

# **Parking Garage Software**

## *Software Requirements Specification*

## Revision History

[illegible]

# Table of Contents

<b>Parking Garage Software.....</b>	<b>1</b>
Revision History.....	2
Table of Contents.....	3
<b>1. Purpose.....</b>	<b>4</b>
1.1. Scope.....	4
1.2. Definitions, Acronyms, Abbreviations.....	4
1.3. References.....	4
1.4. Overview.....	4
<b>2. Purpose.....</b>	<b>5</b>
2.1. Product Perspective.....	5
2.2. Product Architecture.....	5
2.3. Product Functionality/Features.....	5
2.4. Constraints.....	5
2.5. Assumptions and Dependencies.....	5
<b>3. Specific Requirements.....</b>	<b>6</b>
3.1. Functional Requirements.....	6
3.2. External Interface Requirements.....	6
3.3. Internal Interface Requirements.....	7
<b>4. Non-Functional Requirements.....</b>	<b>8</b>
4.1. Security and Privacy Requirements.....	8
4.2. Environmental Requirements.....	8
4.3. Performance Requirements.....	8
<b>Appendix 1: Use Case Specifications.....</b>	<b>9</b>
<b>Appendix 2: UML Use Case Diagrams.....</b>	<b>15</b>
<b>Appendix 3: Class Diagrams.....</b>	<b>16</b>
<b>Appendix 4: Sequence Diagrams.....</b>	<b>17</b>

# 1. Purpose

This document outlines the requirements for the Parking Garage Management System (PGMS).

## 1.1. Scope

The PGMS is designed to manage a network of parking garages, enabling customers to self-park for a fee, while employees handle the process of charging customers based on their parking duration. The system will track parking spots automatically, calculate fees, and generate usage reports. This Java-based application will have a GUI and will operate over TCP/IP without utilizing any external libraries or frameworks.

## 1.2. Definitions, Acronyms, Abbreviations

- **PGMS:** Parking Garage Management System
- **TCP/IP:** Transmission Control Protocol/Internet Protocol
- **GUI:** Graphical User Interface

## 1.3. References

- Appendix 1: Use Case Specification
- Appendix 2: UML Use Case Diagrams
- Appendix 3: Class Diagrams
- Appendix 4: Sequence Diagrams

## 1.4. Overview

The PGMS aims to provide an efficient and reliable solution for managing parking garage operations, allowing customers to self-park, while employees charge them based on their parking time. The system will handle the tracking of parking spots automatically.

## 2. Purpose

### 2.1. Product Perspective

The PGMS is a standalone software solution for parking garages, allowing real-time tracking of available spaces, fee calculation, and report generation. It is intended to work with a network of parking garages connected via TCP/IP.

### 2.2. Product Architecture

The system consists of two main modules:

- **Client Module:** A GUI-based application for customers and employees.
- **Server Module:** A central server that manages parking data, fee calculations, and report generation.

### 2.3. Product Functionality/Features

The high-level features of the system are as follows:

- Track available parking spaces.
- Calculate parking fees based on time spent.
- Employee interface to charge customers based on the system's calculations.
- Provide detailed reports on garage usage.

### 2.4. Constraints

- No external libraries, frameworks, or databases can be used without approval.
- The system must communicate using TCP/IP.
- No web or HTML components are permitted.

### 2.5. Assumptions and Dependencies

- Assumes a Java environment is available.
- The parking garages have the necessary network infrastructure.

### 3. Specific Requirements

#### 3.1. Functional Requirements

##### 3.1.1. Common Requirements

- 3.1.1.1. The system shall track the number of available parking spaces in real-time.
- 3.1.1.2. The system shall calculate parking fees based on the time the vehicle has been parked.
- 3.1.1.3. The system shall allow employees to charge customers using the calculated fees provided by the system.
- 3.1.1.4. The system shall generate daily, weekly, and monthly usage reports.

##### 3.1.2. Client Module Requirements

###### 3.1.2.1. Employee Interface (GUI)

- Charge customers based on their parking duration.
- View parking statuses, payment details, and transaction history.

###### 3.1.2.2. Customer Interface

- Customers can see the amount of spots available, see the fee, park and exit/request to pay.

##### 3.1.3. Server Module Requirements

- **Tracking Parking Spaces:** The server keeps track of all available, occupied, and reserved parking spots in real-time.
- **Managing Parking Durations:** It calculates entry and exit times for customers and determines the total fee.
- **Processing Payments:** When the client sends payment information, the server validates and records it.
- **Maintaining a Central Database:** The server acts as the central repository for all data, including employee actions, customer transactions, and usage reports.

#### 3.2. External Interface Requirements

The system must provide a user-friendly GUI for customers to interact with and for employees to charge customers.

### 3.3. Internal Interface Requirements

- The client module must communicate with the server module over a TCP/IP connection.
- Data exchange between the client and server should be structured and efficient.

## 4. Non-Functional Requirements

### 4.1. Security and Privacy Requirements

- Only authorized employees should have access to administrative features.

### 4.2. Environmental Requirements

- The system should be able to operate in a networked environment with multiple garages.
- It should function correctly on systems running a Java environment.

### 4.3. Performance Requirements

- The system should process user commands (e.g., adding a car, generating a report) quickly.
- The system should support many users accessing the parking management system without performance degradation.
- The system should be able to process and store many new transactions per hour.
- The system should maintain uptime, allowing access and management at all times, except during maintenance.
- The system should be scalable to handle additional garages with minimal changes to the existing codebase.



## Appendix 1: Use Case Specifications

- The goals of the customers are to generate tickets, process payment, and remove a vehicle.
- The employee will manage the parking spots, authenticate users, generate reports, and process payments.
- The payment system will process payments.
- The Parking garage system will manage the parking spaces, calculate fees, remove a vehicle, authenticate users, and generate reports
- The primary actors include customers, employees, the parking system, and the parking garage system

**Use Case ID:** UC1

**Use Case Name:** Ticket generation

**Primary Actor:** Employee or Customer

**Pre-conditions:**

- The user must be authenticated.
- The spot for parking needs to be present.

**Post-conditions:**

- Ticket generation
- The system designates the spot as occupied.

**Basic Flow or Main Scenario:**

1. The clients enter the garage.
2. The system checks in real time the parking spot availability.
3. The client is assigned a parking spot, and a ticket is generated.
4. The user can view from the GUI the details of the ticket.

**Alternate Flows:**

- The client is notified if the garage is full in case no parking spots are present.

**Exceptions:**

- Invalid spot number

**Related Use Cases:** UC3

**Use Case ID:** UC2

**Use Case Name:** Parking spot assignment

**Primary Actor:** Employee, Customer

**Pre-conditions:**

- The client is in the parking garage.
- The client has logged in to the system.
- There is an unoccupied parking spot.

**Post-conditions:**

- The parking system's available spots count is updated accordingly
- A ticket is generated with a timestamp.

**Basic Flow or Main Scenario:**

1. The clients enter the garage.
2. The system checks in real time the parking spot availability.
3. A ticket is generated for the client.
4. The system updates the count of available spots
5. The customer views a printed ticket.
6. The user can view from the GUI the details of the ticket.

**Alternate Flows:**

- There are no spots available
- An error is recorded during ticket generation, prompting the process to restart.

**Exceptions:**

- System failure

**Related Use Cases:** UC3 and UC5

**Use Case ID:** UC3

**Use Case Name:** Payment processing

**Primary Actor:** Employee, Customer

**Pre-conditions:**

- The client wants to remove the car.
- The ticket has all the relevant details, and the parking fee is computed.

**Post-conditions:**

- The fee is paid.
- The transaction is logged.
- The spot is marked unoccupied.
- The revenue is updated.

**Basic Flow or Main Scenario:**

1. The clients pay for the tickets using their preferred mode.
2. The fee is computed based on the duration of the vehicle in the parking lot.
3. The payment is processed using the method used to pay for the fee.
4. The status of the parking spot is updated
5. The client receives a payment receipt.

**Alternate Flows:**

- Lack of funds

**Exceptions:**

- System failure

**Related Use Cases:** UC2

**Use Case ID:** UC4

**Use Case Name:** Generating reports

**Primary Actor:** System Admin

**Pre-conditions:**

- Admin logs to the system
- Monthly data is available.

**Post-conditions:**

- A comprehensive report is provided.

**Basic Flow or Main Scenario:**

1. The administrator logs into the system.
2. They generate reports, either a report on the availability of parking spaces or a revenue report.
3. The data is retrieved in the form of a text file.
4. The report is generated.

**Alternate Flows:**

- No data is selected

**Exceptions:**

- Corruption of data

**Related Use Cases:** UC2 and UC5

**Use Case ID:** UC5

**Use Case Name:** Removing car from the spot

**Primary Actor:** Employee, Customer

**Pre-conditions:**

- There is a valid ticket.

**Post-conditions:**

- The assigned parking spot is no longer occupied.
- The client has exited the garage.

**Basic Flow or Main Scenario:**

1. The client offers a parking ticket to the machine or staff
2. The details of the ticket are retrieved by the system
3. The fees are computed, and the client pays
4. The spot is automatically updated to unoccupied.

**Alternate Flows:**

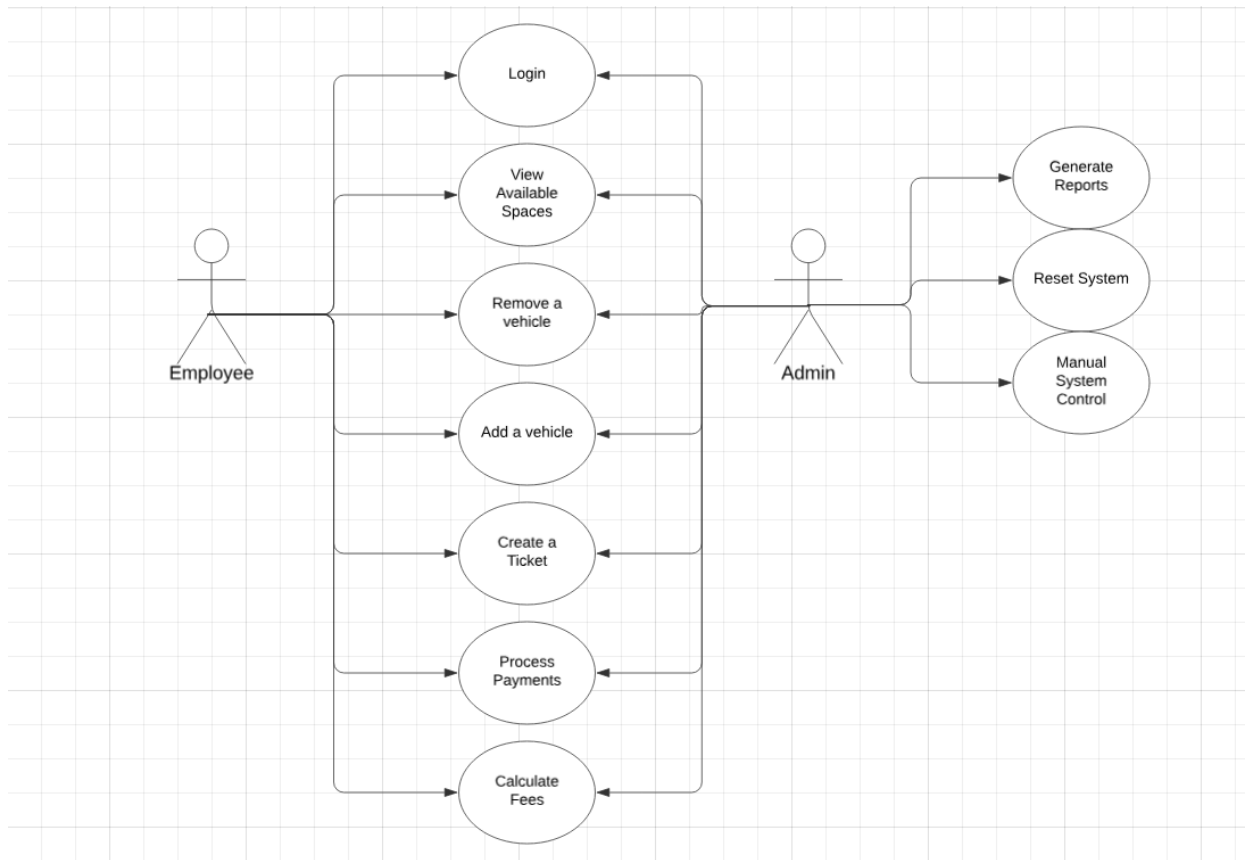
- Lost tickets requiring manual input

**Exceptions:**

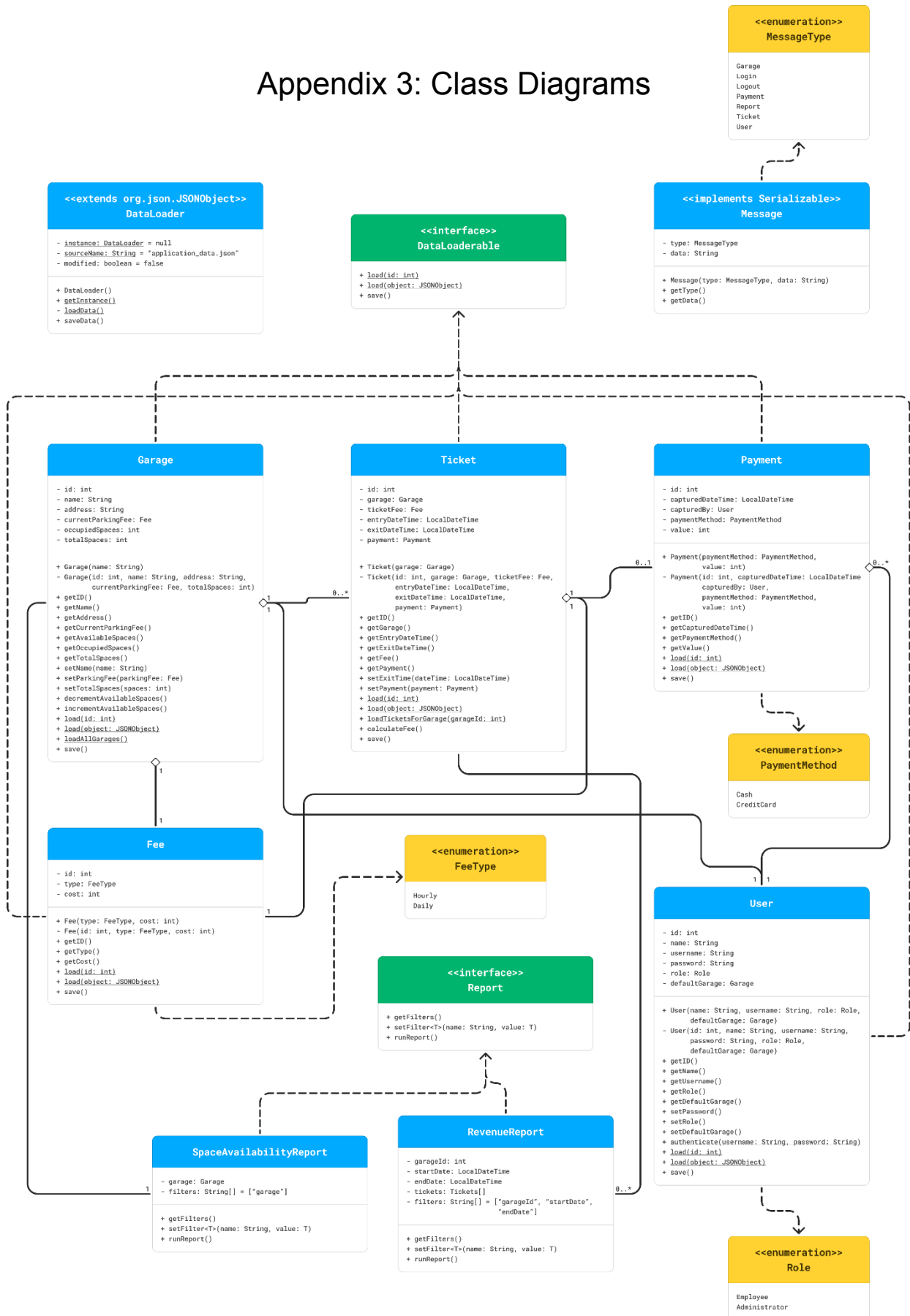
- N/A

**Related Use Cases:** UC2 and UC4

## Appendix 2: UML Use Case Diagrams



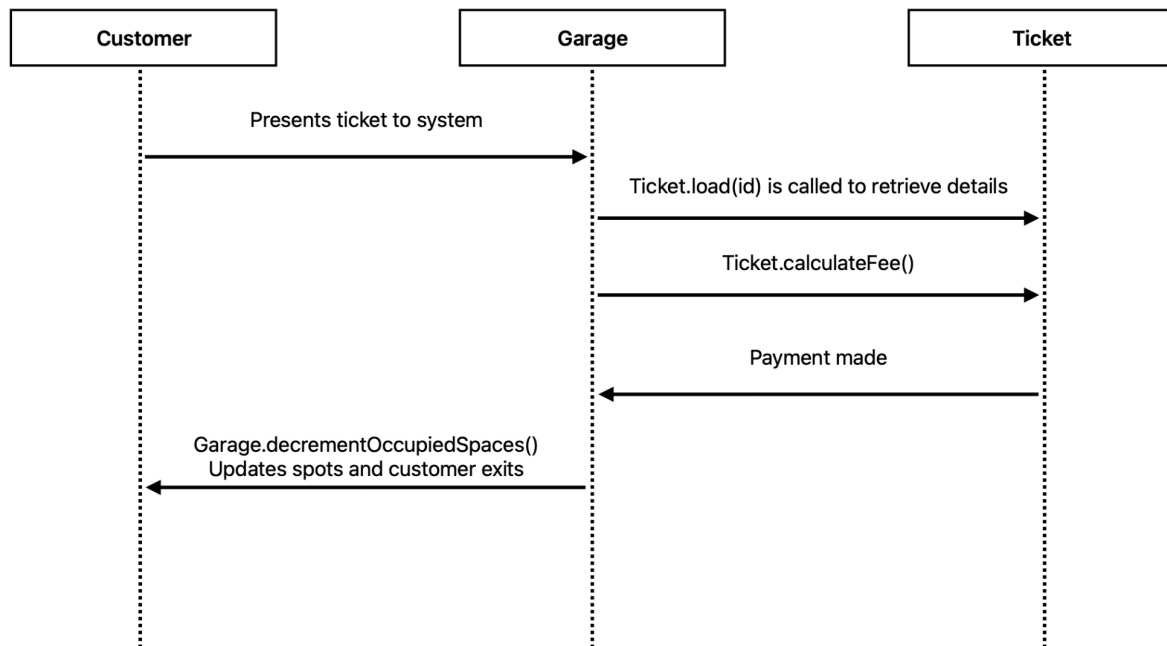
## Appendix 3: Class Diagrams



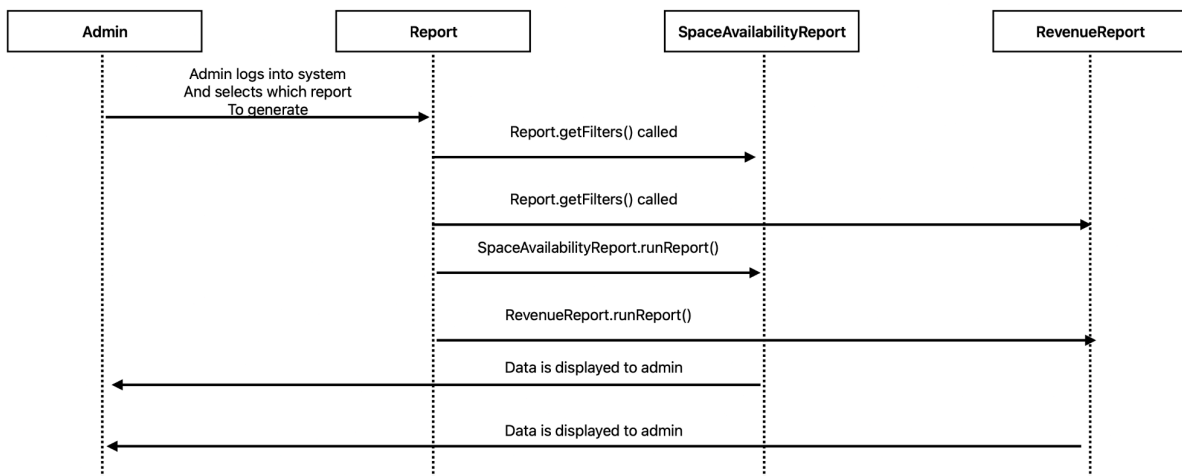


## Appendix 4: Sequence Diagrams

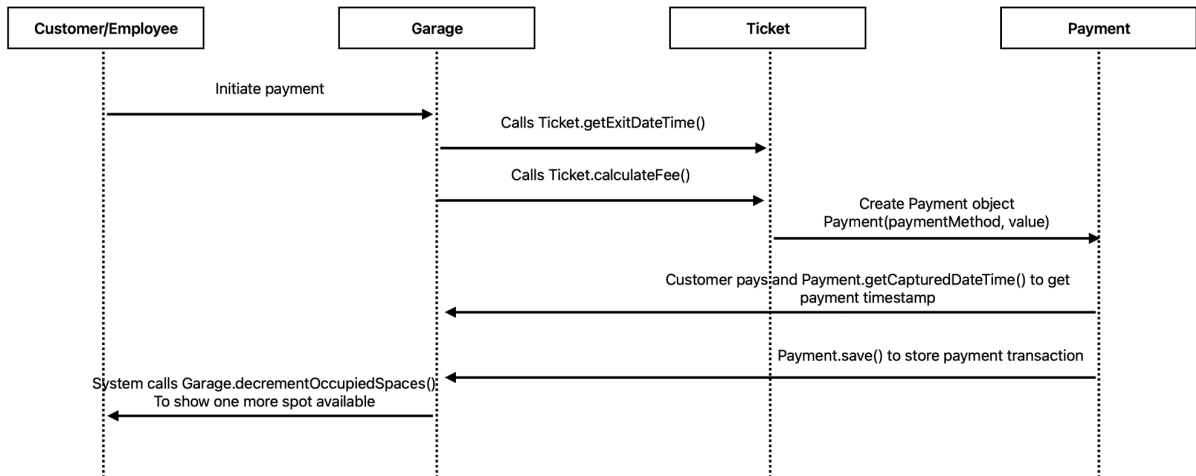
### Car Removal



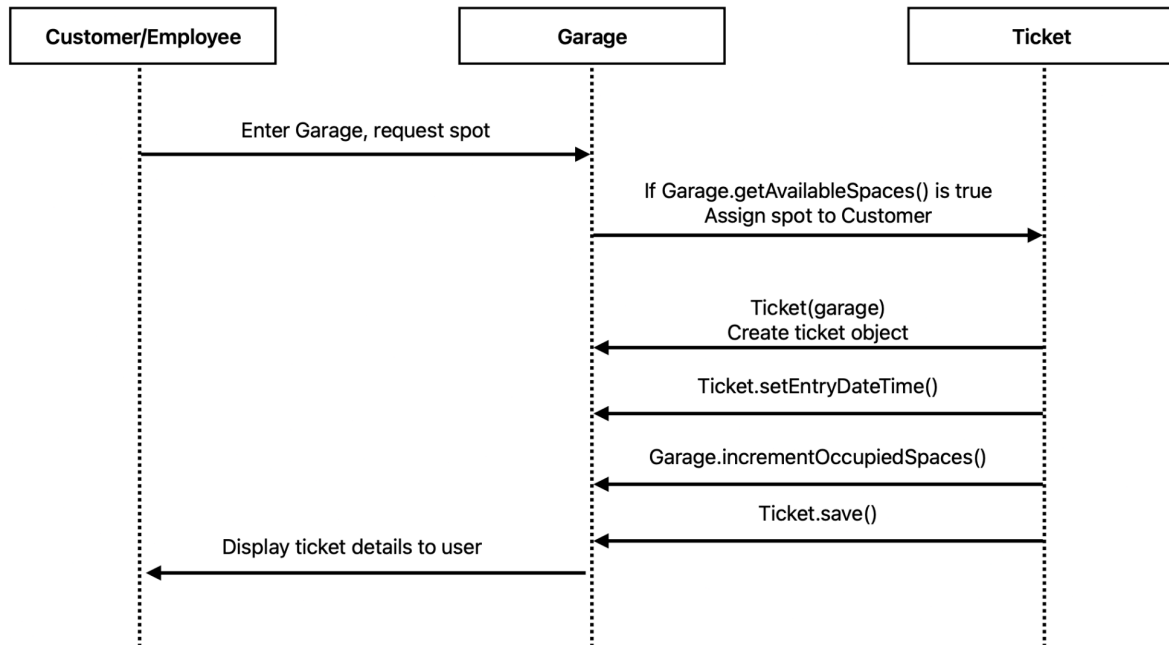
### Report Generation



## Payment Processing



## Parking Spot Assignment



## Ticket Generation

