Software Requirements Specification
Group 1

Revision History

Date	Revision	Description	Author
09/18/2024	1.0	Template	Jeremy Renati
09/25/2024	1.1	Copy over reqs. from Phase 1 doc (By Adrian)	Jeremy Renati
09/26/2024	1.1	Finished copying remaining info from Phase 1 doc/ some cleanup	Jeremy Renati
09/28/2024	1.2	Added definitions to following sections: Scope, Definitions, Overview, Internal and external interface requirements	Adrian McGee
09/30/2024	1.3	Added Sequence Diagram	Timofay Udalkin
10/1/2024	1.3	Added Use Case Diagram	Omar Rosas
10/2/2024	1.4	Made final changes to sections to 2.1 and 2.5	Adrian McGee
10/29/2024	2.0	Added a new use case for GUI	Adrian McGee
11/30/2024	3.0	Updating our requirements (classes) to reflect our changes to the project	Adrian McGee
12/2/2024	4.0	Changed Use cases to reflect the final code	Timofay Udalkin

Table of Contents

Purpose

This document outlines the requirements for the Parking Garage Ticketing System (PGTS).

1.1. Scope

This document will catalog the customer, software system, and hardware requirements for the PGTS system. It will not, however, document how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

- Parking Garage Ticketing System (PGTS
- Graphical User Interface (GUI)
- Refresh Rate
    - The frequency with which the image on a computer monitor or similar electronic display screen is refreshed, usually expressed in hertz.
- Usage Reports
    - A report that displays critical information such as peak times, customer data and gross profit.
- Over Limit Fee
    - A fee that is implemented to encourage customers to adhere to the garage policy and pay the necessary amount for their stay

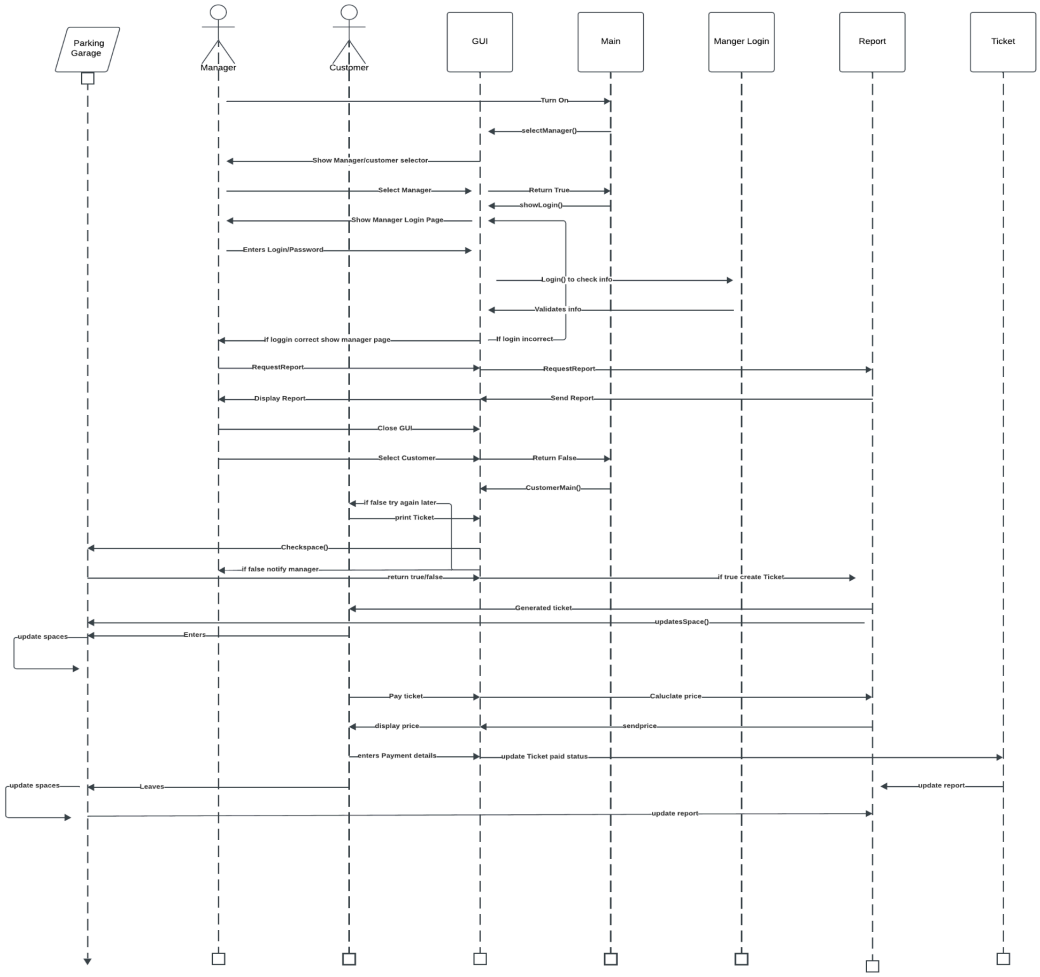
1.3. References

Use Case Specification Document – Step 2 in assignment description

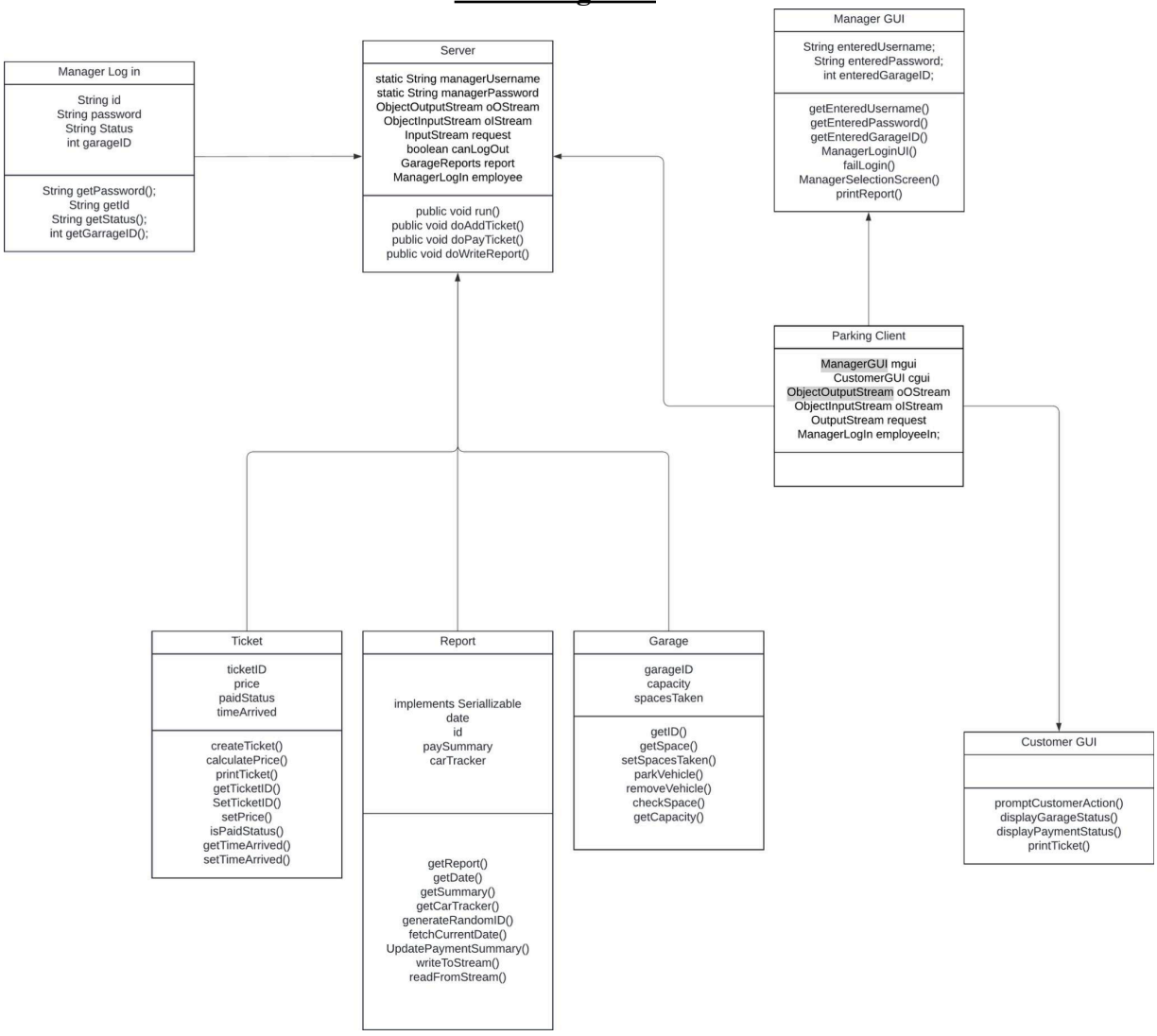
UML Use Case Diagrams Document – Step 3 in assignment description

Class Diagrams – Step 5 in assignment description

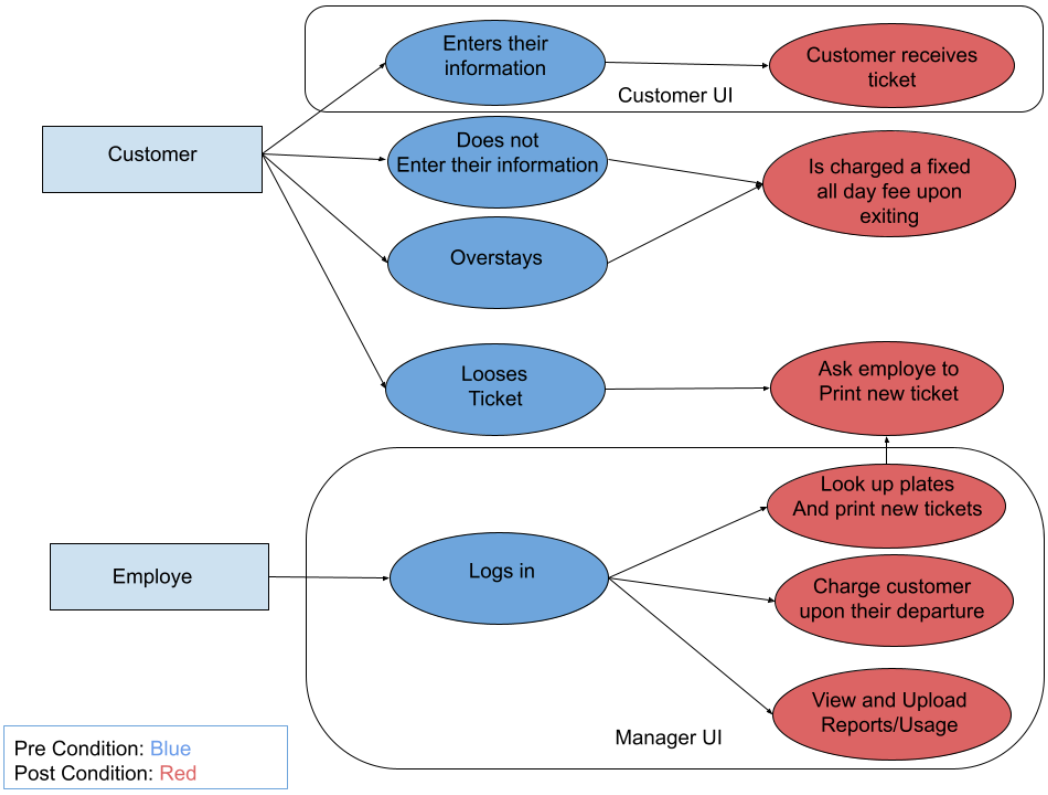
Sequence Diagram:



Class Diagram:



Use Case Diagram:



Use Cases:

Use Case ID: UC001
Use Case Name: Parking Server Program is launched by the manager
Primary Actor: Manager or Authorized Individual
Pre-conditions:

- The program is installed on a machine with an IP that can be connected to.

Post-conditions:

- The server is running successfully.

Basic Flow or Main Scenario:

1. An authorized individual starts the server.
2. The program is launched as a Java project.
3. No arguments are passed.

Use Case ID: UC002
Use Case Name: Parking Client Program(s) launched by the manager (R0)
Primary Actor: Manager or Authorized Individual
Pre-conditions:

- The server is running.

Post-conditions:

- The client is running, connected to the server, and the manager login page is open.

Basic Flow or Main Scenario:

1. The hostname is entered as an argument.
2. The manager launches the program as a Java project.

Use Case ID: UC003
Use Case Name: Manager Prints Report (R0, R1)
Primary Actor: Manager
Pre-conditions:

- The manager is logged in.

Post-conditions:

- A report is printed onto a text file.

Basic Flow or Main Scenario:

1. The manager presses "Print Report."
2. The client sends a request for the report to the server.
3. The server generates and sends an up-to-date report.
4. The client prints the report to a text file and displays the file's location.
5. A window opens, displaying that the report has been printed.
6. The manager presses "OK" to return to the home screen.

Use Case ID: UC004
Use Case Name: Manager Starts Customer GUI
Primary Actor: Manager
Pre-conditions:

- The manager is logged in.

Post-conditions:

- The Customer GUI is started successfully.

Basic Flow or Main Scenario:

1. The manager selects the option to start the Customer GUI from the Manager Selection Screen.
2. The Customer GUI is launched, allowing customers to interact with the system.

Use Case ID: UC005
Use Case Name: Customer Prints Ticket
Primary Actor: Customer
Pre-conditions:

- The Customer GUI is running.
- The server is running.

Post-conditions:

- A ticket with a unique ID, timestamp, and unpaid status is generated.

- The ticket is sent to the client and displayed or printed for the customer.

Basic Flow or Main Scenario:

1. The customer initiates the "Print Ticket" action from the Customer GUI.
2. The Customer GUI sends a request to the server to check garage availability.
3. If the garage is not full, the server generates a ticket with:
    - Unique ID
    - Timestamp
    - Unpaid status
4. The ticket is sent to the client and printed or displayed to the customer.
5. If the garage is full, the Customer GUI informs the customer of the unavailability.

Use Case ID: UC006
Use Case Name: Customer Pays for Ticket
Primary Actor: Customer
Pre-conditions:

- The Customer GUI is running.
- The server is running.
- The customer has an unpaid ticket.

Post-conditions:

- The ticket status is updated to "Paid" on the server.
- The payment amount is logged for reporting.

Basic Flow or Main Scenario:

1. The customer selects the "Pay Ticket" option in the Customer GUI.
2. The customer inserts ticket and payment details (e.g., card information).
3. The Customer GUI sends the payment details and ticket ID to the server.
4. The server validates the ticket ID and processes the payment.
5. If the payment is successful:
    - The ticket status is updated to "Paid."
    - The amount is logged in the server for reporting.
    - A confirmation is displayed to the customer.

6. If the payment fails, an error message is displayed, and the customer is prompted to retry.

Use Case ID: UC007
Use Case Name: Manager Closes Customer GUI and Returns to Manager Selection Screen
Primary Actor: Manager
Pre-conditions:

- The Customer GUI is currently running.

Post-conditions:

- The Customer GUI is closed.
- The Manager Selection Screen is displayed.

Basic Flow or Main Scenario:

1. The manager initiates the "Close Customer GUI" action from the Customer GUI interface.
2. The Customer GUI performs the following:
    - Saves any unsaved state or data.
    - Confirms that there are no ongoing actions (e.g., active transactions).
3. The Customer GUI is terminated.
4. The system automatically redirects the manager back to the Manager Selection Screen.

1.4. Overview

The Parking garage ticketing system (PGTS), is designed to print tickets and collect payment from customers. As operations increase throughout the day, so does the risk of traffic jams and loss of income. This system will assist in the garage's overall effectiveness ensuring a smooth operation for the parking garage management team and customer satisfaction

Overall Description

2.1. Product Perspective

The **Parking Garage Ticketing System (PGTS)** is a comprehensive solution that streamlines the process of managing parking tickets and payments, enabling parking garage operators to optimize their daily operations and enhance customer experience. The PGTS stands out in the market by offering a fully integrated solution that not only automates ticketing and payments but also actively helps prevent revenue loss and congestion. Its adaptability and ease of use make it an ideal choice for parking facilities ranging from small garages to multi-level complexes.

2.2. Product Architecture

The software will be created by utilizing the java coding language. By using object oriented programming with java, we are creating multiple classes that contain methods which will provide the software functionality based on our requirements. To keep a well organized system we will follow 3 modules: Parking Management Module, Fee Calculation Module and Garage Management Module.

2.3. Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

2.4. Constraints

2.4.1 Car and Parking Space Accuracy

- The software should be able to detect how many cars are entering and leaving the parking garage with precision. The margin of error should be within ±1%.

2.4.2 Daily Reporting

- The system should be keeping track of how many cars are entering and leaving each day.
- The amount of money should be calculated at the end of the day.

2.4.3 Performance

- The system should be able to handle multiple users at a time (if we decide to do two lanes coming and out).
- The response time for the user interface (getting tickets, selecting stay times, inserting tickets) must not exceed 10 seconds under normal conditions.

2.4.4 Reliability

- The system must operate 24/7 without downtime and have a failure tolerance of 99.99%.

2.5. Assumptions and Dependencies

2.5.1 Uninterrupted power supply

- The system assumes an uninterrupted power supply for continuous operation. In case of a power outage, the system should display a message that the customer needs to visit the garage office to purchase and receive a ticket in person.

2.5.2 Timely Maintenance

- Regular maintenance and updates for the software. Any lack of maintenance could affect the accuracy and reliability of the system.

2.5.3 User Proficiency:

- The parking garage staff and management team have a basic level of technological proficiency, enabling them to easily navigate and utilize the system's functionalities with minimal training

Specific Requirements

3.1. Functional Requirements

3.1.1 Common Requirements:

3.1.1.1 Functionality

- The software must perform the required tasks as expected and provide the desired functionalities to meet the needs of its customers.
- The system should offer features and capabilities that allow customers to complete their transactions with minimal effort and complexity.

3.1.1.2 Performance

- The software must operate with acceptable performance levels, such as speed, responsiveness, and resource usage.
- The system should handle tasks efficiently without excessive delays, and it should perform well under typical and peak loads.

3.1.1.3 Usability

- The software must be easy to use and intuitive for the end users.
- The interface should be simple, with clear navigation, well-labeled buttons, and instructions where needed.

3.1.1.4 Training

- The parking garage attendant should be trained on how to handle situations where the system either malfunctions or is powered off.
- Needs to know to reset the system with very little complexity
- Needs to know how to use the system such as printing reports and tickets

3.1.2 Parking Management Module Requirements:

3.1.2.1 Checks Space Availability

- Parking availability in real-time as cars enter or exit. (This only works in a parking garage setting)
- Tracks the number of cars parked and available spaces in the parking garage and each floor.

3.1.2.2 Capacity Alert

- An automated alert for the management team to know when capacity is rising.

3.1.2.3 Check-Out

- Includes a timestamp recording and receipt of the transaction.

3.1.2.4 Assigned IDs

- Assign customers when they purchase a ticket an unique ID and transaction number to ensure accurate record keeping.

3.1.2.5 Data Recovery

- Every 10 minutes the software should record the transaction ID and unique Ticket ID onto a text file for the management team just in case of hardware / software malfunction.

3.1.3 Fee Calculation Module Requirements:

3.1.3.1 Calculate Price

- Calculates the price based on how long the customer has stayed at the garage

3.1.3.2 Collect Payment

- Calculates the price based on the users choice of parking selection and collects the payment.

3.1.4 Garage Management Module Requirements:

3.1.4.1 Management Login

- This role requires a login with a username and password.
- This user has the power to access daily summaries and end of day reports.

3.1.4.2 On Demand Summary

- Generates various usage and financial reports. Reports the garage's activity at any time when demanded by the parking garage owner.

3.2.External Interface Requirements

3.2.1 User Interface - customer check out:

- The system should display a menu screen where customers can select their choice of stay and purchase their ticket

3.2.2 Management Interface:

- The system should display a section on the screen that provides a login screen for the management team

3.2.3 Report Interface:

- The management can view daily, weekly, monthly, and on the demand reports of the garage statistics.

3.2.4 Print ticket interface:

- This interface provides a screen where management can print a ticket for the customer on demand.

3.2.5 Graphical User Interface (GUI):

- This interface provides a screen to the customer and manager to provide them access to the necessary function to complete their desired actions
    - Provides the graphical interface for manager operations. Allows managers to log in, view garage reports, and interact with server-side operations through an intuitive GUI
    - Provides the graphical interface for customer operations. Enables customers to check garage status, print parking tickets, and make payments, interacting seamlessly with the ParkingClient.

3.3. Client and Server Requirements

3.3.1 ParkingClient:

- Acts as the client-side application for both managers and customers. It facilitates manager login, customer actions like parking and payment, and integrates with the Ticket and GarageReports classes to manage operations.

3.3.2 ParkingServer:

- Acts as the server-side application managing all client requests. This class processes manager and customer requests, validates logins, and manages garage operations. It serves as the central hub for handling tickets, payments, and report generation.

Non-Functional Requirements

4.1. Security and Data Integrity Requirements

4.1.1 Data Backup:

- In case of a situation where the software crashes and erases all existing data, there should be an external file that handles data storage to ensure the flow business is not interrupted.

4.1.2 Security:

- The software needs to recognize that the customer made a purchase and then it can proceed to the next step of printing a ticket.
- If the customer hasn't paid, the software will not raise the gates for the customer to leave (print message saying so).

4.2. Environmental Requirements

4.2.1 Dust and particles:

- All hardware components, including sensors and vending machine, must be protected against dust and small particles that are common in an outdoor setting.

4.2.2 Power Stability:

- The hardware of the system should operate on a stable and reliable power source to ensure that machines are running at full capacity at all times.

4.3. Performance Requirements

4.3.1 GUI Responsiveness:

- The system's GUI must respond to user inputs (e.g., button clicks, form submissions) within 500 milliseconds to provide a smooth user experience.

4.3.2 Fee Calculation:

- Parking fee calculations must be completed in less than 200 milliseconds for each customer when requested.

4.3.3 Usage Reports:

- Generating usage reports (e.g., daily, weekly, or monthly reports) should not take more than 5 seconds, even for large datasets.

4.3.4 Parking Availability and Car Count:

- must be updated across all clients in real time (within 1 second) when cars enter or leave the garage.

4.3.5 Refresh Rate:

- Our software should be refreshing at minute intervals to display the accurate amount of available spaces to customers to avoid traffic congestion and prevent overparking.

4.3.6 Peak Handling:

- The software should be able to handle multiple customers at a time and rapid transaction rates.