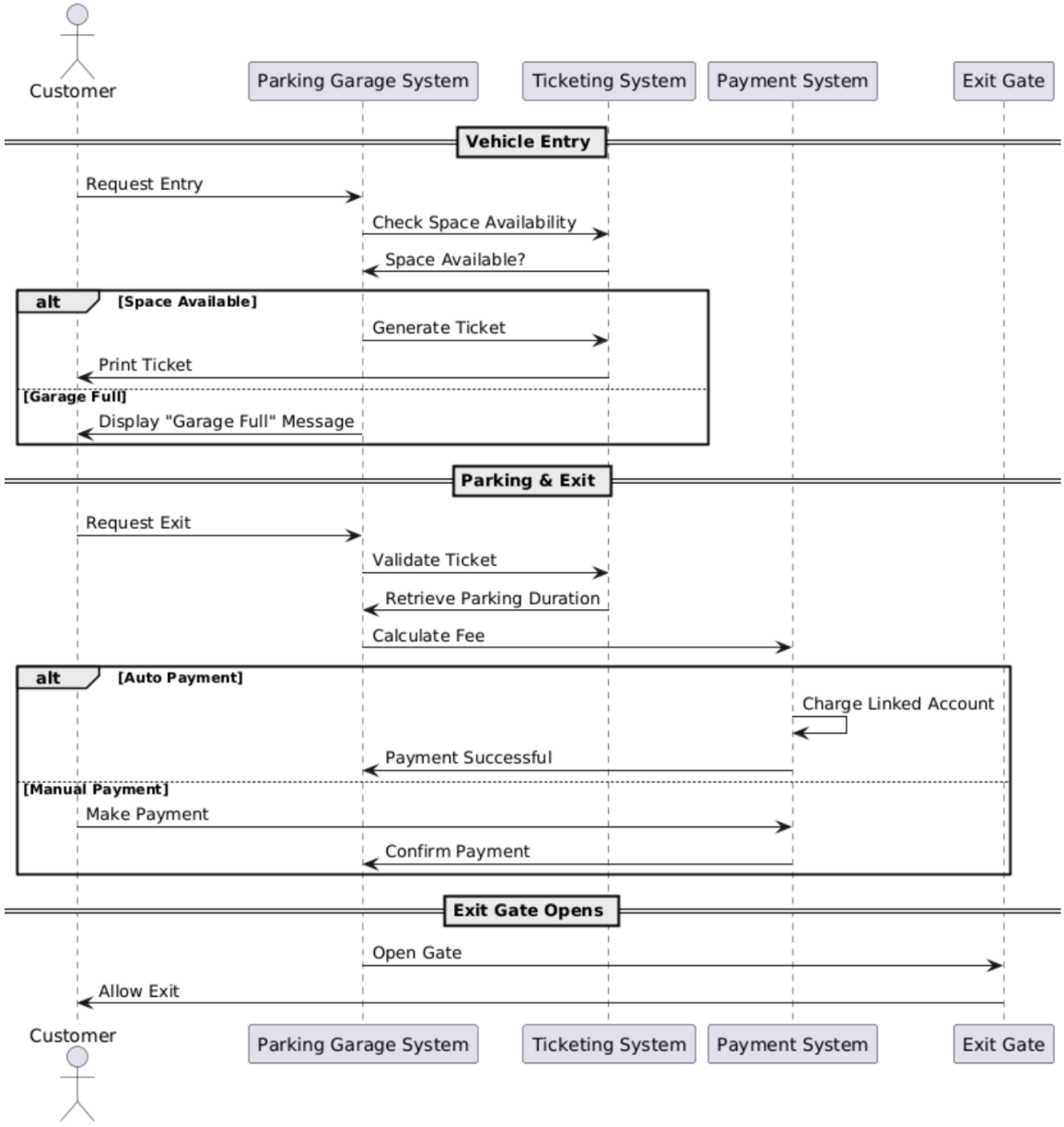
What do I do?

- Methods:
  - recordEvent(): Logs system activities such as vehicle entries, payments, and errors.

Class Diagram:



Sequence Diagrams



Sequence Diagram - Employee Side