

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Description | Author |
| 09/25/2025 | 1.0 | Initial Version | Jonathan Frias |
| 09/25/2025 | 1.1 | Added Purpose | Phat Khuu |
| 09/25/2025 | 1.2 | Added References & Overview | Phat Khuu |
| 09/28/2025 | 1.3 | Added Non-Functional Requirements | Diana Aguirre |
| 09/28/2025 | 1.4 | Updated Functional Requirements, Added to Overall Description | Diana Aguirre |
| 09/28/2025 | 1.5 | Added Product Perspective, and Constrains. | Phat Khuu |
| 9/30/2025 | 1.6 | Added to Use Case List | Lucas Decout |
| 10/1/2025 | 1.7 | Added UML Use Case Diagram | Diana Aguirre |
|  |  |  |  |
|  |  |  |  |

# 

Table Of Contents

[**1. Purpose 4**](#_bmma5csbve1w)

[1.1. Scope 4](#_ypvmotn0x00v)

[1.2. Definitions, Acronyms, Abbreviations 4](#_q4nyuwe3peh)

[1.3. References 4](#_39d4c1iw8jfa)

[1.4. Overview 4](#_y41s6g2h8omo)

[**2. OVERALL DESCRIPTION 5**](#_rq863zbl95w4)

[2.1. Product Perspective 5](#_95xtbgjl7tf5)

[2.2. Product Architecture 5](#_j1s9mw1yfsc8)

[2.3. Product Functionality/Features 5](#_l4d504ceut52)

[2.4. Constraints 5](#_u9f0obi7ijif)

[2.5. Assumptions and Dependencies 5](#_wfd28rofvmpq)

[**3. SPECIFIC REQUIREMENTS 6**](#_lu4it37p7z09)

[3.1. Functional Requirements 6](#_ch0dxbhq6t8j)

[3.2. External Interface Requirements 6](#_npdymbs9ixx)

[3.3. Internal Interface Requirements 6](#_sljetx7tvu8f)

[**4. NON-FUNCTIONAL REQUIREMENTS 7**](#_t9nt6ux5jmab)

[4.1. Security and Privacy Requirements 7](#_s7rutw7ooapu)

[4.2. Environmental Requirements 7](#_rpchbcyf57sv)

[4.3. Performance Requirements 7](#_9k2m0pv2wg2m)

# 

# 1. Purpose

This document outlines the requirements for the Parkly Parking Garage System.

## 1.1. Scope

Note: Parking client-server setup.

The server takes care of the business payment logic, while the client provides an easy to use the interface for both customers and employees.

* Records (track) vehicle entry/exit events.
* Track available parking spaces.
* Generate unique tickets IDs.
* Calculate the fees based on the time usage.
* Provide reporting tools for garage usage (e.g., cars parked, revenue, peak hours).
* Ensure secure employee access and operations.

## 1.2. Definitions, Acronyms, Abbreviations

* GUI - Graphical User Interface
* TCP/IP - Transmission Control Protocol / Internet Protocol
* Ticket ID - Unique identifier for each parked vehicle

## 1.3. References

* Previous team slide deck: [Group Presentation Slides](https://docs.google.com/presentation/d/18vX9JAQ20NaqaytWpViJ0Ml152WtGwaYfWvFAR1ihtg/edit?slide=id.g37982ea7843_0_10#slide=id.g37982ea7843_0_10)
* Use Case Specification - 
* UML Use Case Diagrams – 
* UML Class Diagrams – 
* Sequence Diagrams –

## 1.4. Overview

The Parkly Parking Garage System is designed to enable customers to self-park their vehicles, generate tickets, calculate fees, and allow employees to manage garage operations efficiently.

## 

# 2. OVERALL DESCRIPTION

## 2.1. Product Perspective

The Parkly system functions as an independent application which duplicates the operations of actual parking garage management systems. The system operates best in restricted areas like university campus parking lots and small commercial building parking facilities.

The system functions as a reduced version of enterprise parking management software which serves educational and demonstration needs. The system operates through a client-server architecture which allows the server to handle business operations and data management while the client delivers user-friendly interfaces for customers and staff members.

The system enables kiosk machine integration at garage entrances and exits for printing tickets and processing fees. The system allows academic use of simulated sensors and cameras through software implementation.

## 2.2. Product Architecture

The system will be organized into 8 major modules: the parking management module, the space tracking module, the ticket validation module, the time usage tracker module, the fee calculation module, the customer checkout module, the usage reporting module, and the client-server 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):

1. **Parking Management**
2. **Space Tracking**
3. **Ticket Validation**
4. **Time Usage Tracker**
5. **Fee Calculation**
6. **Customer Checkout Function**
7. **Usage Reporting**
8. **Client-Server Architecture**
9. **No Ticket Issued if Full**
10. **Employee Login Credentials**

## 2.4. Constraints

* + 1. The GUI must be built using Swing or JavaFX.
    2. Communication between client and server must occur over TCP/IP sockets only.
    3. The system does not permanently store data in a database; all data is managed in memory or in simple text files.
    4. System capacity is limited to a simulated number of parking spots (e.g., 500 maximum).
    5. The project must be developed and completed within the course timeline
    6. Reports generated by the system are limited to basic information (cars parked, revenue, peak hours) and do not include advanced analytics.

## 2.5. Assumptions and Dependencies

2.5.1 It is assumed that the maximum number of vehicles in the garage at one time will not exceed the total number of marked spaces (e.g., 500).

2.5.2 It is assumed that customers will always take their ticket when entering the garage and that they will present their ticket at a kiosk before exiting the garage.

2.5.3 The system depends on the local network for communication between kiosks and the central server.

2.5.4 The system depends on functional entry and exit gate hardware to physically allow cars to move in and out of the garage.

# **3. SPECIFIC REQUIREMENTS**

## **3.1. Functional Requirements**

**3.1.1 Common Requirements**

3.1.1.1 Employees must log in with a valid username and password before accessing administrative functions.

3.1.1.2 The system must log all entry, exit, and payment events with timestamps.

**3.1.2 Parking Management Module Requirements**

3.1.2.1 The system must allow customers to self-park their cars by issuing a ticket at entry.

3.1.2.2 The system must record both entry and exit events for each ticket.

3.1.2.3 The system must prevent issuing a new ticket when the garage is full.

**3.1.3 Space Tracking Module Requirements**

3.1.3.1 The system must track the total number of cars currently in the garage.

3.1.3.2 The system must update available spaces when a car enters or exits.

3.1.3.3 The system must provide the current number of available spaces to the display sign.

**3.1.4 Ticket Validation Module Requirements**

3.1.4.1 The system must generate a unique ticket ID for each customer at entry.

3.1.4.2 The system must reject duplicate or invalid ticket IDs at payment or exit.

**3.1.5 Time Usage Tracker Module Requirements**

3.1.5.1 The system must record the entry time when a ticket is issued.

3.1.5.2 The system must track total time parked down to the minute to make sure it is accurate for billing purposes.

**3.1.6 Fee Calculation Module Requirements**

3.1.6.1 The system must calculate parking fees based on the total parking time.

**3.1.7 Customer Checkout Module Requirements**

3.1.7.1 The kiosk must allow customers to pay with either cash or credit/debit card at exit.

3.1.7.2 The kiosk must retrieve the correct fee using the ticket ID before payment.

3.1.7.3 The kiosk must mark the ticket as “paid” after successful payment.

**3.1.8 Usage Reporting Module Requirements**

3.1.8.1 The system must generate daily reports showing the number of cars parked.

3.1.8.2 The system must generate daily revenue reports.

3.1.8.3 The system must generate reports showing peak usage times.

**3.1.9 Client-Server Module Requirements**

3.1.9.1 The system must include a server application to store data and manage operations.

3.1.9.2 The system must include a client application (GUI) for kiosks and employees.

3.1.9.3 The client and server must communicate over TCP/IP.

## 3.2. External Interface Requirements

3.2.1 The system must provide an interface to the entry gate hardware so that the gate opens when a valid ticket is issued.

3.2.2 The system must provide an interface to the exit gate hardware so that the gate opens only after the ticket has been marked as paid.

3.2.3 The system must provide an interface to the payment terminal to process cash and card transactions.

3.2.4 The system must provide an interface to the ticket printer to print unique ticket IDs at entry.

3.2.5 The system must provide an interface to the display sign showing available spaces.

## 3.3. Internal Interface Requirements

3.3.1 The system must send new ticket data from the Parking module to the Validation module.  
3.3.2 The Validation module must pass entry time data to the Fee Calculation module.  
3.3.3 The Fee Calculation module must send the computed fee to the Checkout module.  
3.3.4 The Checkout module must send payment requests to the Payment Terminal and return results.

3.3.5 The Space Tracking module must update the Display Sign when cars enter or exit.  
3.3.6 The Reporting module must collect logs from Tickets, Payments, Exits, and Overrides.  
3.3.7 The Audit Log must record all login, logout, override, and gate actions.

## 

## 

# **4. NON-FUNCTIONAL REQUIREMENTS**

## **4.1. Security and Privacy Requirements**

4.1.1 The system requires employees to log in with valid credentials.

4.1.2 The system must encrypt customer payment data being transmitted over the Internet.

4.1.3 The system must not store customers’ credit card information after the payment is processed.

## **4.2. Environmental Requirements**

4.2.1 The system cannot require that any software other than a web browser be installed on employee computers.

4.2.2 The system must be deployed on existing Linux-based server infrastructure.

## **4.3. Performance Requirements**

4.3.1 The system must process a customer payment and open the gate in less than 5 seconds.

4.3.2 The system must support at least 100 transactions per hour without failure.

4.3.3 The receipt printer must produce a receipt within 5 seconds of payment confirmation.