Parking Garage Software Design Document
Group 1

Revision History

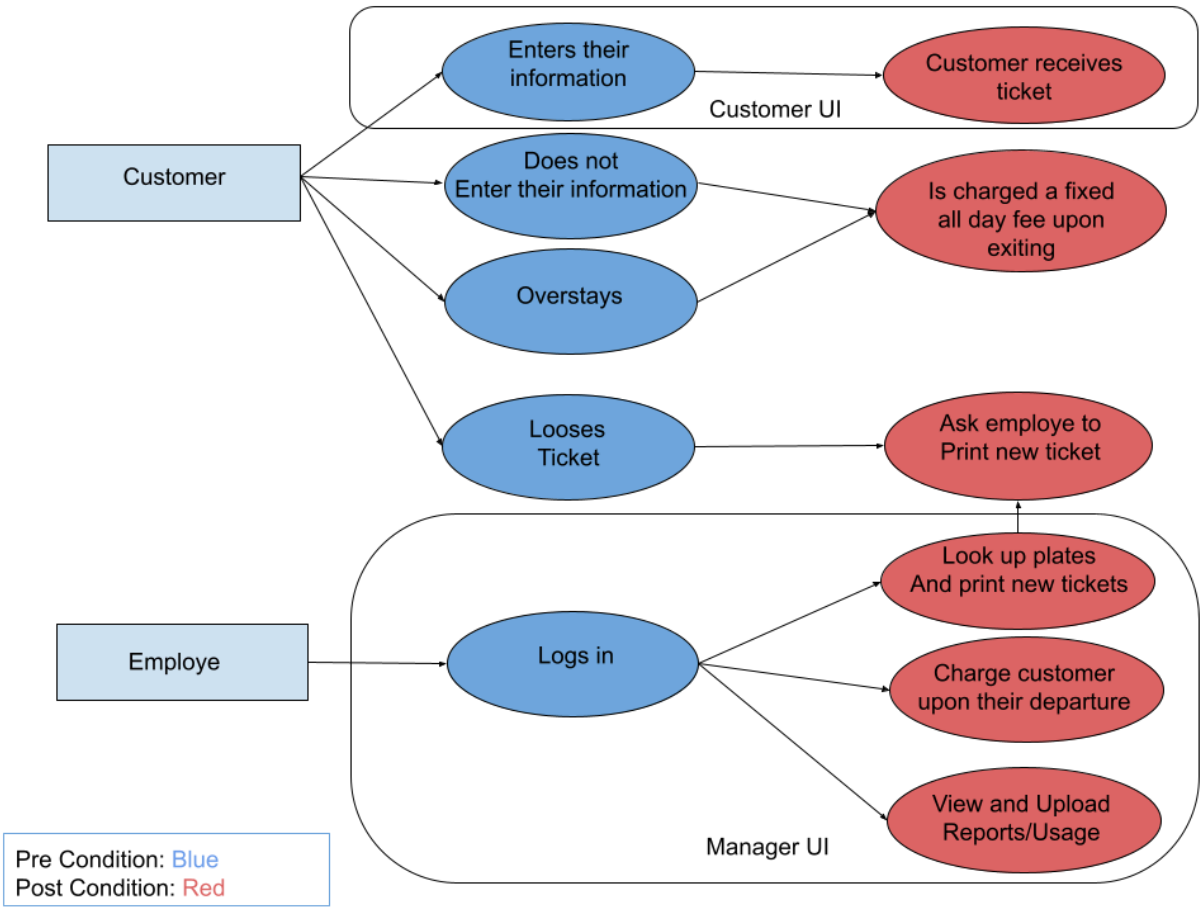
Date	Revision	Description	Author
10/22/2024	0.1	Template	Jeremy Renati
10/25/2024	1.0	Table Of Contents and organizing doc	Jeremy Renati
10/29/2024	1.1	Added Class Diagram	Adrian McGee
10/29/2024	1.2	Added use case for the GUI class	Adrian McGee
10/30/2024	1.3	Updated use case Diagram	Omar Rosas
10/30/24	1.4	Updated class diagram with new added classes +attributes/methods	Marc Bassal
10/30/24	1.5	added a new updated refresh class diagram	Adrian McGee
10/30/24	1.6	added definitions to the class design section	Omar Rosas/Adrian McGee

-

Table of Contents

Use Cases and Sequence

1.1. Use Case Diagram



1.2. Use Cases

Use Case ID: UC001
Use Case Name: Customer enters garage and pays (R0, R4, R5, R7, R8)
Primary Actor: Customer
Pre-conditions: none
Post-conditions: Customer successfully enters garage and pays

Basic Flow or Main Scenario:

1. Customer enters garage at the front gate and collects entry ticket
    a. Ticket has unique ID and timestamp
2. Customer parks and reaches the kiosk to pay
    a. Customer enters unique ID

    b. Customer proceeds to choose their choice of stay from our time limit menu
3. Customer enters their credit card and successfully pays
4. A completed ticket is printed with the Unique ID, Timestamp of successful transaction, transaction ID, and their choice of stay

Use Case ID: UC002
Use Case Name: Customer pays at exit gate (R5, R8, R10)
Primary Actor: Customer
Pre-conditions: Customer does not pay at the kiosk before leaving
Post-conditions: Customer successfully pays at the exit gate and leaves

Basic Flow or Main Scenario:

1. Customer chooses not to pay at the kiosk before leaving for various of reasons
2. Customer arrives at exit gate without a completed ticket
3. Customer is forced to pay the "All day" price since they did not pay kiosk
    a. This gives the system an easy method of collecting pay
4. Customer pays the fee and then successfully leaves the garage

Use Case ID: UC003
Use Case Name: Customer overstays (R4, R5, R9, R10)
Primary Actor: Customer
Pre-conditions: Customer overstayed their original choice
Post-conditions: Customer successfully pays fee and leaves the garage

Basic Flow or Main Scenario:

1. Customer overstayed their original choice of stay
    a. If customers overstay less than 2 hours over their choice, then they will be charged double their original price.
        i. If customer chose 2 hours to stay for $6 and actually stayed 3 hours then they will be charged $12
    b. If customer overstay more than 2 hours over their choice, then they will be charge the "All day fee"

        i. If customer chose 2 hours to stay for $6 and actually stayed more than 4 hours, then they be charged $25
2. Customer pays at kiosk to successfully resolve the over charge
    a. If a customer does not pay at the kiosk and instead at the gate, they are charged the all day fee regardless how long they stayed at the garage.
3. Customer Successfully leaves the garage

Use Case ID: UC004
Use Case Name: Management Login (R11)
Primary Actor: Management
Pre-conditions: None
Post-conditions: Successful login

Basic Flow or Main Scenario:

1. Management provides valid UserID and Password
2. System checks validate the entry
    a. If user enters valid entry then proceed to step 3
    b. If user enter invalid entry thenretrystep2
3. System Grants login for Management

Use Case ID: UC005
Use Case Name: Management checks reports (R11, R12, R13, R14)
Primary Actor: Management
Pre-conditions: Successful login
Post-conditions: Able to view data

Basic Flow or Main Scenario:

1. Management logs in with correct credentials
2. Management clicks drop menu to choose what report they want to see

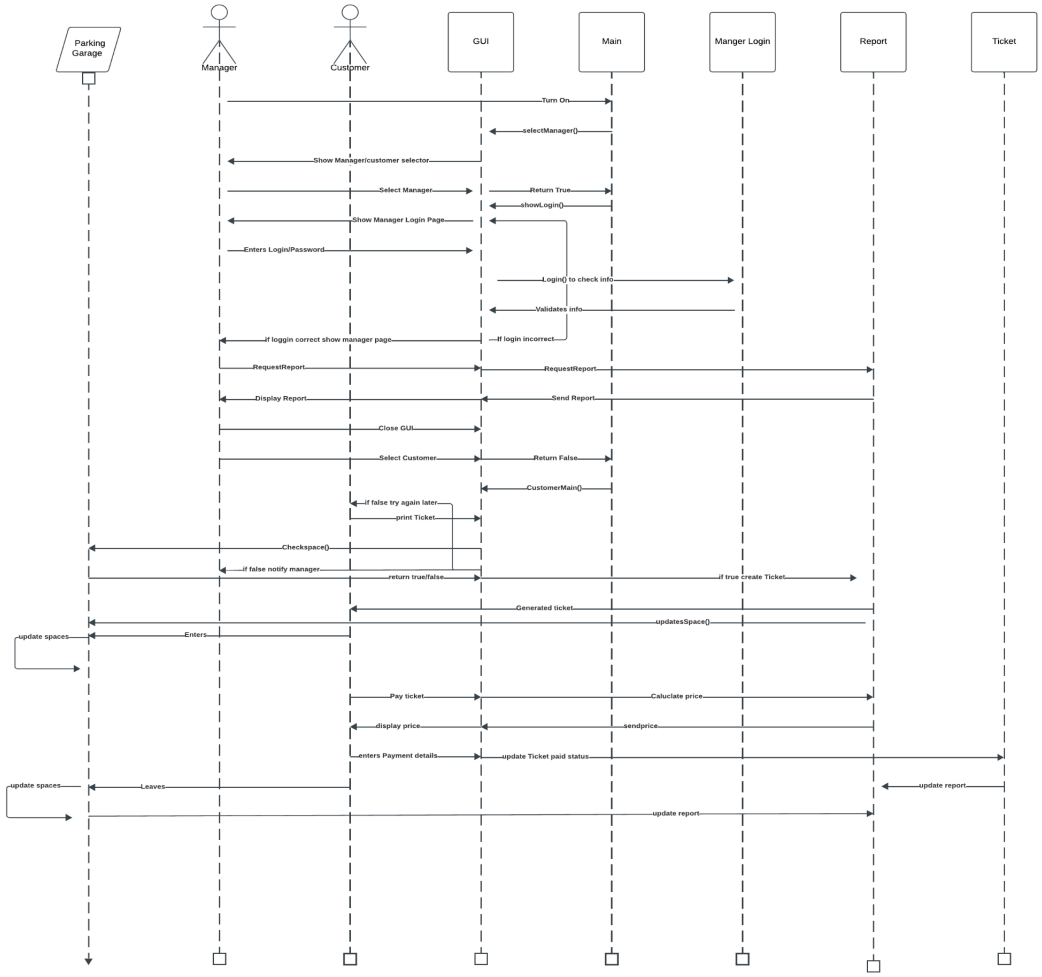
Use Case ID: UC006
Use Case Name: Management prints lost ticket (R4, R5, R11, R15)
Primary Actor: Management and customer
Pre-conditions: Successful login and lost ticket by customer
Post-conditions: Able to print new ticket

1. Basic Flow or Main Scenario
2. Customer loses ticket and talks to management company
3. Management logs in with credentials
4. Management prints new ticket for customer by using their last 4 digits of the card used to make the payment

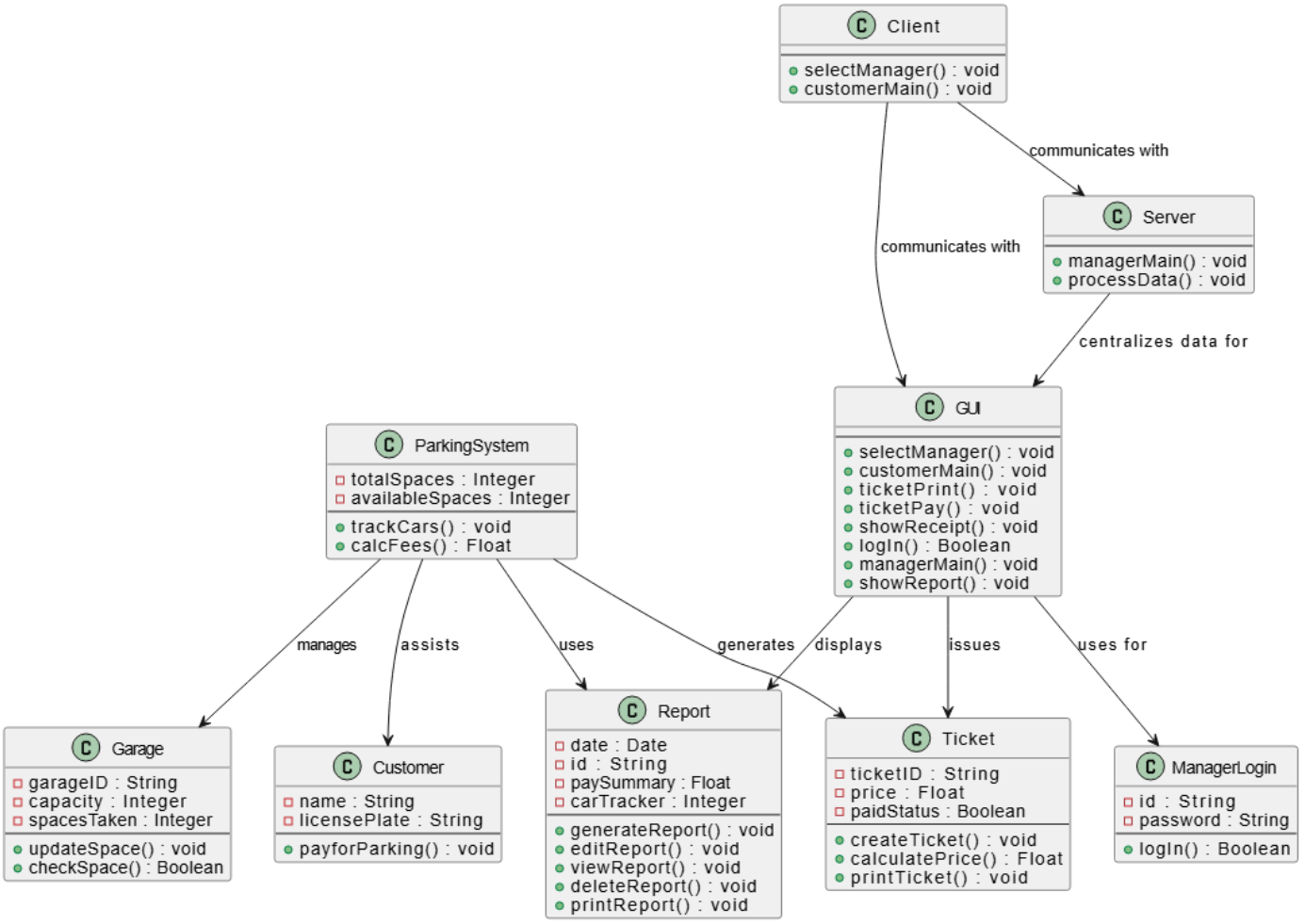
Use Case ID: UC007
Use Case Name: GUI interaction (R?)
Primary Actor: Management and customer
Pre-conditions: none
Post-conditions: Able to interact with GUI
Basic Flow or Main Scenario

1. Customer and manager interacts with GUI (terminal)
    a. Customer has the option to see drop down menu for pricing of the garage
    b. Manager has the option to login and view / print reports
2. Customer and manager complete their desired requests
    a. Customer is able to print their ticket and complete their purchase
    b. Manager is able to view / print reports

1.3. Sequence Diagram



1.4. Class Diagram



Class Design

2.1. GUI

2.1.1. The GUI will provide an interface that is simple and easy to use for the customer and manager to complete the desired actions of both parties.

2.1.2. methods

2.1.2.0. **selectManager( ):** Allows selection between manager and customer modes.

2.1.2.1. **customerMain( )**: Main interface flow for the customer

2.1.2.2. **ticketPrint( )**: provides a way to reprint a customer's ticket

2.1.2.3. **ticketPay( )**: initiates the payment process for the customer's ticket.

2.1.2.4. **showReceipt( )**: Displays a receipt to the customer after payment.

2.1.2.5. **logIn( )**: Authenticates user(like managers) through the GUI interface.

2.1.2.6. **managerMain( )**: Main interface for the manager.

2.1.2.7. **showReport( )**: Displays a generated report for the manager

2.2. Client/Server

2.2.1. The Client-Server class is crucial in a parking garage system because it facilitates seamless communication between the kiosk GUI (customer-facing) and the manager GUI, centralizing operations and ensuring data consistency.

2.2.2. Methods

2.2.2.0. selectManager( )

2.2.2.1. managerMain( )

2.2.2.2. customerMain( )

2.3. Manager Login

2.3.1. The manager-login class allows an employee to verify themselves to be able to use the employee UI.

2.3.2. **Attributes:**

2.3.2.0. **id:** A unique identifier for the manager, used for authentication.

2.3.2.1. **password**: a secret code associated with the id

2.3.3. **Methods:**

2.3.3.0. **logIn( )**: Allows the manager to enter by verifying the id and password, granting access if the credentials match those stored in the system.

2.4. Customer

2.4.1. The Customer class represents a user of the parking garage, linking individual customer details with their parking activities. Attributes like name for the customer's identification and licensePlate for vehicle tracking help in managing customer-specific records. This class helps streamline customer interactions within the system, ensuring efficient ticket assignment and payment handling.

2.4.2. **Attributes**

2.4.2.0. **licensePlate:** stores the customers plate

2.4.2.1. **Time in:** stores the time of entry

2.4.3. **Methods**

2.4.3.0. payforParking( )

2.5. parkingSystem

2.5.1. The ParkingSystem class serves as the central manager for the overall parking garage operations, keeping track of space availability and overseeing revenue calculations. With attributes like totalSpaces to represent the maximum parking capacity and availableSpaces to show real-time parking availability, this class provides a comprehensive view of the garage's occupancy status. This class ensures smooth operations, efficient space management, and accurate financial tracking

2.5.2. **Attributes**

2.5.2.0. The trackCars( )

2.5.2.1. calcFees( )

2.5.3. **Methods**

2.5.3.0. totalSpaces

2.5.3.1. availableSpaces

2.6. Reports

2.6.1. The Report class is essential in a parking garage ticketing system because it consolidates key data on daily operations and financial summaries, helping managers track and analyze parking usage, vehicle flow, and revenue generated within specific periods. This class allows managers to create, view, modify, and maintain records of garage performance.

2.6.2. Attributes

2.6.2.0. **generateReport( )**
2.6.2.1. **editReport( )**
2.6.2.2. **viewReport( )**
2.6.2.3. **deleteReport( )**
2.6.2.4. **printReport( )**

2.6.3. Methods

2.6.3.0. date
2.6.3.1. iD
2.6.3.2. paySummary
2.6.3.3. carTracker

2.7. Ticket

2.7.1. The Ticket class represents individual parking tickets in the system, managing ticket details and payment status. With attributes like ticketID for unique ticket identification, price is to indicate the final cost and paidStatus to indicate if the ticket has been settled, this class centralizes essential ticket information. This class supports smooth ticketing operations and accurate payment processing within the garage system.

2.7.2. Methods

2.7.2.0. **createTicket( )**
2.7.2.1. **calculatePrice( )**
2.7.2.2. **printTicket( )**

2.7.3. Attributes

2.7.3.0. ticketID
2.7.3.1. price
2.7.3.2. paidStatus

2.8. Garage

2.8.1. The Garage class represents the physical parking garage in the system, tracking and managing parking capacity and availability. With attributes like garageID for unique identification, capacity for total parking spaces, and spacesTaken for currently occupied spots, this class maintains up-to-date information on parking availability.

2.8.2. Methods

2.8.2.0. updateSpace( )
2.8.2.1. checkSpace( )

2.8.3. Attributes

2.8.3.0. garageID
2.8.3.1. capacity
2.8.3.2. spacesTaken