**Parking System Software SRS**

**Authors:**

Brian Nguyen

Gabriel Smith

Javier Jasso

Kirk Iliev

Mannat Mannat

Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 9/25/2024 | 1.0 | Initial Version |  |
| 9//28/2024 |  | Usage Reports | Mannat Mannat |
| 9/30/2024 | 1.1 | Added Interface Module | Javier Jasso |
| 9/30/2024 | 1.2 | Updated Capacity Module requirements | Kirk Iliev |
| 9/30/2024 | 1.3 | Updated Tracking Time Module requirements | Brian Nguyen |
| 9/30/2024 | 1.4 | Updated Payment System Module Requirements | Gabriel Smith |
| 10/1/2024 | 1.5 | Added gantt chart and overall desc. page | Javier Jasso |
| 10/2/2024 | 1.6 | Added Use Cases 1-6 & class diagrams | Javier Jasso |
| 10/2/2024 | 1.7 | Sequence Diagram | Mannat Mannat |
| 10/30/2024 | 1.8 | Minutes and Project Gantt Chart Update | Javier Jasso |
| 10/30/2024 | 1.9 | Added More Use Cases & Use Cases Diagram | Brian Nguyen |
| 10/28/2024 | 1.10 | Refined Class Diagrams and relationships | Kirk Iliev |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Overall Description**

**Product Architecture**

The system will be organized into x major modules:. .

**Product Functionality/Features**

The high-level features of the system are as follows

* Parking ticket generator
* Ticket payment system
* Parking Lot Capacity Display
* Data Reporting

**Constraints**

* Software must integrate with GitHub and Junit
* Java must be used as the source code
* No databases are to be used, only .doc files
* The project must be completed by 10/2/2024

**Assumptions and Dependencies**

* Data encryption will not be needed
* Employee login information will be used for security authentication and will be provided
* Parking lot capacity information will be provided by client

**References:**

* **GitHub Repository Link:**
  + <https://github.com/jj0789/CS-401-Group7>
* **Use Cases:** See page 10
* **Use Case Diagrams:** See page 13
* **Sequence Diagram:** See page 14
* **Class Diagram:** See page 15
* **Project Schedule Breakdown:** See page 16

**FUNCTIONALITY MODULES**

**Payment System** - Gabriel Smith

**Functional Requirements:**

* Allow three Payment Methods - Cash, Card, or Mobile
* Present a User Interface to allow the user to pay for parking and issue a ticket allowing them to leave
* In the case of Mobile and Card, will not issue ticket if the user has insufficient funds

**Non-Functional Requirements**

* Log payment history, recording amount paid and info of users

**Capacity Module Requirements** - Kirk Iliev

* Real-time capacity updates
  + System shall update the capacity count immediately upon vehicle entry or exit.
* Prevent overcapacity
  + System shall prevent entry of additional vehicles when the garage is at full capacity.
* Multiple entry/exit point management
  + System must be able accurately track capacity in real-time with consideration towards multiple entry or exit points in a single garage
* Network synchronization
  + System shall synchronize data across all the garages connected to the network
* Capacity alerts
  + System must be able to generate an alert when garage is nearing its capacity (for example 85-90%) full
* Reserved spaces tracking
  + System must take into account spaces that have been reserved in advance for the total available capacity.
* Interface with usage reports
  + System must be able to provide digestible data to the usage report module to generate usage reports.
* Changes to overall capacity
  + System must be able to accept modification/adjustment of total capacity in the case of adding/removing spaces to maintain adaptability

**Tracking Time Module** - Brian Nguyen

**Functional Requirements**:

* Track Vehicle Entry/Exit Times
  + The system must be able to automatically record the accurate time of when a vehicle enters or exits the parking garage.
* Parking Duration
  + The system must be able to calculate the vehicle’s total parking time by subtracting the time vehicle enters and the time vehicle exits.
* Parking Status
  + The system must show accurate data for vehicles that are currently parked, including the entire parking duration.
  + Vehicles parking duration going beyond their time limit will quickly notify the system.

**Non-Functional Requirements**:

* Performance
  + Any time operations, including vehicle entry/exits and parking duration calculations, must be calculated with minimal latency and delay.
* Reliability
  + System must be up and running all the time while the parking garage is active to prevent any data loss and missed tracking time records.
* Security
  + Tracking time data must be secure enough to prevent any data tempering.
* Usability
  + Interface must be fairly easy and straightforward to use, making it efficient to input and edit tracking time records.

**Usage Reports** - Mannat

## **Requirements for the Usage Reports Module**

### **Functional Requirements**

* **Usage Tracking:**
  + **Vehicle Entry and Exit:** The system should accurately record the time a vehicle enters and exits each parking garage.
  + **Vehicle Type:** The system should be able to identify and track different types of vehicles
  + (e.g., cars, motorcycles, trucks).
  + **Usage Duration:** The system should calculate the duration of each parking session.
* **Reporting:**
  + **Daily Reports:** The system should generate daily reports summarizing the total number of vehicles that entered and exited each garage, the total parking revenue, and the average parking duration.
  + **Hourly Reports:** Similar to daily reports, but with hourly granularity.
  + **Peak Usage Periods:** The system should identify peak usage periods (e.g., busiest hours of the day, busiest days of the week).
  + **Revenue Analysis:** The system should provide insights into revenue generation, including total revenue,average revenue per parking session, and revenue trends over time.
* **Data Visualization:(exceptional)**
  + **Charts and Graphs:** The system should present usage data in a visually appealing format, using charts and graphs (e.g., line charts, bar charts, pie charts).
  + **Customizable Reports:** Users should be able to customize reports to focus on specific data points or time periods.

### **Non-Functional Requirements**

* **Accuracy:** The system should accurately track vehicle entries, exits, and usage durations.
* **Performance:** The system should be able to handle high volumes of data and generate reports in a timely manner.
* **Security:** The system should protect sensitive data (e.g., vehicle information, usage data) from unauthorized access.
* **Scalability:** The system should be able to accommodate future growth in the number of parking garages and users.
* **Usability:** The system should be easy to use, with a user-friendly interface.
* **Integration:** The system should be able to integrate with other parking management systems (e.g., capacity l,payment systems).

### **Additional Considerations**

* **Data Retention:** The system should define a policy for data retention.
* **Alerts and Notifications:** The system should be able to send alerts or notifications based on specific criteria (e.g.,low parking availability, revenue thresholds).
* **Future Enhancements:** The system should be designed to accommodate future enhancements.

**Interface Requirements**

**External Interface Requirements** :

* The system must provide a user interface that will display the menu option to begin parking usage or to pay for existing parking
* Interface system will provide a payment option for customer
* Interface will have access to internal logic for payment calculations
* Interface will have access to storage system for data tracking

**Internal Interface Requirements**

* Interface will give access to parking system data for backend users
* Layout will have security authentication screen by asking for username and password of user
* Interface must have access to data storage system
* Interface will display the menu options to display current capacity data for each parking lot, display past usage data, display financial data and the option to output the data
* Data output will be in a .doc file format

**Use Cases   
x1**

**Use Case ID**: 1

**Use Case Name:** Start Parking

**Relevant Requirements**:

**Primary Actor:**  User

**Pre-conditions:** None

**Post-conditions:** Ticket created

**Basic** **Flow or Main Scenario:**

1. User uses interface to begin parking
2. Tracking begins

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:** 2

**Use Case ID**: 2

**Use Case Name:** Calculate Fee

**Relevant Requirements**:

**Primary Actor:** System

**Pre-conditions:** Tracking begins

**Post-conditions:** Display Fee

**Basic** **Flow or Main Scenario:**

1. User is given the fee amount
2. Ticket will be able to be created

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:**

**Use Case ID**: 3

**Use Case Name:** Pay for Parking

**Relevant Requirements**:

**Primary Actor:** user

**Pre-conditions:** Ticket created

**Post-conditions:** Ticket paid for

**Basic** **Flow or Main Scenario:**

1. User uses interface to enter current parking ticket
2. User pays for parking

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:** 1

**Use Case ID**: 4

**Use Case Name:** Get Capacity

**Relevant Requirements**:

**Primary Actor:** System

**Pre-conditions:** None

**Post-conditions:** Display capacity

**Basic** **Flow or Main Scenario:**

1. Capacity display needed
2. Display capacity

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:**

**Use Case ID**: 5

**Use Case Name:** Login to Server

**Relevant Requirements**:

**Primary Actor:** System

**Pre-conditions:**

**Post-conditions:**

**Basic** **Flow or Main Scenario:**

1. Connection to server is made

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:**

**Use Case ID**: 6

**Use Case Name:** Parking Info

**Relevant Requirements**:

**Primary Actor:** System

**Pre-conditions:** Login to Server has been established

**Post-conditions:**

**Basic** **Flow or Main Scenario:**

1. Parking info is being displayed

**Extensions or Alternate Flows:**

**Exceptions:** None

**Related Use Cases:**

**Use Case ID**: 7

**Use Case Name:** Usage Report

**Relevant Requirements**:

**Primary Actor:** Backend user

**Pre-conditions:** Date & Parking Lot ID

**Post-conditions:** Display usage data for given day

**Basic** **Flow or Main Scenario:**

1. User request data for a specific parking lot
2. System display data

**Extensions or Alternate Flows:**

**Exceptions:** Invalid Login

**Related Use Cases: 6**

**Use Case ID**: 8

**Use Case Name:** Financial Report

**Relevant Requirements**:

**Primary Actor:** Backend user

**Pre-conditions:** Date & Parking Lot ID

**Post-conditions:** Display financial data for given day

**Basic** **Flow or Main Scenario:**

1. User request data for a specific parking lot
2. System display data

**Extensions or Alternate Flows:**

**Exceptions:** Invalid Login

**Related Use Cases: 6**

**Use Case ID**: 9

**Use Case Name:** Print Report

**Relevant Requirements**:

**Primary Actor:** Backend user

**Pre-conditions:** Report selected

**Post-conditions:** Prints report to .doc file

**Basic** **Flow or Main Scenario:**

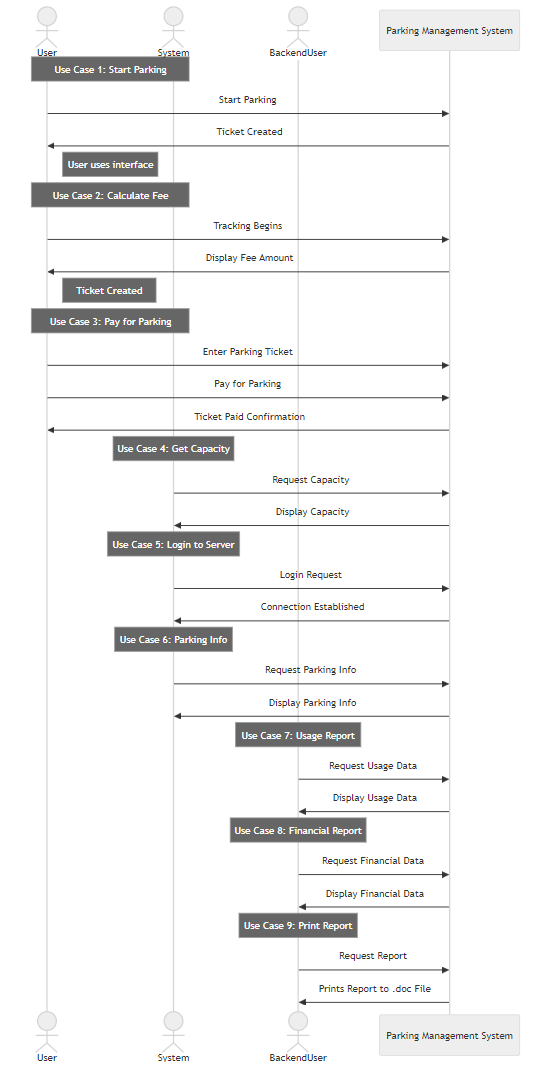
1. User request report
2. System prints report

**Extensions or Alternate Flows:**

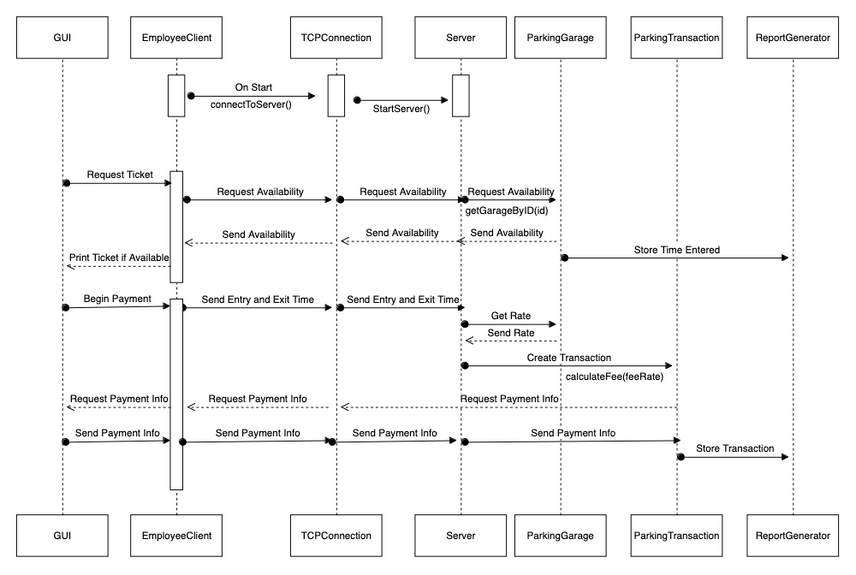
**Exceptions:** None

**Related Use Cases:**

**Use Cases Diagram:**

****

**Sequence Diagram**

****

**Class Diagrams**

