Report 2:

Smart Parking Garage Management System (SPGMS)

CSCI 441 VC Software Engineering

Fall 2023

Submitted October 16, 2023

Team B

Nicole Brandenburg

Gene Holt

Oakley Cardwell

https://github.com/nbrandenburg/CSCI_441_VC_Team_B

Team Member Contributions Breakdown

NAME	CONTRIBUTION	PERCENTAGE
Nicole Brandenburg, Team Leader	Proposal: ● Proposed Solution Report 1, Part 1: ● Functional Requirements ● Non-Functional Requirements ● User Interface Requirements ● Glossary of Terms Report 1, Part 2: ● Traceability Matrix ● Fully-Dressed Descriptions ● System Sequence Diagrams ● Preliminary Design ● User Effort Estimation Report 1, Part 3: ● Identifying Subsystems ● Architectural Styles ● Global Control Flow ● Project Size Estimation ● References Report 2, Part 1: ● Conceptual Model ● Concept Definitions ● Association Definitions ● Attribute Definitions ● Traceability Matrix Report 2, Part 2: ● Traceability Matrix Report 2, Part 3: ● User Interface Design and Implementation	33.3%
Gene Holt	Proposal: ● Problem Diagnosis Report 1, Part 1: ● Project Management ● Glossary of Terms Report 1, Part 2: ● Use Case Diagram ● Project Management Update Report 1, Part 3: ● Project Management Update ● References	33.3%

	Report 2, Part 1: ● System Operation Contracts ● Project Management Update Report 2, Part 2: ● Interaction Diagrams Report 2, Part 3: ● Design of Tests ● Plan of Work	
Oakley Cardwell	Proposal: ● Proposal Rough Draft ● Plan of Work Report 1, Part 1: ● Customer Problem Statement ● Decomposition into Sub-Problems ● Business Goals ● Glossary of Terms Report 1, Part 2: ● Stakeholders ● Actors and Goals ● Casual Descriptions ● Use Case Diagrams ● Traceability Matrix Report 1, Part 3: ● Identifying Subsystems ● Connectors and Network Protocols ● Hardware Requirements ● References Report 2, Part 1: ● Data Model and Persistent Data Storage Report 2, Part 2: ● Class Diagrams ● Data Types and Operation Signatures Report 2, Part 3: ● Design of Tests	33.3%

Table of Contents

1. Customer Problem Statement

a. Problem Statement

Many parking garages face challenges such as inefficient space utilization, difficulty in tracking available spots, and lack of real-time data for users. From a user perspective, not finding a spot in a timely manner delays planned activities, increases the chances for the user to park further away from their intended area, and not being able to easily locate their vehicle when they want to leave. If the lot/garage they park at requires payment, the user will need to take time away from their plans to ensure that payment is made when they need to extend their visit or risk being towed.

The more people searching for a parking space in any given parking lot/garage, the more difficult it is to enter/exit and navigate the parking lot/garage due to congestion. From an owners/operators perspective, if a spot is available and there is no method to advertise it's availability, it can cause potential loss in revenue if no user randomly discovers it's availability. These problems are amplified in densely populated areas where people are randomly searching for parking spaces.

Our team plans to create a program that introduces interventions such as a user-friendly website for drivers, and predictive analysis data for garage operators. Drivers will be able to check spot availability, reserve spots, and even pay for their parking in advance. Garage operators will have historical statistical data from which to make business decisions regarding pricing and promotions, and to minimize unused space.

b. Decomposition into Sub-Problems

To address the overarching challenges presented in the problem statement, it is essential to break down the proposed solution into specific sub-problems. By doing so, we can ensure that each facet of the problem is addressed comprehensively. Here are the identified sub-problems:

Parking Management

- Spot Addition/Removal: As the physical layout or the usage of the parking garage changes, there should be a mechanism to add or remove parking spots within the system. This ensures that the digital representation of the parking garage is always up-to-date.
- Spot Status Update: The system should be able to update the status of each parking spot in real-time, indicating whether it's occupied or available. This will help in providing accurate information to the users.

Admin Login

- Secure Access: Only authorized personnel should have access to the administrative features of the system. This includes adding/removing spots, viewing analytics, and managing user data.
- Multi-level Access: Different roles (e.g., manager, cashier, security) might require different levels of access. The system should cater to these varying needs.

Analytics

- Parking Trends: The system should be able to analyze and present trends such as peak parking times, average duration of parking, and frequent users.
- Busy Time Analysis: By understanding the busiest times for the garage, operators can make informed decisions about pricing and promotions.
- Space Utilization: Analytics should provide insights into how efficiently the space is being used, highlighting areas that are underutilized.

Payment Gateway

- Flexible Payment Options: Users should be able to pay using various methods, such as credit/debit cards, digital wallets, or even cash (if there's a physical kiosk).
- Payment Extensions: If a user needs to extend their parking time, they should be able to do so easily through the system, without having to physically return to their vehicle.
- Overstay Alerts: Users should be notified if their parking time is about to expire, giving them the option to extend if necessary.

Database Management

- Data Storage: All the data related to parking spots, user reservations, payments, and analytics should be stored securely.
- Data Retrieval: The system should be able to quickly retrieve data when required, ensuring a smooth user experience.
- Backup and Recovery: There should be mechanisms in place to back up the data regularly and recover it in case of any failures.

User Experience

- Spot Reservation: Users should be able to reserve spots in advance, ensuring they have a space when they arrive.
- Vehicle Locator: To address the challenge of users not being able to locate their vehicles, the system could provide a feature that helps users remember where they parked.
- Real-time Updates: The website or application should provide real-time updates on spot availability, ensuring users have the most accurate information.

By addressing each of these sub-problems, the proposed solution will provide a comprehensive answer to the challenges faced by parking garages and their users.

c. Glossary of Terms

The following glossary provides definitions for key terms and concepts related to the Smart Parking Garage Management System (SPGMS). Understanding these terms will help stakeholders and users better comprehend the functionalities and objectives of the system.

Admin Login: A secure access point in the system where authorized personnel can log in to access administrative features.

Analytics: Data-driven insights and trends derived from the system, helping garage operators make informed business decisions.

Database Management: The system's backend operations that handle the storage, retrieval, backup, and recovery of all data related to the SPGMS.

Digital Transactions: Payments made electronically, either through credit/debit cards, digital wallets, or other online payment methods integrated within the SPGMS.

Driver: A user of the system who is specifically a customer of the garage with a need for a space to park a vehicle for a certain amount of time.

Garage Administrator: A user of the system who is specifically an owner, employee, or person otherwise involved in the operation of the parking garage.

Multi-level Access: A feature in the admin login that provides different levels of system access based on the role of the personnel (e.g., manager, cashier, security).

Overstay Alerts: Notifications sent to users when their parking time is nearing its expiration, giving them an option to extend if necessary.

Parking Management: The process of efficiently managing parking spaces within a garage, including the addition/removal of spots and updating spot statuses in real-time.

Payment Gateway: An integrated system within SPGMS that allows users to make payments for their parking, offering various payment methods.

Real-time Updates: Instantaneous updates provided by the system about spot availability, ensuring users have the most current information.

Space Utilization: The efficiency with which parking spaces in the garage are used, aiming for maximum occupancy.

SPGMS (Smart Parking Garage Management System): A comprehensive software solution designed to address the challenges faced by parking garages and their users, offering features like spot reservation, real-time spot availability, and analytics for garage operators.

Spot Reservation: A feature that allows users to book a parking spot in advance, ensuring its availability upon their arrival.

Systems Administrator: Person responsible for the design and maintenance of the SPGMS.

User Experience: The overall experience and interaction a user has with the SPGMS, encompassing features like spot reservation, vehicle locator, and real-time updates.

Vehicle Locator: A feature within the SPGMS that assists users in remembering and locating where they parked their vehicle within the garage.

2. Goals, Requirements, and Analysis

a. Business Goals

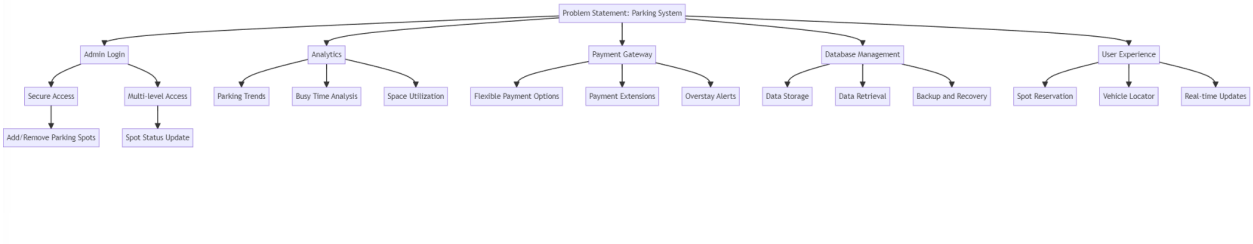


Figure 2-1: Diagram HQ

The Smart Parking Garage Management System (SPGMS) aims to revolutionize the way parking garages operate and how users interact with them. The primary business goals (Figure 2.1) for implementing the SPGMS are:

- Optimize Space Utilization: Ensure that every available parking spot is utilized to its maximum potential, thereby increasing the revenue for garage operators.

- Enhance User Experience: Provide a seamless and hassle-free parking experience for users by offering features like spot reservation, real-time spot availability updates, and vehicle locator.

- Increase Revenue Streams: By offering advanced features such as spot reservation and predictive analysis for pricing, the system aims to create new revenue streams for garage operators.

- Reduce Operational Costs: With real-time data and analytics, garage operators can make informed decisions that can lead to reduced operational costs. For instance, understanding peak times can help in efficient staffing.

- Promote Digital Transactions: By integrating a payment gateway, the system encourages users to make digital payments, reducing the need for physical cash handling and the associated risks.

- Enhance Security: With features like admin login and multi-level access, the system ensures that only authorized personnel can access critical functionalities, thereby maintaining the integrity of the system.

- Data-Driven Decision Making: The analytics provided by the system will empower garage operators to make decisions based on data, such as adjusting pricing during peak times or offering promotions during off-peak hours.

- Environmental Impact: By reducing the time users spend searching for parking spots, the system indirectly contributes to a reduction in vehicle emissions, promoting a greener environment.

- Expand Market Reach: With a user-friendly interface and advanced features, the system aims to attract more users, expanding the market reach for garage operators.

- Future Scalability: The system is designed keeping future growth in mind. As the needs of the garage operators evolve, the system can be scaled up or modified to accommodate new features or expanded operations.

By achieving these business goals, the SPGMS will not only address the current challenges faced by parking garages and their users but also ensure sustainable growth and profitability for garage operators in the future.

b. <u>Functional, Non-Functional, and User Interface Requirements</u>

Requirements are broken down by sub-problem, as described in Section 1 of this report. For each sub-problem, the functional, non-functional, and user interface requirements are provided.

The functional requirements of our system were designed to address the problems faced by our target customers: garage administrators and their customer base of drivers. They are our team's goals for our system presented as concrete, testable statements.

Non-functional requirements were designed using the FURPS structure as described by Marsic (75). FURPS accounts for system properties relating to additional functionality (such as security), usability, reliability, performance, and supportability.

The user interface requirements describe what the website will present on the screen for users to navigate and access the features of the system. Preliminary graphics are provided to demonstrate the proposed layout of the user interface.

Priority weights were calculated on a scale of 1 to 3, with 1 representing requirements of the highest priority. Priority 1 requirements are critical for the system's core functionality and provide significant business value. Priority 2 requirements are important to enhance

user experience or system efficiency, but not critical. Priority 3 requirements are of a lower priority and can be implemented in later phases.

Parking Management

Functional Requirements		
REQ	**Priority**	**Description**
REQ-01	3	The system shall allow administrators to add and remove parking spaces if the physical layout of the garage changes.
Non-Functional Requirements		
REQ-02	1	The system shall update the availability status of each parking spot in real-time.

Admin Login

Functional Requirements		
REQ	**Priority**	**Description**
REQ-03	1	The system shall allow the creation of a variety of administrator accounts, e.g. manager, cashier, security.
Non-Functional Requirements		
REQ	**Priority**	**Description**
REQ-04	1	The system shall provide secure access for parking garage administrator accounts for viewing analytics.

REQ-05	1	The system shall provide secure access for parking garage administrator accounts for managing user data.

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-06	1	The system shall allow for multiple levels of administration accounts. (Figure 2-2)

Figure 2-2

Analytics

Functional Requirements		
REQ	**Priority**	**Description**
REQ-07	2	The system shall display peak parking times to administrators.
REQ-08	2	The system shall display average duration of parking to administrators.
REQ-09	3	The system shall display frequent drivers to administrators. Frequent drivers will be considered drivers who utilize the garage at least once per week for at least a month.
REQ-10	3	The system shall highlight parking areas that are underutilized to administrators. Parking areas shall be considered underutilized if they are empty more than 30% of the peak time frame.
Non-Functional Requirements		
REQ	**Priority**	**Description**
REQ-11	2	The system shall analyze peak parking times.
REQ-12	2	The system shall analyze average duration of parking.
REQ-13	3	The system shall analyze frequency of use by each driver.
REQ-14	3	The system shall analyze frequency of use of each parking space.

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-15	2	The system shall display analytical data for administrators. Figure (2-3)

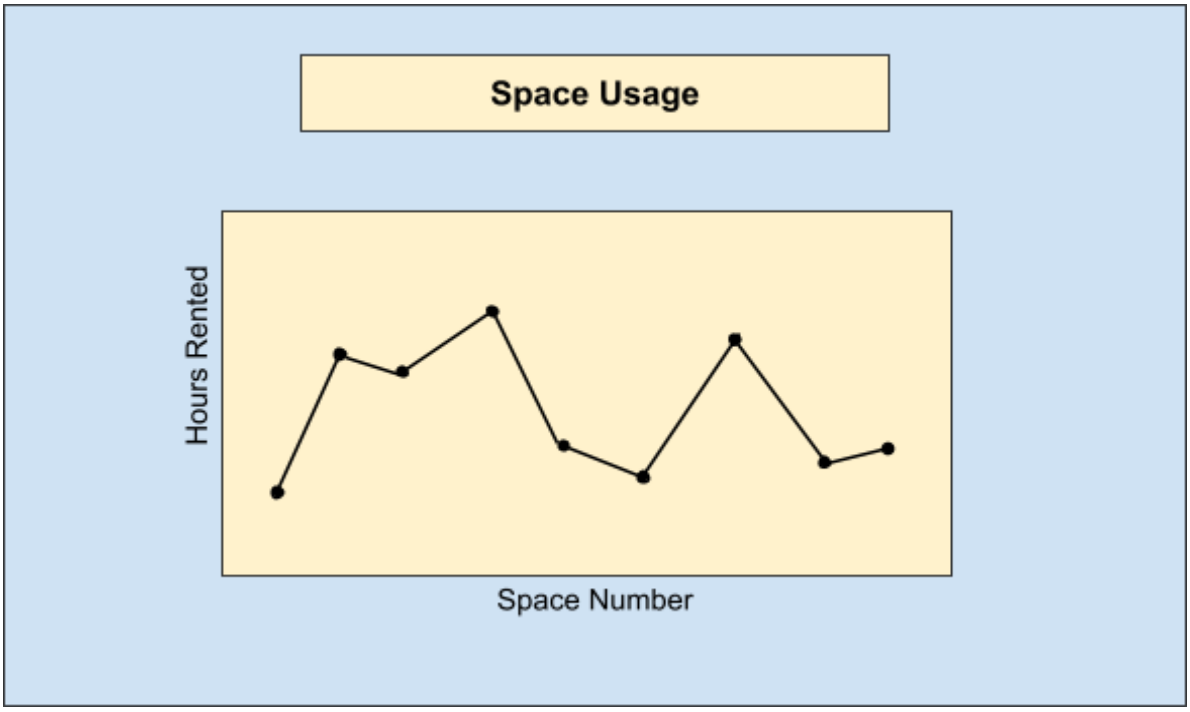


Figure 2-3

Payment Gateway

Functional Requirements		
REQ	**Priority**	**Description**
REQ-16	1	The system shall allow for payment using various methods, such as credit/debit cards, digital wallets, and cash.
REQ-17	2	The system shall allow for extension of parking time and payment.

REQ-18	2	The system shall send alerts when parking time is close to expiration.

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-19	2	The system shall alert the driver when their parking time is within 15 minutes of expiration and offer the option to extend their time. (Figure 2-4)

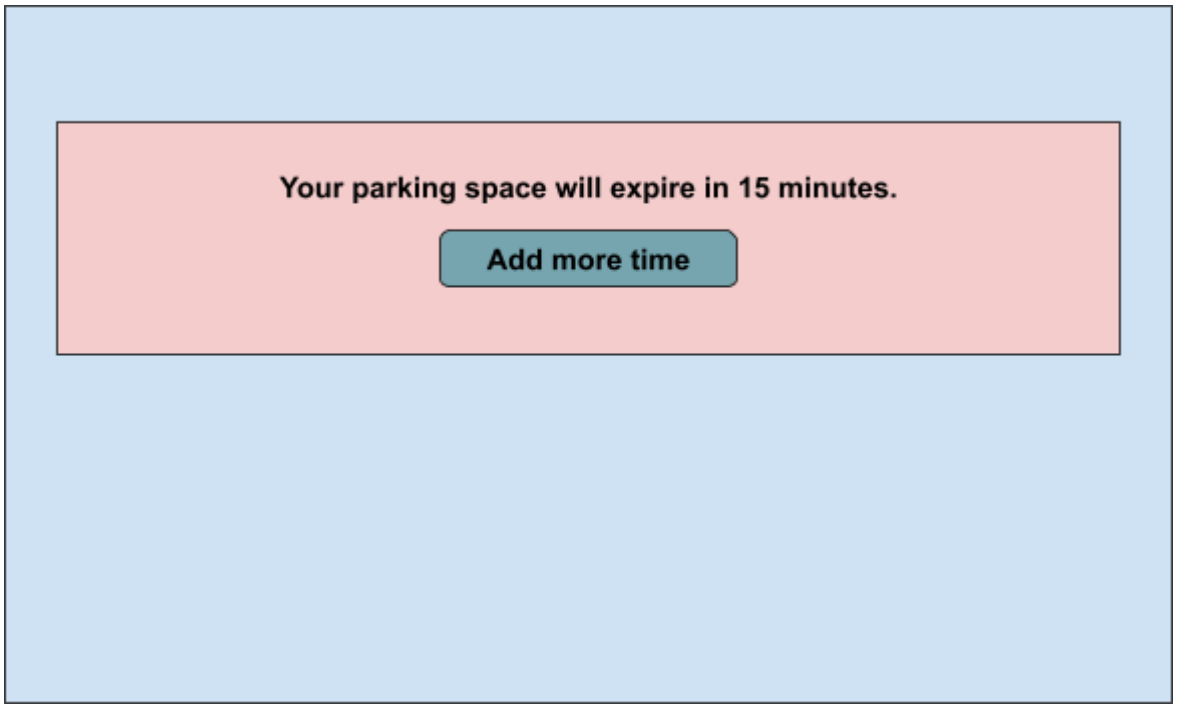


Figure 2-4

Database Management

Non-Functional Requirements		
REQ	**Priority**	**Description**
REQ-20	1	The system shall store all data relating to parking spots, user data, payments, and analytics to be stored securely.
REQ-21	1	The system shall retrieve data quickly.
REQ-22	1	The system shall backup data daily.
REQ-23	1	The system shall provide recovery of data from backups in case of failures.

User Experience

Functional Requirements		
REQ	**Priority**	**Description**
REQ-24	3	The system shall provide vehicle location assistance for users returning to their vehicles.
REQ-25	1	The system shall allow for recurring reservations for drivers needing a regular parking space.
REQ-26	1	The system shall display nearby attractions to interested drivers.

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-27	1	The system shall allow drivers to create personal accounts. (Figure 2-5)

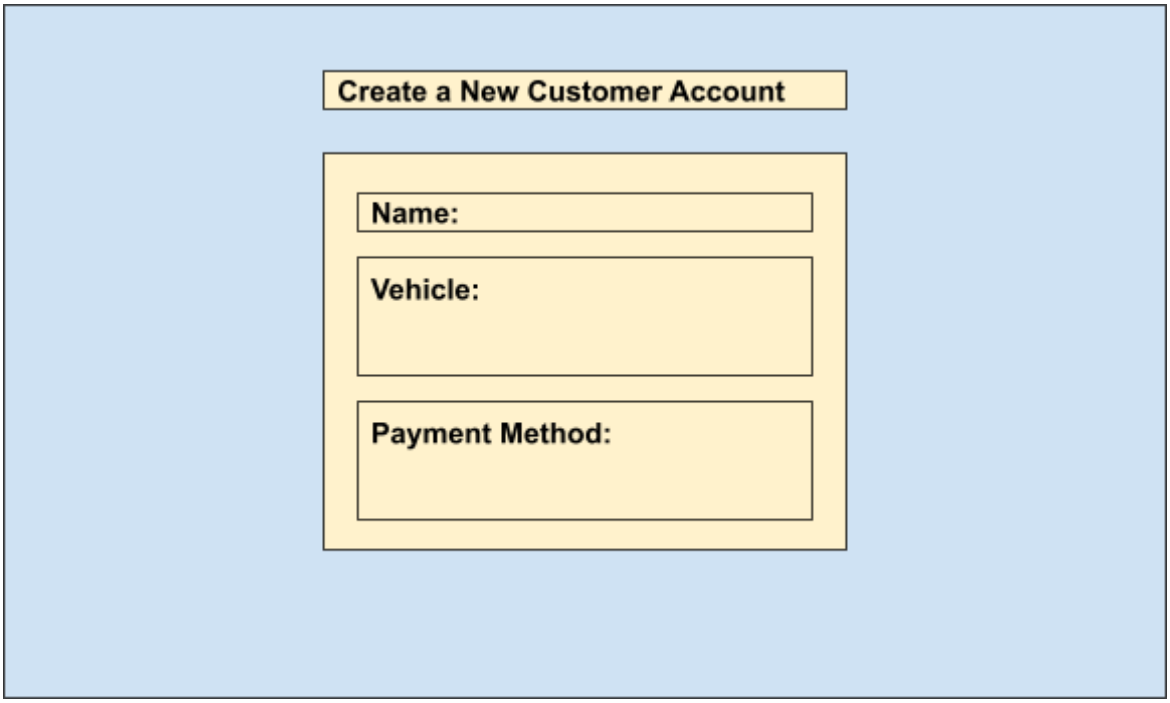


Figure 2-5

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-28	1	The system shall allow the driver to reserve a parking space in advance. (Figure 2-6)

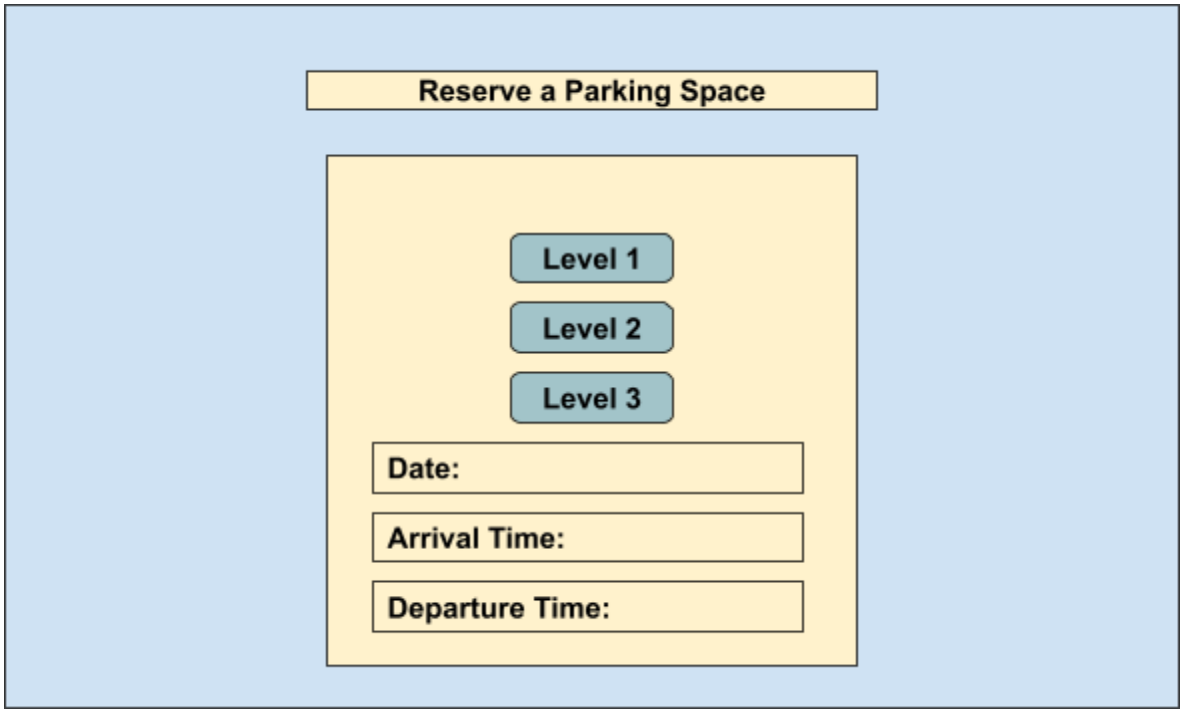


Figure 2-6

User Interface Requirements		
REQ	**Priority**	**Description**
REQ-29	3	The system shall provide vehicle location assistance for drivers returning to their vehicles. (Figure 2-7)

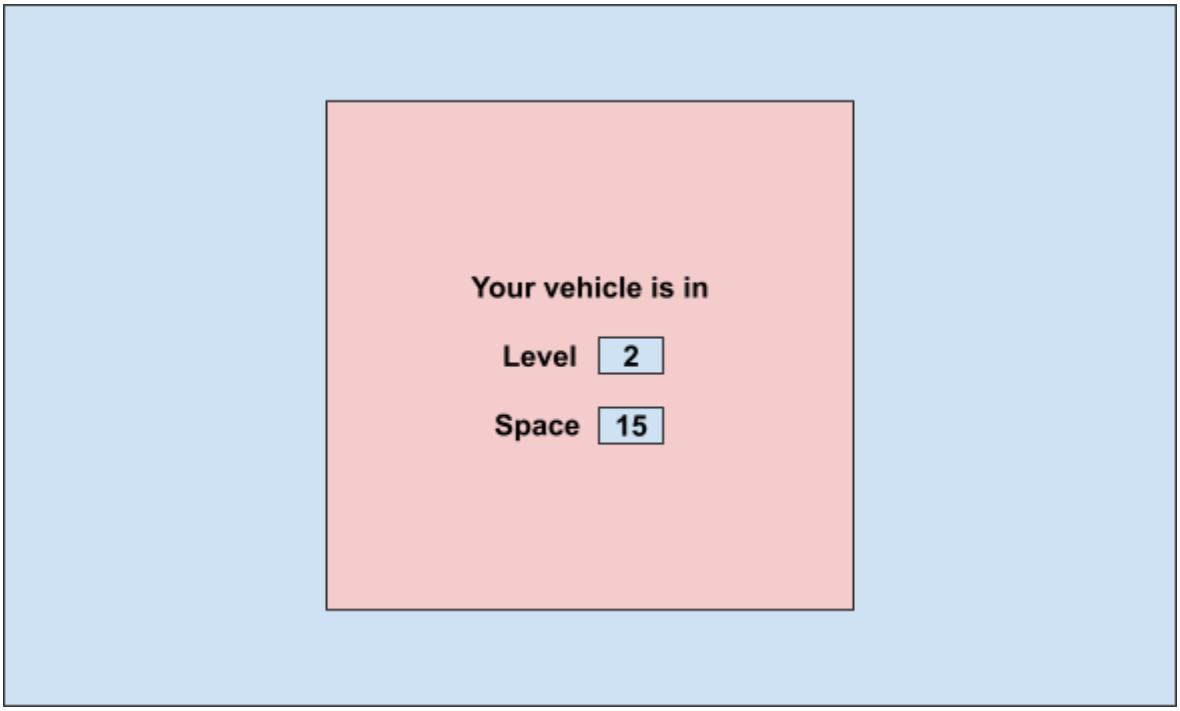


Figure 2-7

3. Use Cases

a. Stakeholders

End Users (Drivers)

- Description: Individuals who will use the Smart Parking Garage Management System to find and reserve parking spaces.
- Needs & Interests:
  - Easy-to-use interface.
  - Real-time availability of parking spaces.
  - Secure payment methods.
  - Quick reservation process.

Parking Garage Owners/Operators

- Description: Entities or individuals who own or operate the parking garages integrated with SPGMS.
- Needs & Interests:

- Efficient space management.
- Accurate reporting and analytics.
- Increase in revenue due to optimized space utilization.
- Seamless integration with existing infrastructure.

System Administrators

- Description: Individuals responsible for the maintenance, updates, and overall health of the SPGMS.
- Needs & Interests:
  - Robust and secure system architecture.
  - Easy troubleshooting and diagnostic tools.
  - Regular updates and patches.

Local Authorities/City Planners

- Description: Government entities or organizations responsible for urban planning and transportation.
- Needs & Interests:
  - Reduction in traffic congestion due to efficient parking management.
  - Data sharing for urban planning and development.
  - Compliance with local regulations and standards.

Third-party Developers/Integrators

- Description: Developers or companies that might want to integrate their solutions or apps with SPGMS.
- Needs & Interests:
  - Open APIs and documentation.
  - Support and community forums.
  - Compatibility with various platforms and technologies.

b. <u>Actors and Goals</u>

Actor: Driver - Daily Commuter

- Primary Goals:
  - Find a parking space near their workplace.
  - Set up recurring reservations for convenience.
  - Receive daily updates or reminders about their reservation.
- Secondary Goals:
  - Opt for discounted monthly or weekly parking rates.
  - Report issues like blocked parking spaces or malfunctioning equipment.

Actor: Driver - Occasional Visitor (e.g., Tourist, Shopper)

- Primary Goals:
  - Locate a parking garage near their destination.
  - Understand the pricing and maximum parking duration.
  - Get directions to the parking garage.
- Secondary Goals:
  - Find out about nearby attractions or businesses.
  - Get recommendations for parking based on their planned activities.

Actor: Garage Administrator - Parking Garage Security Personnel

- Primary Goals:
  - Monitor the security cameras and systems of the parking garage.
  - Handle any security-related incidents or emergencies.
  - Ensure the safety of vehicles and visitors.
- Secondary Goals:
  - Assist visitors with directions or issues.
  - Report maintenance needs or hazards.

Actor: Garage Administrator - Parking Garage Maintenance Staff

- Primary Goals:
  - Ensure all equipment (like ticket machines, barriers, lights) is functional.

    - Address any reported maintenance issues promptly.
    - Keep the parking area clean and free from obstructions.
- Secondary Goals:
    - Schedule regular maintenance checks.
    - Recommend upgrades or replacements for outdated equipment.

Actor: Payment Processing System

- Primary Goals:
    - Process payments securely and promptly.
    - Handle refunds or disputes.
    - Generate payment receipts and transaction records.
- Secondary Goals:
    - Offer multiple payment options (credit card, mobile payment, etc.).
    - Ensure compliance with financial regulations and standards.

Actor: Website

- Primary Goals:
    - Provide a user interface for drivers and garage administrators to access functionality of the system
- Secondary Goals:
    - Serve as an information source for near-by attractions and business promotions

Actor: Database

- Primary Goals:
    - Securely store user data
    - Provide access to data as needed by the system
- Secondary Goals:
    - Store data to be used by Garage Administrators for business analysis purposes

Actor: Local Business Owner (e.g., Restaurant, Shop)

- Primary Goals:

- Collaborate with SPGMS for promotional offers (e.g., validated parking).
- Attract more customers by ensuring convenient parking options.

- Secondary Goals:
  - Advertise their business on the SPGMS platform.
  - Get insights on parking usage to predict customer footfall.

c. Use Cases

i. Casual Description

UC-1: User registers online

- Actor: Driver or Garage Administrator
- Description: The user creates a secure account on the system website.

UC-2: User logs in online

- Actor: Driver or Garage Administrator
- Description: The user accesses their secure account on the system website.

UC-3: Driver makes a reservation online

- Actor: Driver or Garage Administrator
- Description: The Driver makes a reservation from their account on the system website.

UC-4: Driver arrives without a reservation (walk-in customer)

- Actor: Driver or Garage Administrator
- Description: A walk-in customer arrives at the garage to park their vehicle without a reservation.

UC-5: Vehicle parks in a space

- Actor: Driver
- Description: A Driver checks into the garage and parks in a space.

UC-6: Vehicle exits the garage

- Actor: Driver
- Description: A Driver exits the garage.

UC-7: Daily Commuter reserves a recurring spot

- Actor: Daily Commuter
- Description: The commuter uses the system to reserve a parking spot near their workplace on a recurring basis, ensuring they have a spot every workday.
- Steps:
    - Commuter logs into the SPGMS.
    - Searches for a parking garage near their workplace.
    - Selects a recurring reservation option.
    - Sets the frequency (e.g., every weekday).
    - Makes a payment or links to a payment method.
    - Receives a confirmation of the recurring reservation.
    -

UC-8: Occasional Visitor finds nearby attractions

- Actor: Occasional Visitor
- Description: After parking, the visitor uses the system to find nearby attractions or businesses.
- Steps:
    - Visitors access the SPGMS after parking.
    - Selects the option to view nearby attractions.
    - Browse through a list of recommended places.
    - Chooses a place and gets directions or details.

UC-9: Security Personnel monitors garage activity

- Actor: Parking Garage Security Personnel
- Description: The security personnel use the system's integrated cameras and sensors to monitor activity and ensure safety.
- Steps:

- Security logs into the security module of SPGMS.
- Views live feeds from various cameras.
- Receives alerts for any suspicious activity.
- Takes necessary action based on alerts.

UC-10: Maintenance Staff schedules equipment check

- Actor: Parking Garage Maintenance Staff
- Description: The maintenance staff schedules regular checks for equipment like ticket machines and barriers.
- Steps:
  - Maintenance staff accesses the maintenance module of SPGMS.
  - Schedules a check for specific equipment.
  - Receives reminders as the scheduled date approaches.
  - Logs any issues found and actions taken.

UC-11: Payment processing for parking fees

- Actor: Payment Processing System
- Description: The system processes payments made by users for parking reservations.
- Steps:
  - User selects a parking spot and duration.
  - User chooses a payment method.
  - Payment system processes the payment securely.
  - User receives a payment confirmation and receipt.

UC-12: Local Business offers parking discounts

- Actor: Local Business Owner
- Description: A local restaurant owner collaborates with SPGMS to offer parking discounts to their customers.
- Steps:
  - Business owner logs into the SPGMS business portal.

- Sets up a promotional offer (e.g., 1-hour free parking with a meal purchase).
- Provides validation codes or methods to customers.
- Tracks the usage and success of the promotion.

ii. Use Case Diagram

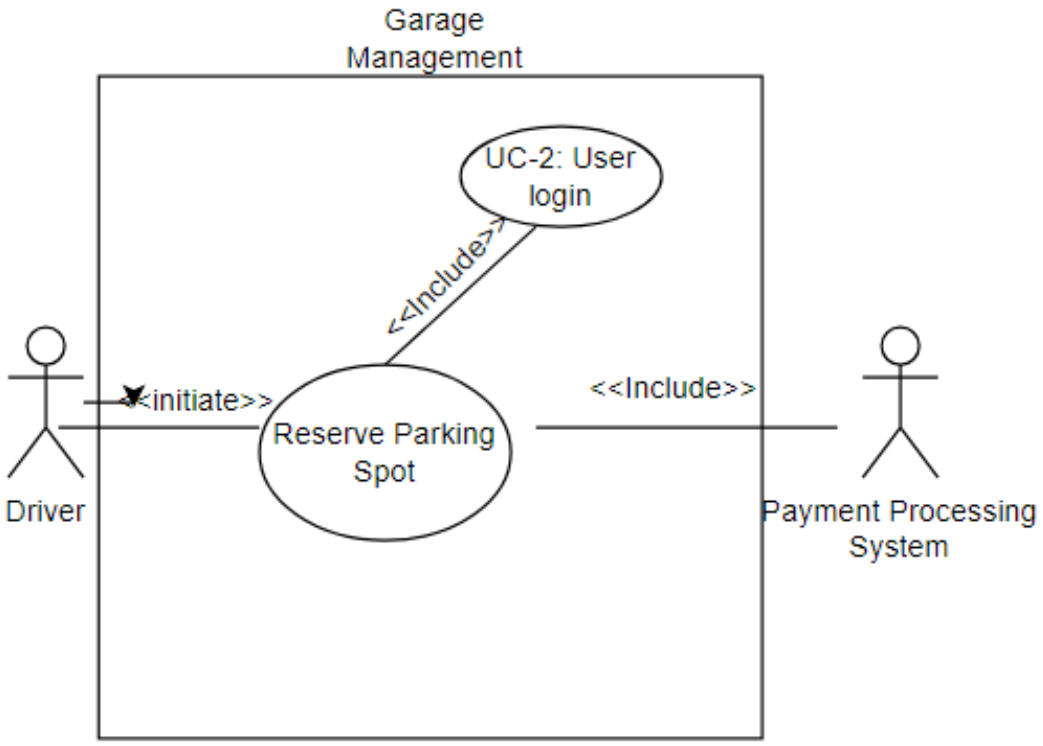


Figure 3-1

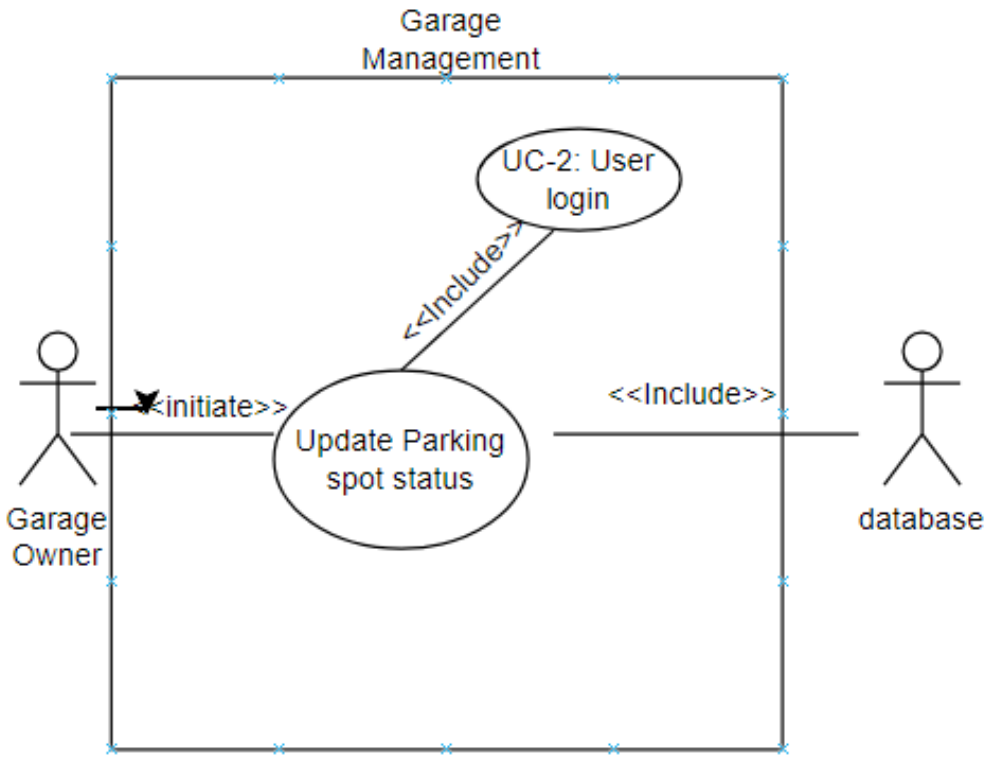


Figure 3-2

iii. Traceability Matrix

The Traceability Matrix will map the system requirements outlined in the Functional, Non-Functional, and User Interface Requirements section of this report to the use cases. The format of the Traceability Matrix was influenced by Full Report 1 of the Blockchain and Docker Assisted Secure Automated Parking Garage System (24).

REQ	Priority Weight (PW)	USE CASE					
		User Registers	User Logs In	Reservation is Made	Vehicle Parks	Vehicle Exits	Analytics Viewed
01	3		X				
02	1			X	X	X	
03	1	X					
04	1		X				
05	1		X				
06	1	X	X				
07	2		X				X
08	2		X				X
09	3		X				X
10	3		X				X
11	2		X				X
12	2		X				X
13	3		X				X
14	3		X				X
15	2		X				X
16	1		X	X	X		
17	2		X	X	X		
18	2			X	X		
19	2		X	X	X		
20	1	X					
21	1	X					X
22	1	X					X
23	1	X					X

REQ	Priority Weight (PW)	USE CASE					
		User Registers	User Logs In	Reservation is Made	Vehicle Parks	Vehicle Exits	Analytics Viewed
01	3		X				
02	1			X	X	X	
03	1	X					
04	1		X				
05	1		X				
06	1	X	X				
24	3		X		X	X	
25	1	X	X	X			
26	1	X	X		X		
27	1	X					
28	1		X	X			
29	3		X				

iv. Fully-Dressed Descriptions and System Sequence Diagrams

UC-4: Driver arrives without a reservation (walk-in customer)
Related Requirements: REQ-02, REQ-13, REQ-16, REQ-20
Initiating Actor: Driver
Actor's Goal: Obtain a parking space immediately.
Participating Actors: Driver, Payment Processing System
Preconditions: The Driver has arrived in person at the garage and there is a space available.

Postconditions: The Driver's vehicle is parked in the garage.	
Flow of Events for Main Success Scenario:	
→	1. Driver checks in at the Entrance Security Station.
→	2. Security Personnel accesses Admin Portal, Check-In Screen.
→	3. Security Personnel accesses Payment Portal.
→	4. Payment information is saved.
←	5. Driver parks.
→	6. Driver checks out at the exit security station.
→	7. Security Personnel accesses Admin Portal, Check-Out Screen.
←	8. Payment is processed based on length of stay.

UC-7: Daily Commuter Reserves a Recurring Spot (Figure 3-3)
Related Requirements: REQ-02, REQ-09, REQ-13, REQ-16, REQ-18, REQ-19, REQ-20, REQ-25, REQ-26
Initiating Actor: Driver
Actor's Goal: Reserve a parking space on a recurring basis
Participating Actors: Driver, Website, Database, Payment Processing System
Preconditions: The Driver has a user account in the SPGMS System.

Postconditions: There is a space automatically allocated to the Driver on a recurring basis.	
Flow of Events for Main Success Scenario:	
→	1. Driver submits a request for a recurring parking space for a specified time frame.
→	2. Website searches Database for spaces available for the requested time frame.
←	3. Website displays available spaces to Driver.
→	4. Driver selects a space to reserve.
→	5. Website opens Payment Processing System.
←	6. Payment Processing System requests payment from Driver.
→	7. Driver provides payment information securely on the Payment Processing System.
←	8. Space is reserved in the Database on a recurring basis.
←	9. Website displays reservation to Driver.

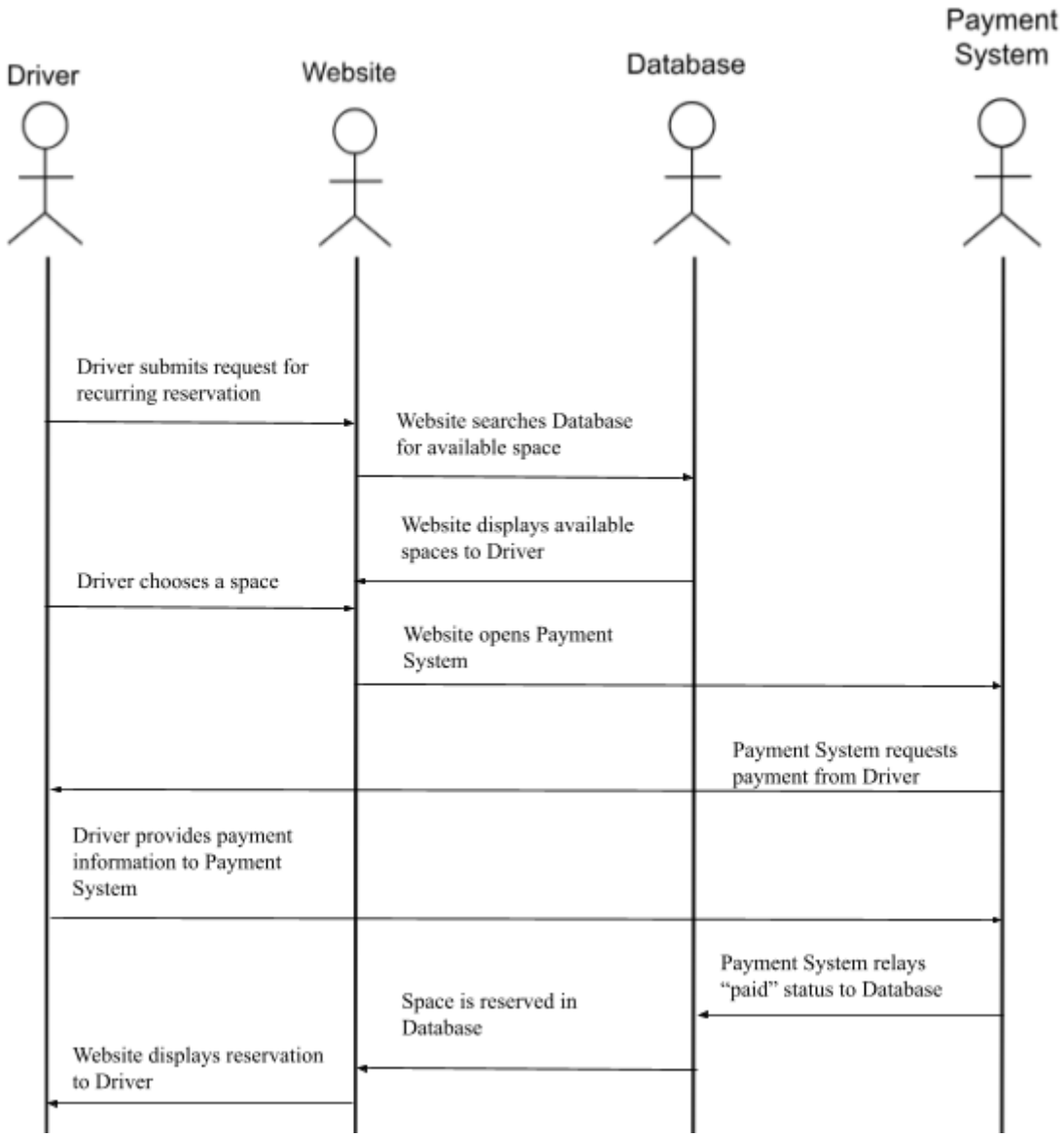


Figure 3-3. System Sequence Diagram for Use Case: Daily Commuter Reserves a Recurring Spot

UC-8: Occasional Visitor Finds Nearby Attractions (Figure 3-4)	
Related Requirements: REQ-26	
Initiating Actor: Driver	
Actor's Goal: After parking in the garage, use the system to find nearby attractions or businesses.	
Participating Actors: Driver, Website	
Preconditions: The Driver has parked in a smart parking garage space and needs directions to a nearby attraction or business.	
Postconditions: The System directs the Driver to a desired destination.	
Flow of Events for Main Success Scenario:	
→	1. Driver accesses the SPGMS Website.
→	2. Driver selects the option to view nearby attractions.
←	3. Website displays a list of recommended places.
→	4. Driver chooses a destination.
←	5. Website provides details and directions to the selected destination.

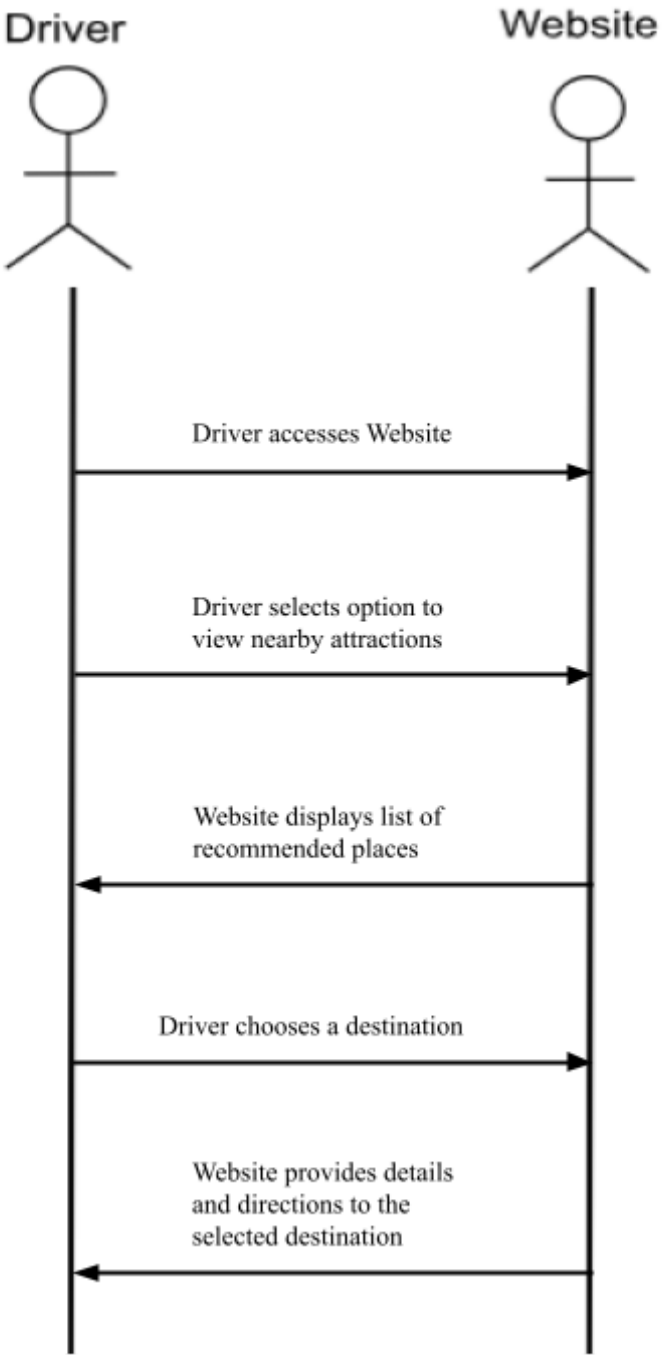


Figure 3-4: Use Case: Occasional Visitor Finds Nearby Attraction

UC-9: Security Personnel Monitors Garage Activity (Figure 3-5)	
Related Requirements: REQ-02, REQ-03, REQ-06	
Initiating Actor: Parking Garage Security Personnel	
Actor's Goal: Identify and respond to any security issues within the parking garage from a single station/location.	
Participating Actors: Parking Garage Security Personnel, System, Cameras	
Preconditions: Security Personnel is on staff in a parking garage equipped with security cameras. The Security Personnel has access to a SPGMS account with security level permissions.	
Postconditions: Security issues within the parking garage are identified by appropriate personnel.	
Flow of Events for Main Success Scenario:	
→	1. Security Personnel accesses an administrator account with security level permissions.
→	2. The security account accesses video feed from cameras throughout the parking garage.
←	3. The system displays the video feed to authorized security personnel.

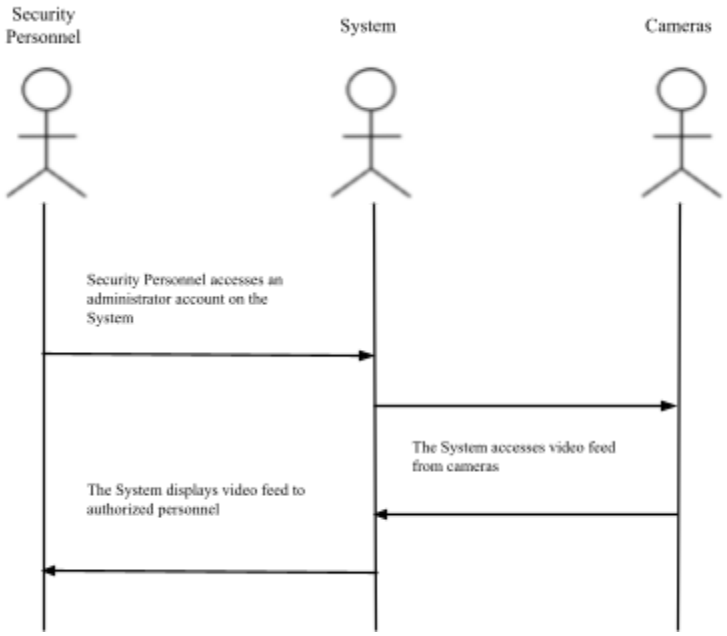


Figure 3-5: Use Case: Security Personnel Monitors Garage Activity

4. User Interface Specification

a. Preliminary Design

The preliminary design for the screen that will be displayed to the driver to schedule a recurring reservation for a parking space is shown in Figure 4-1.

The driver will select the days of the week that they wish to reserve a space. Then, they will type in the arrival and departure time for each day. They will enter the dates they would like to begin and end the schedule.

The display of the parking spaces will change color to reflect the availability of each space according to the dates and times entered by the driver. The driver may click on the buttons for additional levels to view more spaces. Once the driver has decided on an available space, they may select the space. When all of the date and time information has been entered and a space has been selected, the driver will click on the “proceed to payment” button to move to the next screen.

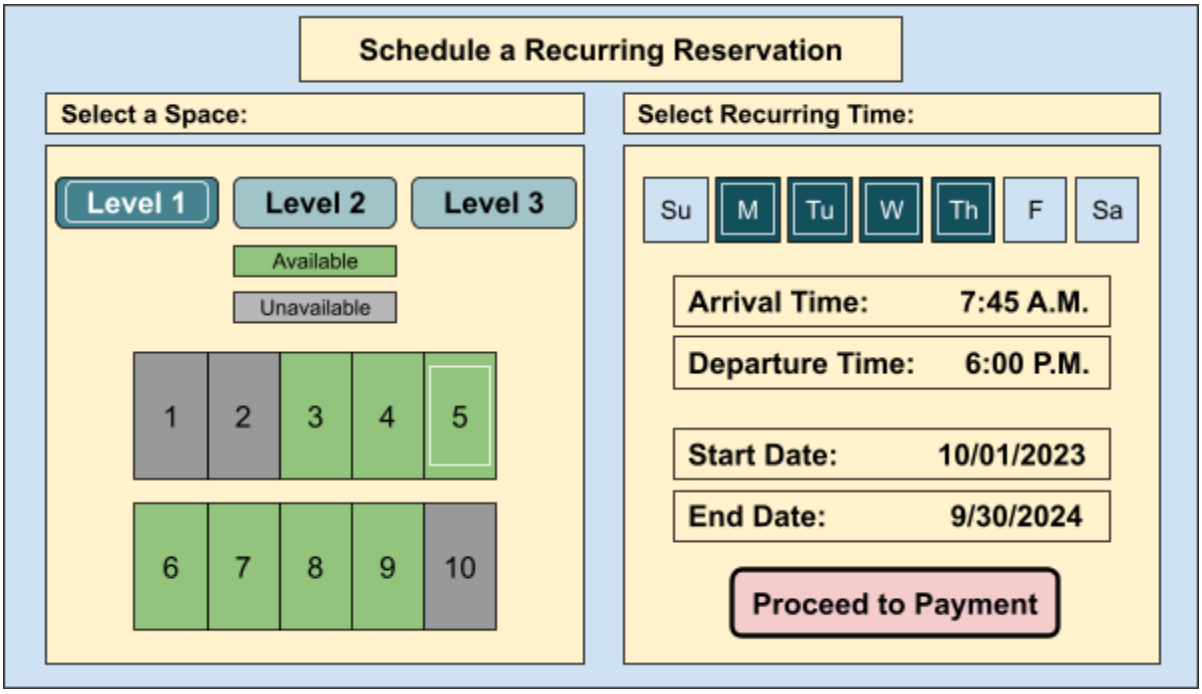


Figure 4-1: Use Case: Daily Commuter Reserves a Recurring Spot

b. User Effort Estimation

For Use Case: Daily Commuter Reserves a Recurring Spot, the user will begin by clicking a button to schedule a recurring reservation on their personal SPGMS dashboard. Counting this button as the first click, this use case should require between 8 and 14 clicks, depending on how many days of the week the user selects, and how many levels of the garage they view. Keystrokes for this use case will be limited to typing the times and dates, and then entering their payment information on the following screen. These totals assume that the user already had an existing SPGMS account.

5. System Architecture

a. Identifying Subsystems

The SPGMS subsystems are outlined in the package diagram in Figure 5-1.

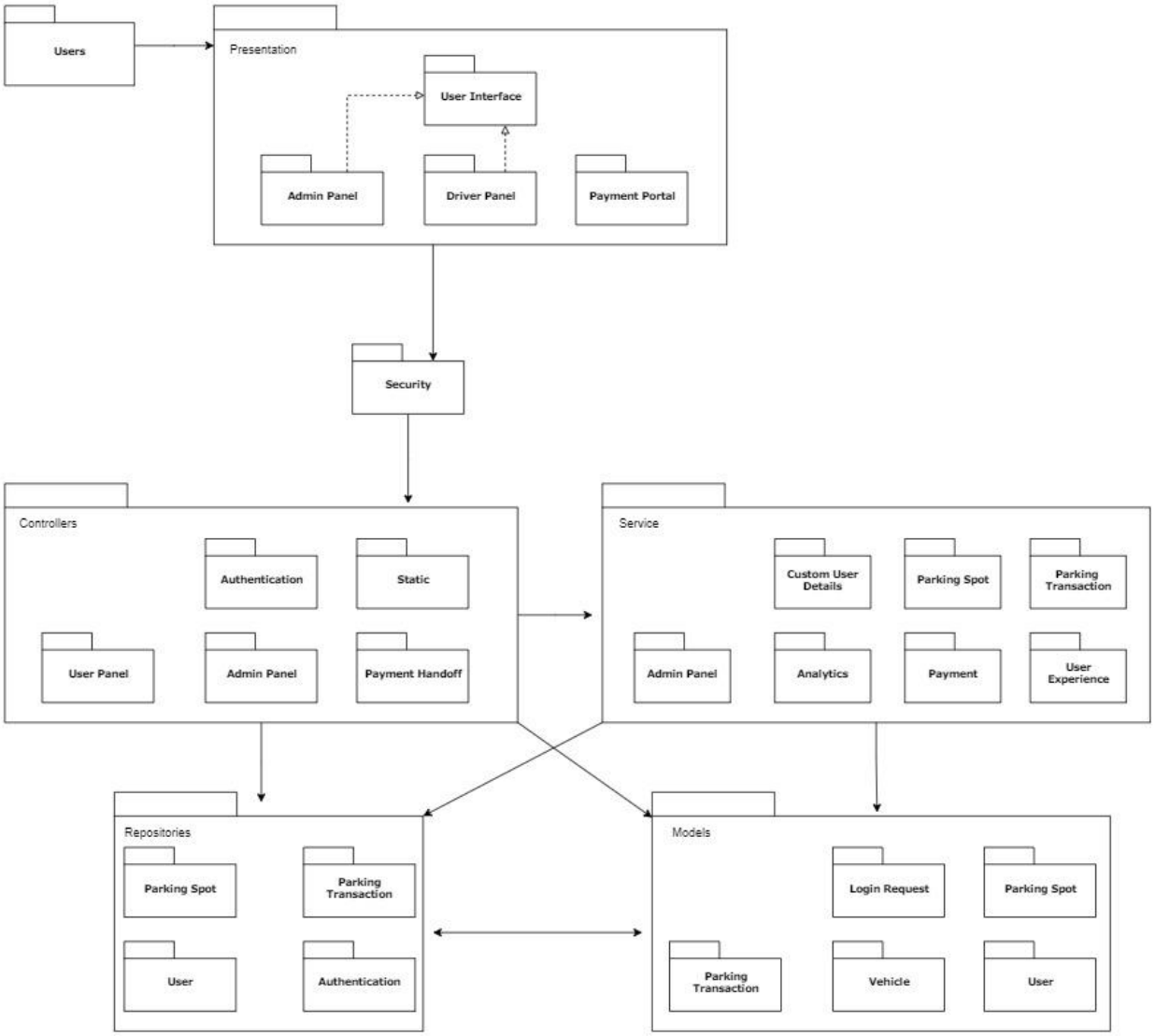


Figure 5-1: SPGMS Package Diagram

b. Architectural Style

The system architecture has a layered style, as described by Sharma, Kumar and Agarwal (2015). The layered style features three basic layers: presentation, business, and infrastructure. The SPGMS presentation layer contains the user interface, which includes the user panels (driver and administrator), the payment portal, and the security packages. The business layer consists of the controllers and service packages. The infrastructure layer contains the repositories.

The system also contains elements of client-server architecture, which is used by many applications that access data via the internet (Sharma, Kumar and Agarwal 2015). The website (client) will request information from the server database and return it to the user as needed.

c. <u>Connectors and Network Protocols</u>

The Smart Parking Garage Management System (SPGMS) is designed with a focus on secure, efficient, and reliable communication between its various components. Here's an overview of the connectors and network protocols employed:

Spring Boot Server on EC2:

- Connector: Embedded Tomcat server, which comes by default with Spring Boot, will be used as the servlet container to deploy the application.
- Protocol: HTTPS (HTTP over TLS/SSL) will be employed to ensure secure communication between the client and the server.
- Port: Production will use port 8443.

Communication between EC2 and RDS:

- Connector: JDBC (Java Database Connectivity) will be used to connect the Spring Boot application with the RDS databases.
- Protocol: Secure Socket Layer (SSL) encryption will be enabled for connections between the EC2 instance and RDS to ensure the confidentiality and integrity of data during transmission.
- Port: The default port for MySQL is 3306.

By employing these connectors and protocols, SPGMS ensures a robust and secure communication framework, facilitating efficient data exchange and real-time updates where necessary.

d. Global Control Flow

The SPGMS will operate in real-time. The status of the parking spaces (available or occupied) will have to be accurate in real-time in order for reservations to be made remotely and for costs to be calculated accurately. Business analysis functions will be available for Garage Administrators based on blocks of time of various lengths, such as hourly, daily, monthly, and yearly.

The Driver functions of the system will be procedure-driven. The reservation and payment steps for the Driver will always happen in the same order. However, the analysis functions available to the Garage Administrator will be accessible in any order, and will therefore be more event-driven. For example, a Garage Administrator will be able to view the annual data for the garage without first viewing the hourly, daily, and monthly data if desired.

e. Hardware Requirements

The Smart Parking Garage Management System (SPGMS) will be hosted on Amazon Web Services (AWS), leveraging its robust and scalable infrastructure. Below are the AWS resources and potential hardware components that will be utilized:

- Amazon EC2 (Elastic Compute Cloud): This service will provide the resizable compute capacity required to run the SPGMS application. EC2 instances will be configured to ensure optimal performance, security, and scalability.
- Amazon S3 (Simple Storage Service): S3 will be used for storing and retrieving any data, assets, or backups related to SPGMS. It offers high durability and availability, ensuring that our data remains safe and accessible.
- Amazon RDS (Relational Database Service): RDS will host the relational databases for SPGMS. It will be set up to ensure data integrity, security, and high availability. Regular backups and maintenance tasks will be scheduled to keep the database in optimal condition.
- Amazon Route 53: This will be the DNS web service used to route end users to the SPGMS application. It will ensure that the domain name of SPGMS resolves quickly and reliably to the application.

- Amazon SES (Simple Email Service): SES will handle all email communications for SPGMS, ensuring that notifications, alerts, and other communications are delivered promptly and securely to the intended recipients.
- Amazon ELB (Elastic Load Balancing): To distribute incoming application traffic across multiple EC2 instances, ensuring fault tolerance and high availability.
- Amazon CloudFront: A content delivery network (CDN) service that will help deliver content to users with low latency and high transfer speeds.

Physical Hardware Components:

- Cameras: For surveillance and license plate recognition, aiding in the automation of entry and exit points.
- Gate Control Hardware: Automated gates or barriers that can be controlled by the system to manage vehicle entry and exit.
- Payment Kiosks: Physical machines where users can make payments for parking if not done through the application.
- Display Boards: To show real-time parking availability, directions, and other relevant information to users.

By leveraging these AWS resources and hardware components, SPGMS aims to provide a seamless and efficient experience for its users, ensuring high availability, security, and scalability.

6. Analysis and Domain Modeling

a. Conceptual Model

The sub-problems that are of the highest priority for the system's functionality are account creation, account log-in, parking space reservation, and the admission of Drivers into the garage. For the purposes of this report, domain models were created for parking space reservation and the admission of Drivers into the garage.

The process that was followed in creating the domain models was similar to the one recommended by Marsic (109). The concepts that involved interactions between the system and the actors were identified first. The internal concepts were extrapolated later.

The Domain Models for the two major sub-problems that will be focused on first are shown in Figures 6-1 and 6-2.

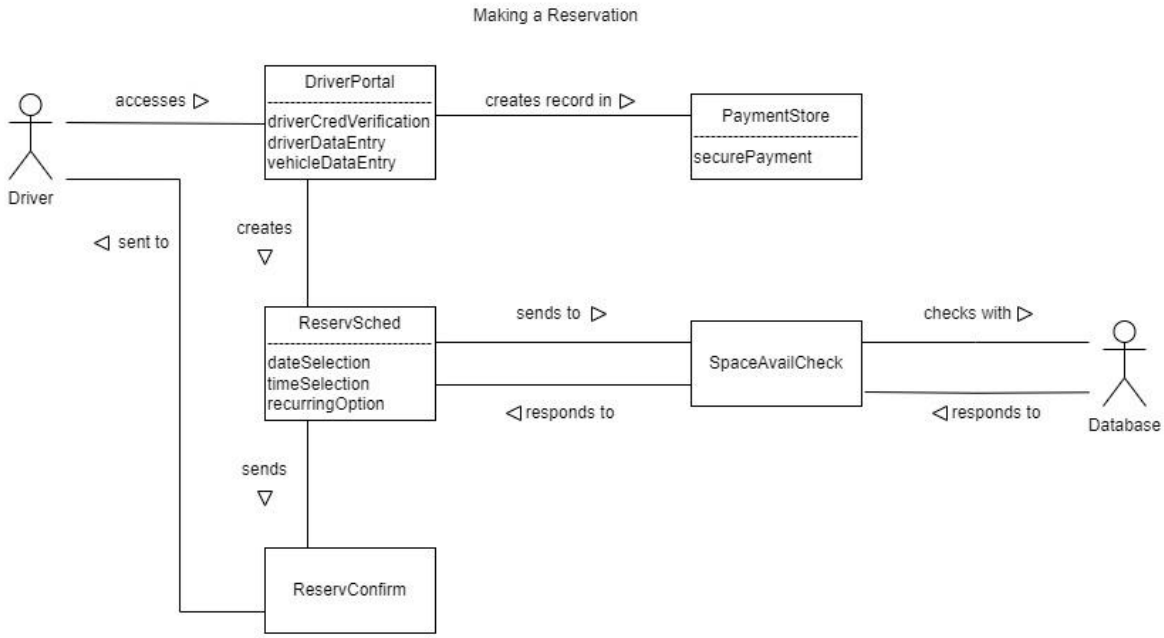


Figure 6-1: Domain Model for Making a Reservation

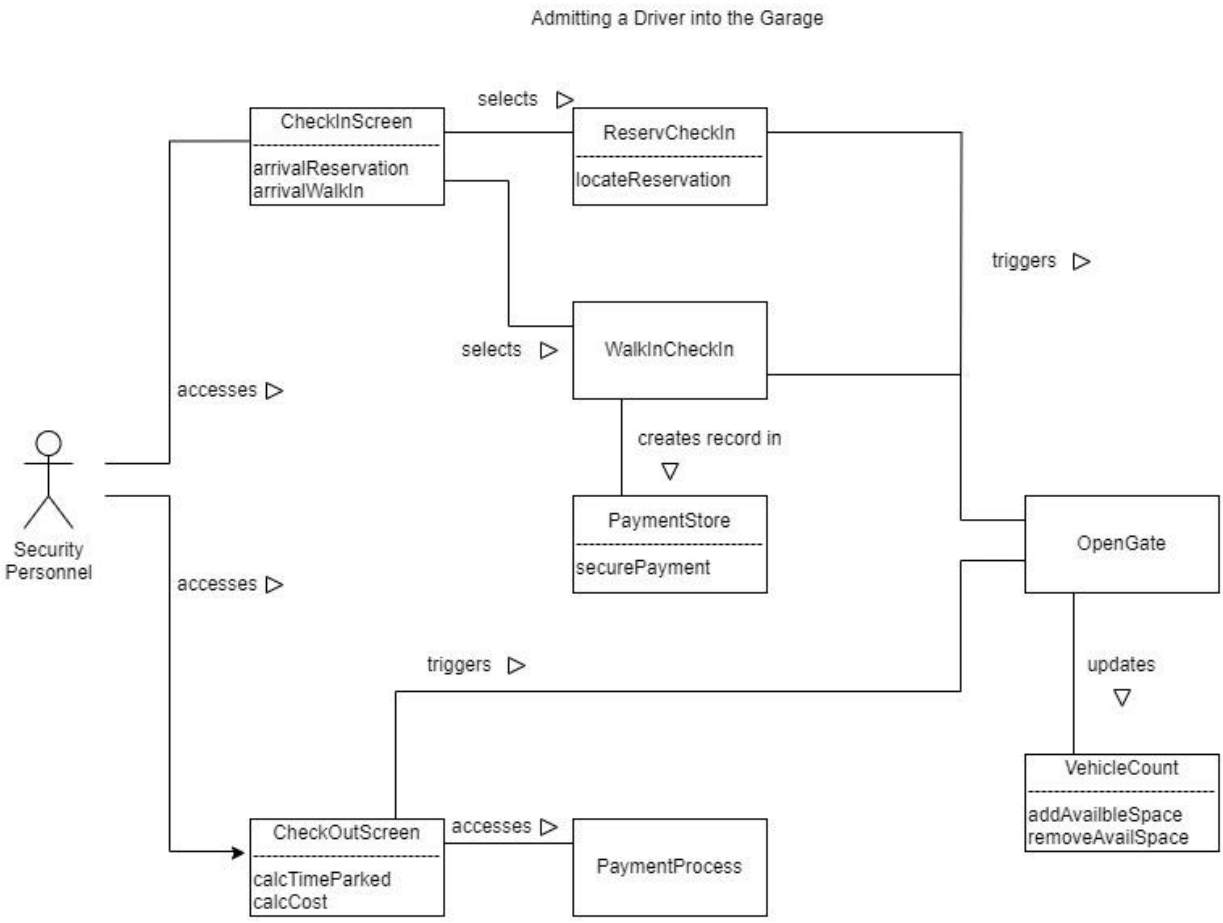


Figure 6-2: Domain Model for Admitting a Driver into the Garage

i. Concept Definitions

The Concept Definitions table includes the name of the concept, the responsibility associated with the concept, and the type. In this case, type refers to doing (D) or knowing (K), indicating whether the concept focuses on an action or the storage of information.

Concept Definitions: Making a Reservation		
Concept Name	**Type**	**Responsibility Description**
DriverPortal	K	User interface that accesses profile, payment, and reservation information.

PaymentStore	K	Securely stores payment information saved by Drivers in their profiles.
ReservSched	D	Schedules a reservation date and time.
SpaceAvailCheck	D	Accesses garage data in the database to determine if the requested reservation date and time are possible.
ReservConfirm	D	Provides confirmation to the Driver if the reservation has been completed.

Concept Definitions: Admitting a Driver into the Garage		
Concept Name	**Type**	**Responsibility Description**
CheckInScreen	D	User interface whereby Security Personnel selects whether an arriving Driver has a reservation or is a walk-in customer.
ReservCheckIn	K	Retrieves reservation information from database.
WalkInCheckIn	D	Directs Security Personnel to Payment Portal.
PaymentStore	K	Stores information entered by Security Personnel for a walk-in customer.
CheckOutScreen	D	User interface whereby Security Personnel checks out a customer exiting the garage.
PaymentProcess	D	Processes payment based on time stayed for a Driver exiting the garage.
OpenGate	D	Command triggered by Security Personnel to open a gate, allowing Drivers to enter or exit the garage.
VehicleCount	D	Monitors the number of vehicles currently in the garage, the number of spaces being held for reservation, and the number of spaces available for new reservations or walk-in customers.

ii. Association Definitions

The associations between each concept are elaborated in the following tables.

Association Definitions: Making a Reservation		
Name	**Association Description**	**Concept Pair**
accesses	The Driver accesses the Driver Portal by logging in with their username and password.	Driver, DriverPortal
creates record in	Payment information entered in the DriverPortal user interface is stored securely by the PaymentStore.	DriverPortal, PaymentStore
creates	Information entered into the DriverPortal is passed to ReservSched to schedule a reservation.	DriverPortal, ReservSched
sends to / responds	The date and time of a requested reservation is processed, and the availability status is returned to the Driver.	ReservSched, SpaceAvailCheck
checks with / responds	A requested reservation is compared to the record of available spaces, and the availability status is determined.	SpaceAvailCheck, Database
sends	After a reservation has been determined to be possible, a confirmation message is created.	ReservSched, ReservConfirm
sent to	The reservation confirmation message is passed to the Driver.	ReservConfirm, Driver

Association Definitions: Admitting a Driver into the Garage		
Name	**Association Description**	**Concept Pair**
accesses	Security Personnel accesses the command to check in a Driver by logging into their Admin Account with their username and password.	Security Personnel, CheckInScreen
selects	Security Personnel clicks option to check in a Driver with a reservation.	CheckInScreen, ReservCheckIn
selects	Security Personnel clicks option to check in a walk-in customer.	CheckInScreen, WalkInCheckIn

creates record in	A walk-in customer provides their payment information to be entered by the Security Personnel and stored in the Payment Portal.	WalkInCheckIn, PaymentStore
triggers	Security Personnel clicks option to complete the check in of a Driver and open the gate into the garage.	ReservCheckIn or WalkInCheckIn, OpenGate
accesses	Security Personnel accesses the command to check in a Driver by logging into their Admin Account with their username and password.	Security Personnel, CheckOutScreen
accesses	The Payment Portal is accessed to process payment.	CheckOutScreen, PaymentProcess
triggers	Security Personnel clicks option to complete the check out of a Driver and open the gate exiting the garage.	CheckOutScreen, OpenGate
updates	The counts of vehicles in the garage and available spaces are updated.	OpenGate, VehicleCount

iii. Attribute Definitions

Attribute Definitions: Making a Reservation		
Concept	**Attributes**	**Attribute Description**
DriverPortal	driverCredVerif	Verify username and password
	driverDataEntry	Allow for changes to Driver information
	vehicleDataEntry	Allow for changes to vehicle information.
PaymentStore	securePayment	Ensure that the payment method is stored securely.
ReservSched	dateSelection	Allow for the selection of a reservation date.
	timeSelection	Allow for the selection of a reservation time frame.
	recurringOption	Allow for the selection of a recurring reservation.

Attribute Definitions: Admitting a Driver into the Garage		
Concept	**Attributes**	**Attribute Description**
CheckInScreen	arrivalReserv	Directs actor to the Check-In screen for a Driver with a parking reservation
	arrivalWalkIn	Directs actors to the Check-In screen for a walk-in customer.
ReservCheckIn	locateReserv	Retrieves reservation record from the Database.
PaymentStore	securePayment	Ensure that payment methods are stored securely.
CheckOutScreen	calcTimeParked	Calculate the time the exiting Driver spent in the parking garage.
	calcCost	Calculate the cost owed by the exiting Driver.
VehicleCount	addAvailSpace	Decrease the count of vehicles and increase the count of available spaces in the garage.
	remAvailSpace	Increase the count of vehicles and decrease the count of available spaces in the garage.

iv. Traceability Matrix

Traceability Matrix: Making a Reservation							
		Domain Concepts					
Use Case	PW	DriverPortal	PaymentStore	ReservSched	SpaceAvailCheck	ReservConfirm	PaymentProcess
UC-1	1		X				
UC-2	1		X				
UC-3	1		X	X	X	X	
UC-4	1	X	X				X
UC-5	1	X	X				X
UC-6	1	X					X
UC-7	1	X	X	X	X	X	X

Traceability Matrix: Admitting a Driver into the Garage									
		Domain Concepts							
Use Case	PW	CheckInScreen	ReservCheckIn	WalkInCheckin	PaymentStore	CheckOutScreen	PaymentProcess	OpenGate	VehicleCount
UC-1	1	X	X		X	X			
UC-2	1	X	X		X	X			
UC-3	1	X	X		X	X			
UC-4	1	X		X	X	X		X	X
UC-5	1	X						X	X
UC-6	1					X	X	X	X
UC-7	1	X	X		X			X	X
UC-8	3	X						X	X
UC-11	1				X		X		
UC-12	3						X		

b. <u>System Operation Contracts</u>

UC-1: User Registration

- Precondition:
  - Functioning database to store user account information
  - Driver is not an existing user
  - Username, email address and password
  - Make and model of car
- Postcondition:
  - Driver account created and stored in database

UC-2: User logs in online

- Precondition:
  - Driver enters username and password

    - Driver is a registered user
- Postcondition:
    - Driver logged in

UC-3 Driver makes a reservation online

- Precondition:
    - Driver select available spot
    - Driver selects time/date
    - Driver pays
- Postcondition:
    - Spot reserved

UC-4 Driver arrives without a reservation (walk-in customer)

- Precondition:
    - Spot is available
    - Driver pays
- Postcondition:
    - Spot is reserved

UC-5: Vehicle parks in a space

- Precondition:
    - Driver has reservation for spot
- Postcondition:
    - Spot becomes unavailable to be reserved while the car is parked.

UC-6: Vehicle exits the garage

- Precondition:
    - Driver leaves the spot
- Postcondition:
    - Spot becomes available

UC-7: Daily commuter reserves a recurring spot

- Precondition:
  - Driver selects spot
  - Driver pays
  - Discount applied
- Postcondition:
  - Spot is reserved

UC-8: Occasional Visitor finds nearby attraction

- Precondition:
  - Driver selects spot
  - Driver pays
  - Discount applied
- Postcondition:
  - Spot is reserved

UC-9: Security Personnel monitors garage activity

- Precondition:
  - Security personnel verifies cars are parked in appropriate spots
  - Security personnel verifies no unauthorized parking has occurred
  - Security personnel prevents mischievous activities
- Postcondition:
  - Lot secured

UC-11: Payment processing for parking fees

- Precondition:
  - Driver enters payment preference
- Postcondition:
  - Payment processed

UC-12: Local Business offers parking discounts

- Precondition:
  - Local businesses registers for account
- Postcondition:
  - Local business discounts is available for drivers

c. Data Model and Persistent Data Storage

The Smart Parking Garage Management System (SPGMS) will utilize a relational database hosted on AWS to store and manage its data. The primary entities or tables in this database will be Parking Spot, Parking Transaction, User, and Authentication.

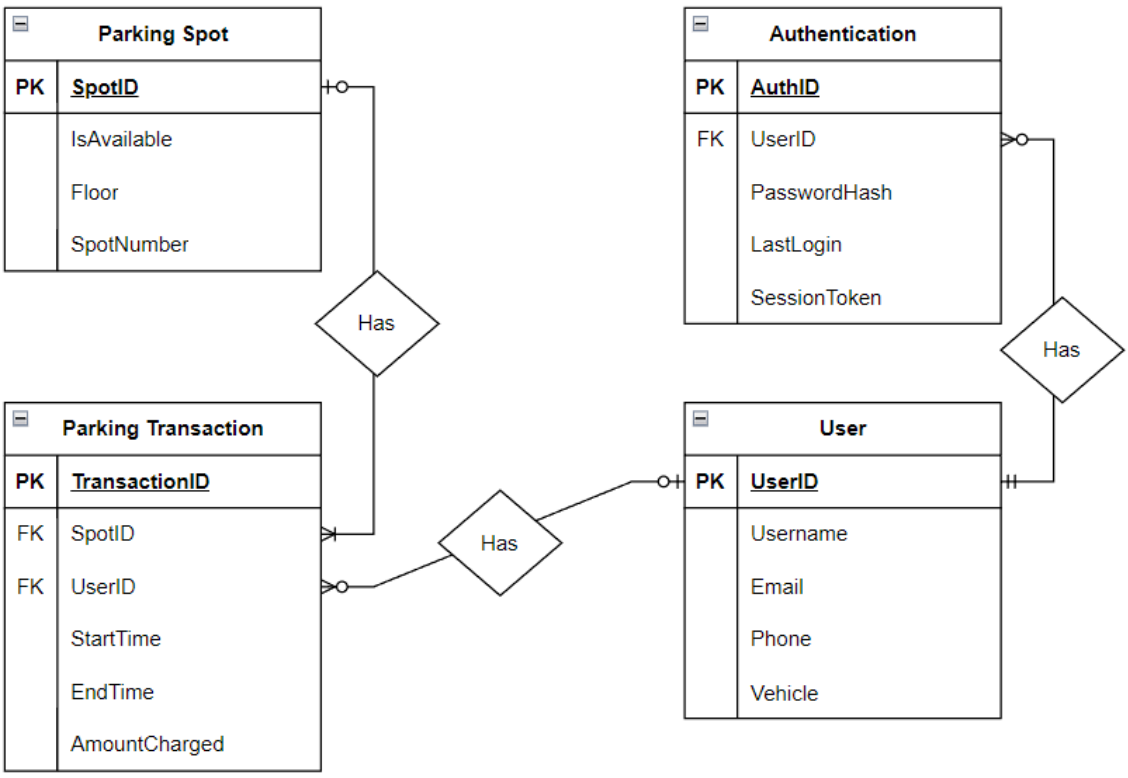


Figure 6-3: ERD for the Parking Management System Schema.

Repositories

Parking Spot Repository:

- SpotID: Unique identifier for each parking spot.

- Location: Descriptive location or coordinates of the parking spot.
- Status: Indicates whether the spot is occupied, reserved, or available.

Parking Transaction Repository:

- TransactionID: Unique identifier for each parking transaction.
- SpotID: Reference to the associated parking spot.
- UserID: Reference to the user who made the transaction.
- StartTime: Timestamp indicating when the parking started.
- EndTime: Timestamp indicating when the parking ended.
- Amount: The total amount charged for the parking.

User Repository:

- UserID: Unique identifier for each user.
- Username: User's chosen name for login.
- Email: User's email address.
- Phone: User's contact number.
- Vehicle: Information about the user's vehicle, such as color, make/model, and license plate number.

Authentication Repository:

- AuthID: Unique identifier for each authentication record.
- UserID: Reference to the associated user.
- PasswordHash: Encrypted password for security.
- LastLogin: Timestamp of the user's last login.
- SessionToken: Token generated for user sessions.

Persistent Data Storage on AWS:

The system will utilize Amazon RDS (Relational Database Service) to host the relational database. Amazon RDS provides a scalable and secure environment, ensuring high availability and data durability. Regular backups will be scheduled, and the database will be set up in a multi-AZ deployment for failover support. The Spring Boot application will connect to this RDS instance using JDBC and the necessary AWS SDKs.

Data Security and Integrity:

To ensure data security, all sensitive data, such as passwords, will be encrypted using industry-standard encryption algorithms. AWS RDS also provides built-in encryption at rest and in transit. Data integrity will be maintained using foreign key constraints and other relational database features.

7. Interaction Diagrams

User registration

The website is responsible for allowing the user to register an account with the SPGMS system. The website displays the appropriate fields which the user will have to enter to create an account. The system sends the information to the database which will verify and enter the information for the new user. The database will also verify if the user enters the correct information when they log in.

UC-1: User registration
Initiating Actor: Driver
Actor's Goal: Create an account
Participating Actors: Driver, Website, database
Preconditions: User enters information (first name, last name, username, email address, car make/model, payment method)
Postconditions: Account created
Flow of Events for Main Success Scenario:

→	1. Driver accesses the SPGMS Website.
→	2. Driver selects the option register for an account
←	3. Website displays a registration page.
→	4. Driver enters information.
→	5. Database entry is created with user information for a new account
←	6. Registration complete page is displayed for user

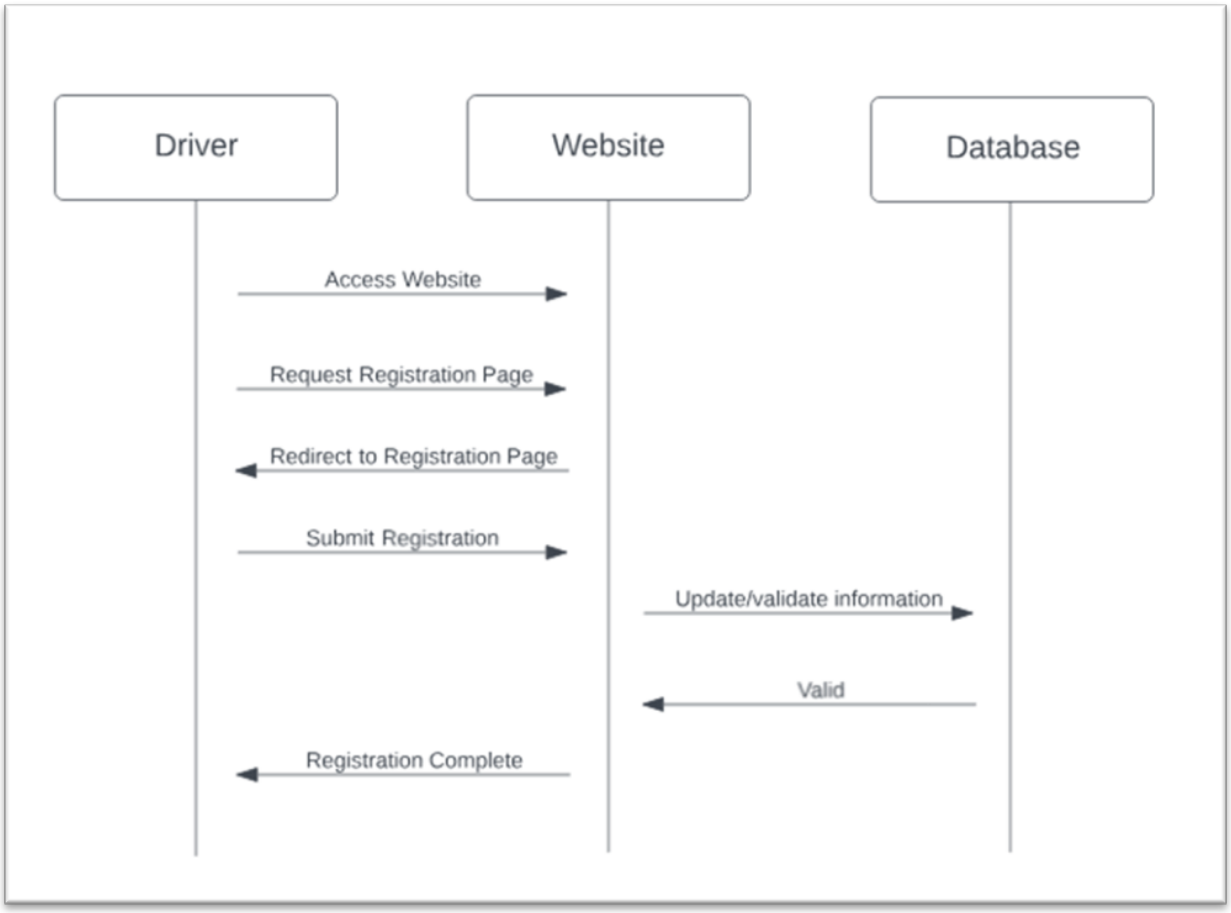


Figure 7-1 Interaction diagram of online registration

User login

The user accesses the SPGMS system by logging in using the credentials of an account that is existing in the database. When the user selects login, the system checks the database. If the database locates an account and the credentials are a match, it redirects the user to the user portal. If the credentials are not correct, it prompts the user to reenter the credentials and increase the login attempts. After 3 failed attempts, the account is locked.

UC-2: User login	
Initiating Actor: Driver	
Actor's Goal: Log into an existing account	
Participating Actors: Driver, Website, database	
Preconditions: Driver enters username and password	
Postconditions: Driver logs into the SPGMS system	
Flow of Events for Main Success Scenario:	
→	1. Driver accesses the SPGMS Website.
→	2. Driver enters username and password
←	3. SPGMS user portal is display if credentials are valid

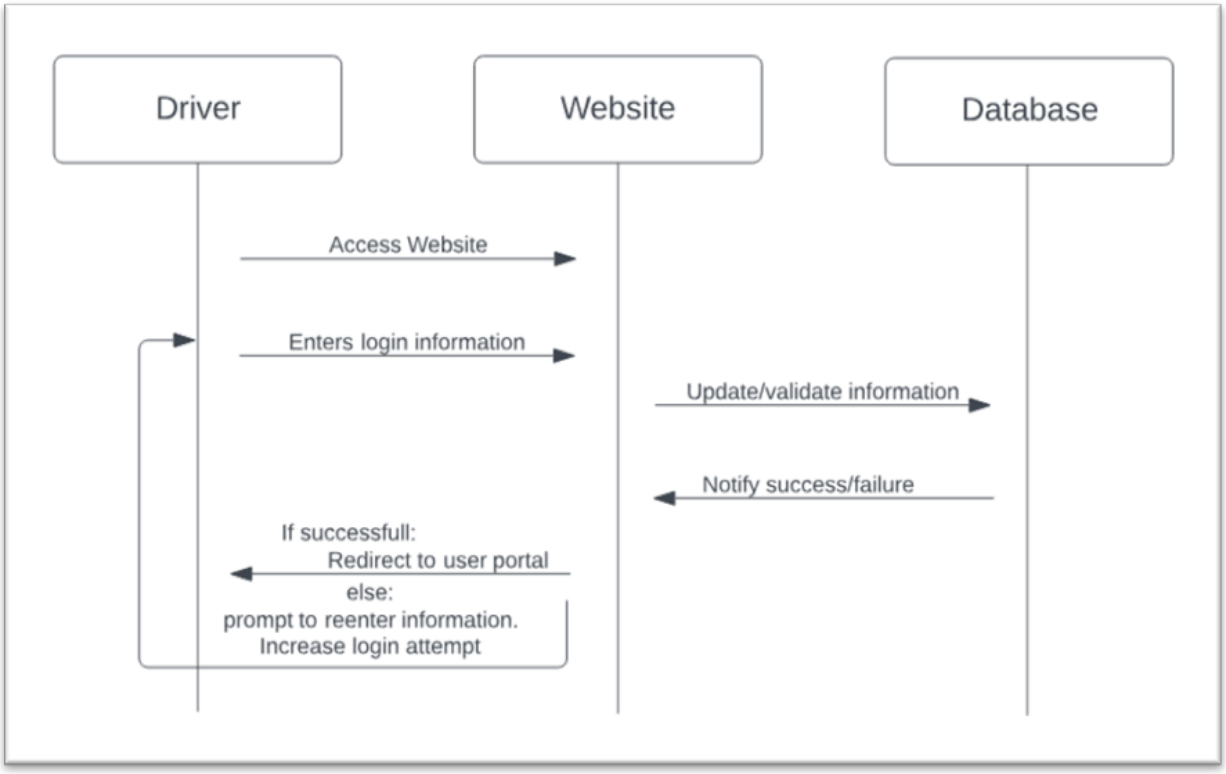


Figure 7-2 Interaction diagram of user login

Reservation

When a user is logged into the system, they have the ability to reserve a spot at a garage. The system displays a list of available parking spots for the user to select from based off of the time the user selects. The user selects the spot and the database will reserve the spot for the selected time. The system then displays the confirmation page for the reserved spot.

UC-3: Reservation
Initiating Actor: Driver
Actor's Goal: Reserve a spot to park
Participating Actors: Driver, Website, database

Preconditions: Driver logs into system and selects an available spot to reserve	
Postconditions: Spot reserved	
Flow of Events for Main Success Scenario:	
→	1. Driver accesses the SPGMS Website.
→	2. Driver enters username and password
←	3. SPGMS user portal is display if credentials are valid
←	4. Get available spot for reservations
→	5. Select spot to reserve.
→	6. Database marks spot as reserved
←	7. Notify driver of reservation

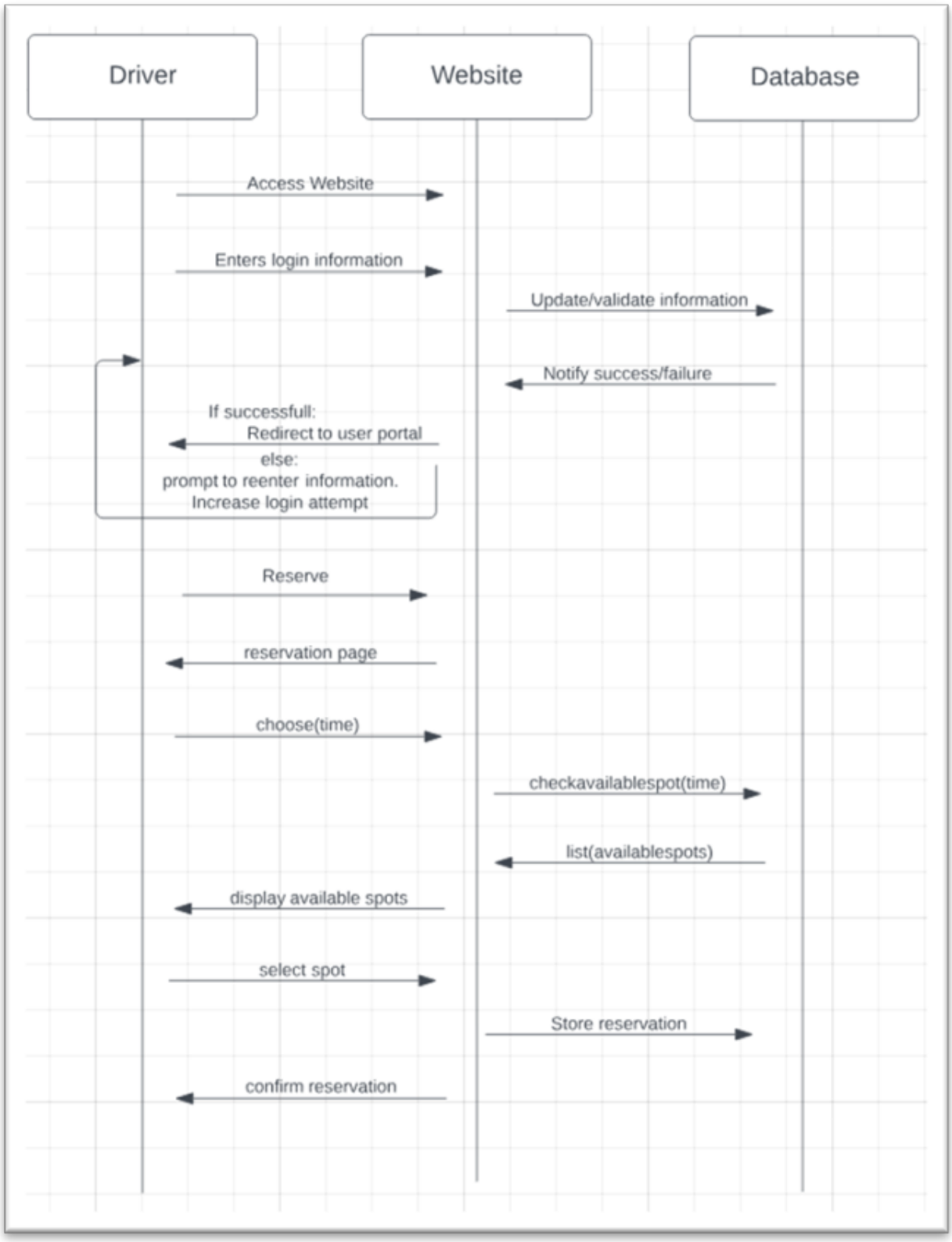


Figure 7-3 Interaction diagram of user reservation

8. Class Diagram and Interface Specification

a. Class Diagram

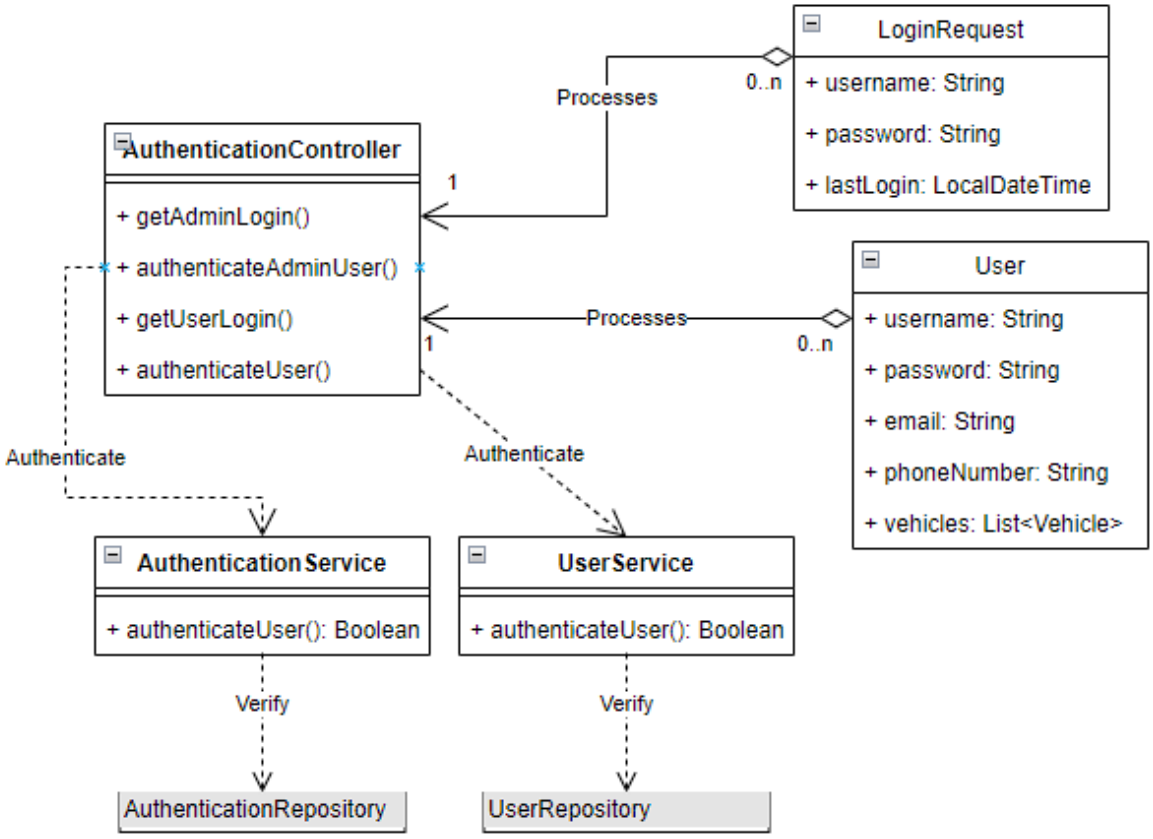


Figure 8-1 The AuthenticationController processes login requests from webpages, directs them to the appropriate service, and verifies the credentials

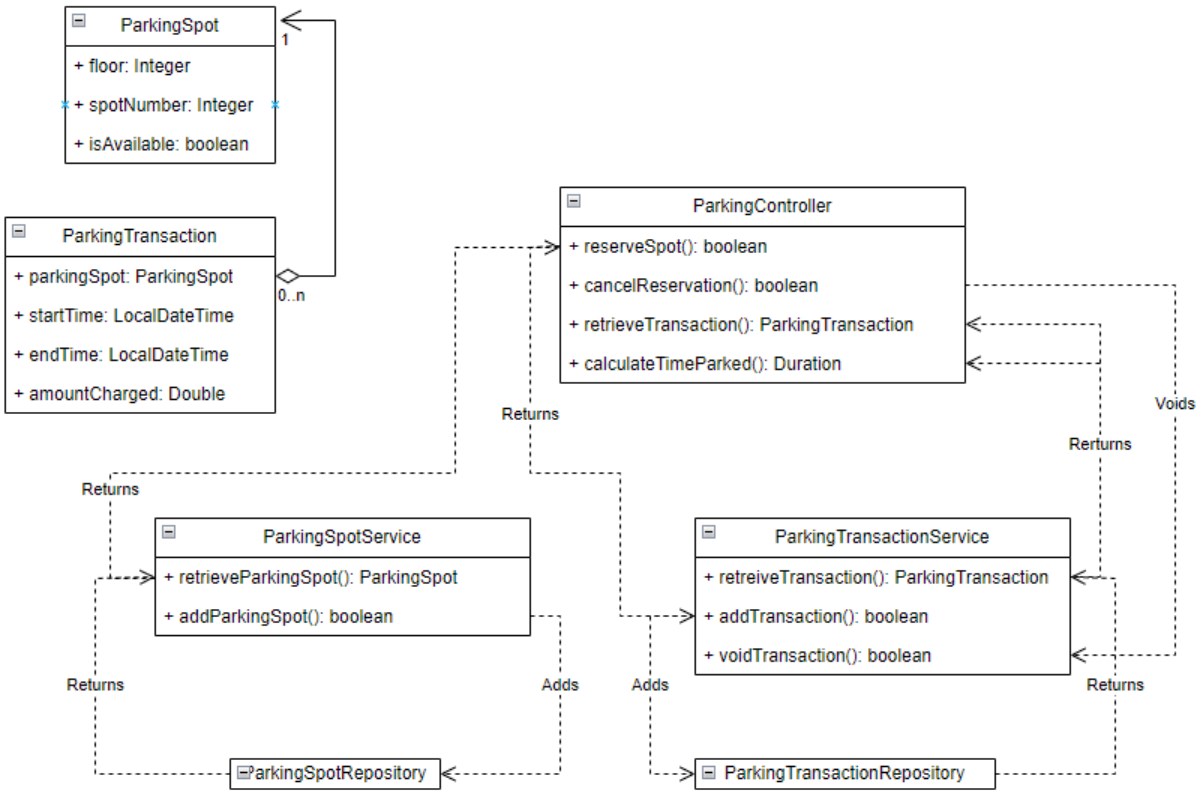


Figure 8-2 The ParkingController manages requests related to parking services, including creating and terminating reservations.

b. Data Types and Operation Signatures

This section contains a list of the system class specifications, followed by a plain language description of each class.

Class Specifications:

- AuthenticationController
  - Operations:
    - login(): void
    - logout(): void

- StaticResourcesController
  - Operations:

        - loadResource(): void

- DriverPortal
    - Attributes:
        - driverCredentialVerification: Boolean
        - driverDataEntry: String
        - vehicleDataEntry: String
    - Operations:
        - accessProfile(): void
        - updateProfile(): void
        - makeReservation(): void

- PaymentStore
    - Attributes:
        - securePayment: String
    - Operations:
        - storePaymentInfo(): void

- ReserveSchedule
    - Attributes:
        - dateSelection: Date
        - timeSelection: Time
        - recurringOption: Boolean
    - Operations:
        - scheduleReservation(): void

- SpaceAvailableCheck
    - Operations:
        - checkAvailability(): Boolean

- ReserveConfirm

    - Operations:
        - sendConfirmation(): void

- CheckInScreen
    - Attributes:
        - arrivalReservation: Boolean
        - arrivalWalkIn: Boolean
    - Operations:
        - displayOptions(): void
        - selectReservationCheckIn(): void
        - selectWalkInCheckIn(): void

- ReserveCheckIn
    - Attributes:
        - locateReservation: String
    - Operations:
        - retrieveReservation(): void

- WalkInCheckIn
    - Operations:
        - directToPaymentPortal(): void

- CheckOutScreen
    - Attributes:
        - calculateTimeParked: Time
        - CalculateCost: Float
    - Operations:
        - displayCheckoutOptions(): void
        - processPayment(): void

- PaymentProcess

    - Operations:
        - processExitPayment(): void

- OpenGate
    - Operations:
        - openEntranceGate(): void
        - openExitGate(): void

- VehicleCount
    - Attributes:
        - addAvailableSpace: Int
        - removeAvailableSpace: Int
    - Operations:
        - updateVehicleCount(): void

- SecurityPersonnel
    - Attributes:
        - username: String
        - password: String
    - Operations:
        - login(): void
        - verifyParking(): void
        - preventUnauthorizedParking(): void

- Database
    - Attributes:
        - reservationData: List<Reservation>
        - vehicleData: List<Vehicle>
    - Operations:
        - retrieveReservationData(): List<Reservation>
        - retrieveVehicleData(): List<Vehicle>

- Reservation
  - Attributes:
    - reservationID: String
    - driverID: String
    - spotID: String
    - reservationTime: Time
  - Operations:
    - createReservation(): void
    - cancelReservation(): void

- Vehicle
  - Attributes:
    - vehicleID: String
    - driverID: String
    - licensePlate: String
    - color: String
    - make: String
    - model: String
  - Operations:
    - registerVehicle(): void
    - updateVehicleInfo(): void

- LoginRequest
  - Attributes:
    - username: String
    - password: String

- ParkingSpot
  - Attributes:
    - spotID: String

        - availability: Boolean

- ParkingTransaction
    - Attributes:
        - transactionID: String
        - driverID: String
        - spotID: String
        - startTime: Time
        - endTime: Time

- User
    - Attributes:
        - userID: String
        - username: String
        - password: String

- AuthenticationService
    - Operations:
        - authenticateUser(): Boolean

- ParkingSpotRepository
    - Operations:
        - getParkingSpot(): ParkingSpot
        - updateParkingSpot(): void

- ParkingTransactionRepository
    - Operations:
        - getTransaction(): ParkingTransaction
        - createTransaction(): void

- UserRepository

    - Operations:
        - getUser(): User
        - updateUser(): void

- UserService
    - Operations:
        - manageUser(): void

- ParkingSpotService
    - Operations:
        - manageParkingSpot(): void

- ParkingTransactionService
    - Operations:
        - manageTransaction(): void

Definitions in Plain Language:

AuthenticationController: Manages user authentication, including login and logout functionalities.

StaticResourcesController: Handles the loading of static resources.

DriverPortal: A user interface for drivers to access and manage their profile, vehicle, and reservation details.

PaymentStore: Manages and securely stores payment information of drivers.

ReserveSchedule: Allows drivers to schedule parking reservations, including date, time, and recurrence options.

SpaceAvailableCheck: Checks the availability of parking spaces for a given reservation date and time.

ReserveConfirm: Sends a confirmation to the driver once a reservation is made.

CheckInScreen: The interface used by security personnel to check in drivers upon arrival, either with a reservation or as a walk-in.

ReserveCheckIn: Retrieves reservation details for drivers who have pre-booked.

WalkInCheckIn: Directs security personnel to the payment portal for drivers without a reservation.

CheckOutScreen: Interface used by security personnel to process driver checkouts, calculating parking duration and cost.

PaymentProcess: Manages the payment process based on parking duration.

OpenGate: Controls the opening of entrance and exit gates in the parking facility.

VehicleCount: Monitors and updates the number of vehicles in the parking facility.

SecurityPersonnel: Represents security staff who manage and monitor the parking facility, ensuring only authorized parking.

Database: Central storage for all reservation and vehicle data.

Reservation: Represents a parking reservation with details like ID, driver, parking spot, and reservation time.

Vehicle: Represents a vehicle with details like ID, driver, license plate, color, make, and model.

LoginRequest: Contains the credentials required for user authentication.

ParkingSpot: Represents a specific parking spot in the facility with its availability status.

ParkingTransaction: Represents a parking transaction, including details of the driver, parking spot, and duration.

User: Represents a registered user with their credentials.

AuthenticationRepository: Manages user authentication data and operations.

ParkingSpotRepository: Manages data and operations related to parking spots.

ParkingTransactionRepository: Manages data and operations related to parking transactions.

UserRepository: Manages user data and related operations.

UserService: Provides services related to user management.

ParkingSpotService: Provides services related to parking spot management.

ParkingTransactionService: Provides services related to parking transaction management.

c. Traceability Matrix

Traceability Matrix: Making a Reservation						
	Domain Concepts					
Class	DriverPortal	PaymentStore	ReservSched	SpaceAvailCheck	ReservConfirm	PaymentProcess
AuthenticationController	X	X				X
StaticResourcesController	X	X	X	X	X	X
DriverPortal	X					
PaymentStore		X				
ReserveSchedule			X			
SpaceAvailableCheck				X		
ReserveConfirm					X	
PaymentProcess						X
Database			X		X	
Reservation			X	X	X	
Vehicle			X		X	
LoginRequest	X					
User	X					
AuthenticationRepository	X					
ParkingSpotRepository				X		
ParkingTransactionRepository			X	X		
UserRepository	X					

Traceability Matrix: Admitting a Driver into the Garage								
	Domain Concepts							
Class	CheckInScreen	ReservCheckIn	WalkInCheckin	PaymentStore	CheckOutScreen	PaymentProcess	OpenGate	VehicleCount
AuthenticationController	X							
StaticResourcesController	X	X	X		X			
DriverPortal								
PaymentStore	X		X	X				
CheckInScreen	X							
ReserveCheckIn		X						
WalkInCheckIn			X					
CheckOutScreen					X			
PaymentProcess						X		
OpenGate							X	
VehicleCount								X
SecurityPersonnel	X							
Database				X		X		X
Reservation		X						
Vehicle		X						
LoginRequest								
ParkingSpot								
ParkingTransaction	X				X	X		
AuthenticationRepository	X							
ParkingSpotRepository								X
ParkingTransactionRepository								X

9. User Interface Design and Implementation

Home Page:

- Links:
  - Create a New Account
  - Customer Log In

Create a New Account (customer):

The Create a New Account page will contain a form to collect user data.

- Form Fields:
  - Username
  - Password
  - Email
  - Phone Number
  - Vehicle
    - License Plate
    - Make
    - Model
    - Color
- Buttons:
  - Submit
    - The Submit button will commit the data to the user's record in the Database.
  - Cancel
    - The Cancel button will discard any information in the fields and return to the home page.

Customer Log-In:

The Customer Log-In page will contain a form to accept log-in credentials (Figure 9-1).

- Form Fields:
  - Username
  - Password

- Buttons:
    - Submit
        - The Submit button will verify the credentials using the AuthenticationController.
    - Cancel
        - The Cancel button will discard any information in the form fields and return to the home page.

If the credentials are verified, the user will be directed to the Driver Panel page.

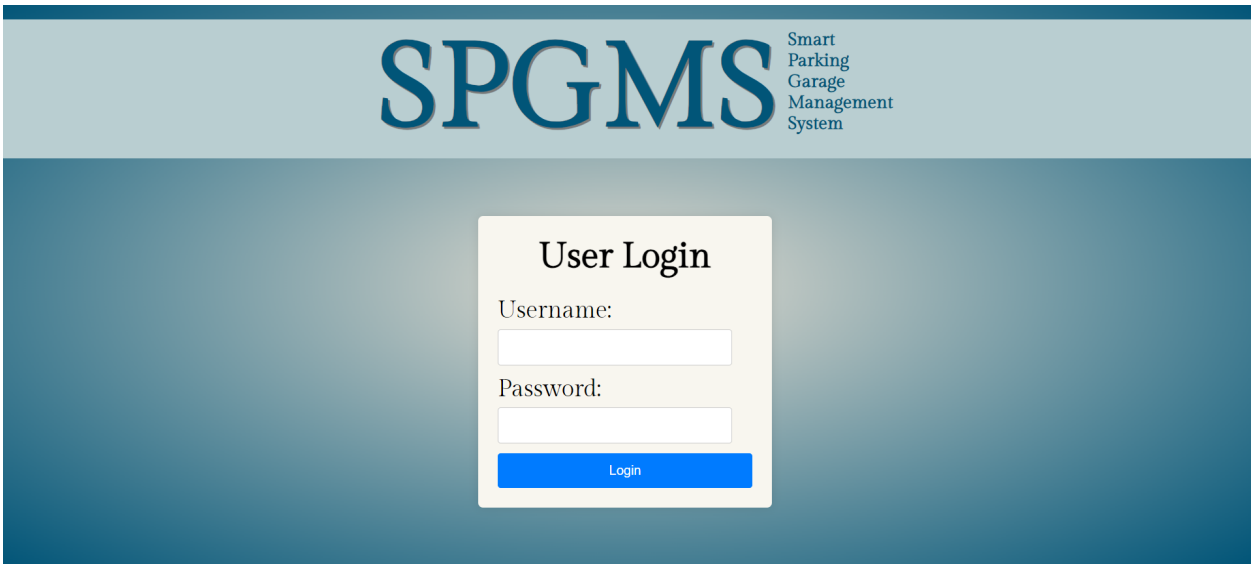


Figure 9-1: Customer Log-In Page

Create a New Account (Admin):

Administrative account information will not be accessible via the home page. Garage administrative personnel will have a link to access employee-only pages.

This page will contain a form to collect administrator data.

- Form Fields:
    - Username
    - Password
    - Name

    - Employee ID
- Buttons:
    - Submit
        - The Submit button will commit the data to the administrator's record in the Database.
    - Cancel
        - The Cancel button will discard any information in the fields and return to the home page.

Administrator Log-In:

The Administrator Log-In page will contain a form to accept log-in credentials (Figure 9-2).

- Form Fields:
    - Username
    - Password
- Buttons:
    - Submit
        - The Submit button will verify the credentials using the AuthenticationController.
    - Cancel
        - The Cancel button will discard any information in the form fields and return to the home page.

If the credentials are verified, the user will be directed to the Administrator Panel.

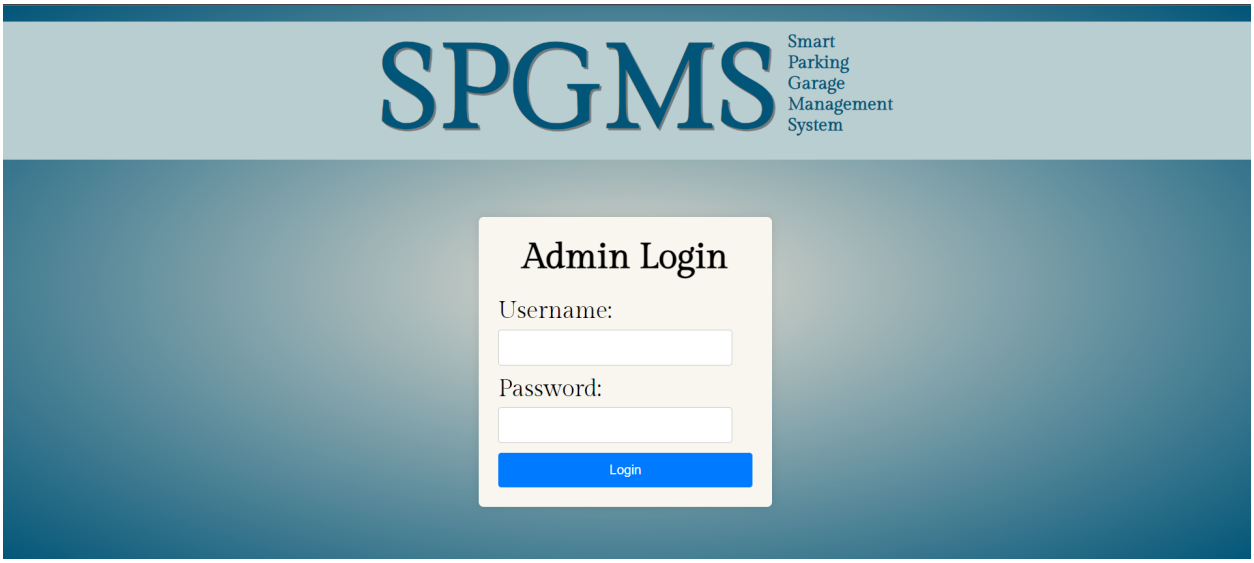


Figure 9-2: Admin Log-in Page

Driver Panel:

The Driver Panel will provide access to the majority of the system functionality available to the customer.

- Links:
    - Update Profile
    - Make a Reservation
    - Cancel a Reservation

Admin Panel:

The Admin Panel will provide access to the system functionality available to the administrator account. Garage personnel may require a different set of functions depending on their job description.

Toll Booth Attendant Check-In Page:

This functionality will be accessible by employees assigned to the garage entrance and exit stations. They will need the ability to check in customers, access and enter customer data and reservation information, and process payments. They will also require notification if the garage reaches capacity.

- Links:
    - Walk-in Customer Check-In
    - Reservation Check-In
    - Check-Out

Walk-In Check-In Page:

Walk-in customers will not be required to have an existing account or any previous data on file before they approach the garage entrance booth. Upon checking in with the toll-booth attendant, they will be required to provide payment information before parking. The payment information will be stored, and payment will be processed when the customer exits the garage.

This screen will require a form for the toll-booth attendant to enter the customer's name, vehicle, and payment information.

- Form Fields:
    - Customer Name
    - Customer Phone Number
    - Vehicle Information
        - License Plate
        - Make
        - Model
        - Color
    - Payment Information
- Buttons:
    - Submit
        - The Submit button will commit the user data to the Database and open the gate to the garage.
    - Cancel
        - The Cancel button will discard any information in the form fields and return to the Toll-Booth Attendant Check-In page.

Reservation Check-In Page:

Customers who have existing reservations will already have user data, vehicle information, and payment information on file. The toll-booth attendant will need to access a search function to locate the reservation.

Once the toll-booth attendant has verified that the user, vehicle, and payment information match the customer checking in, they will allow the customer access to the garage.

- Buttons:
    - Confirm Arrival
        - The Confirm Arrival button will change the reservation status to "arrived" and open the gate.
    - Cancel
        - The Cancel button will discard any changes and return to the Toll-Booth Attendant Check-In page.

Check-Out Page:

When a customer is exiting the garage with their vehicle, they will stop at the exit station. The cost of their transaction will be calculated and their payment processed.

The toll-booth attendant will need to access a search function to locate the transaction. They will select the transaction, which will calculate the cost and process the payment.

- Buttons:
    - Close Transaction
        - The Close Transaction button will change the status of the transaction to "closed" and open the gate to allow the customer to exit the garage.

Changes from the Original Design

Some changes were made since the completion of the User Interface Specification section of this report. The original design allowed a customer to select and reserve a specific parking space. This approach has been modified due to complications with integrating walk-in customer spaces with reservation customer spaces, and making sure that a customer doesn't take a reserved space by mistake. At this time, the plan is to allow for reservation of a general space in the garage. The system will monitor the count of vehicles in the garage and the number of spaces available for reservation and for immediate use, but will not reserve a specific space. This functionality may be updated further in future implementations.

10. Design of Tests

Backend:

Application Initialization

Application.java

- Test if the application initializes without any errors.
- Test if all required beans and dependencies are correctly injected.

Authentication

AuthenticationController.java

- Test if a user can successfully log in with valid credentials.
- Test if a user cannot log in with invalid credentials.
- Test if an admin can successfully log in with valid credentials.
- Test if an admin cannot log in with invalid credentials.

User Management

CustomUserService.java

- Test if a user's details can be successfully loaded by username.
- Test if a user can be authenticated using a login request.

Parking Spot Management

ParkingSpotService.java

ParkingSpotRepository.java

- Test if available parking spots can be retrieved.
- Test if parking spots can be filtered by floor.
- Test if parking spots can be filtered by availability on a specific floor.

Parking Transactions

ParkingTransactionService.java

ParkingTransactionRepository.java

- Test if transactions can be retrieved by parking spot.
- Test if transactions can be filtered by payment status.
- Test if transactions can be filtered by start and end time.

Data Loading

DataLoader.java

- Test if post-data is successfully loaded on context refresh.

Security Configuration

SecurityConfig.java

- Test if specific endpoints are accessible without authentication.
- Test if other endpoints require authentication.

Basic Functionality Test

ParkingGarageOptimizerApplicationTests.java

- Test if the application context loads correctly.

Frontend:

Creating an account

- Check if the user can create a new account.

Log into account

- Registered customers can login with the correct username and password.
- System returns error if credentials are wrong.

Making reservation

- Check if the user can make a reservation with select date/time, credit card and phone number.

User reservation status

- Check if users can see reservations.
- Check if users can edit/cancel reservations.

11. Project Management and Plan of Work

a. Merging Contributions of Team Members

The Team meets on a weekly basis to plan for the required report sections for the coming week. Each team member chooses a section or sections to focus on. The Team works from a shared document using Google Docs, which works well for compiling material from multiple contributors. Once all information has been added by each team member, Nicole reviews the report for formatting issues and updates the Table of Contents and Individual Contributions sections before submitting the report.

While this system of dividing up responsibilities worked well, there were occasions where a team member found that the section that they were assigned was out of the scope of their knowledge or experience. Questions were posed to other team members throughout the week via the messaging app, Discord. Team members were supportive of each other in the completion and review of their respective sections.

b. Project Coordination and Progress Report

The Team evaluated the use cases and divided them into “essential” and “additional” functionality for the system. Use cases that were deemed “essential” have been the focus of

Report 2 and Demo 1. “Additional” functionality use cases have been set aside for Report 3 and Demo 2.

Essential Functionality Use Cases

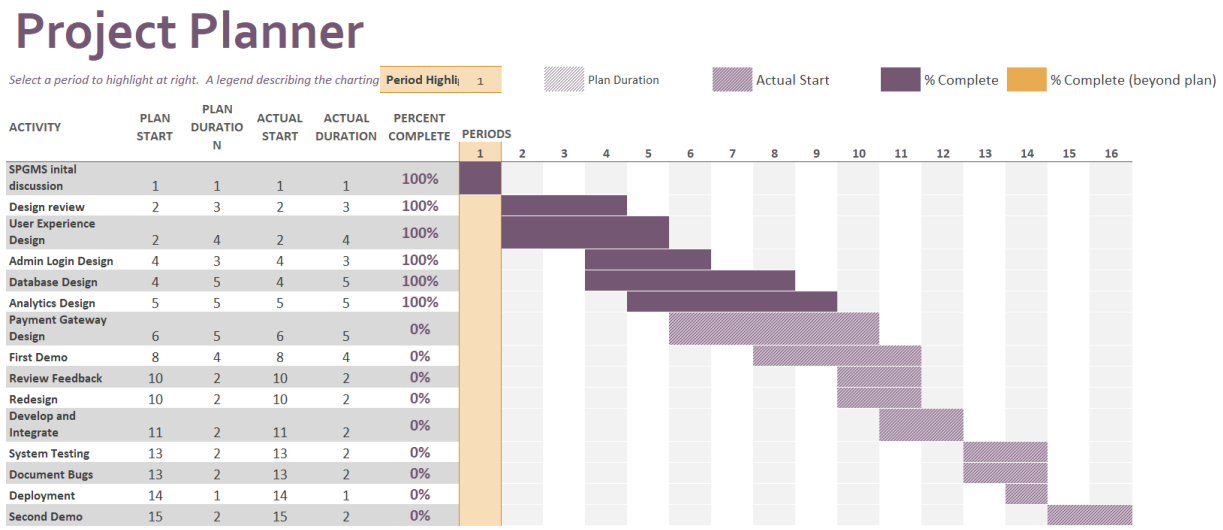
- Account creation and login
- Payment
- Walk-in customers
- Advance reservations
- Recurring reservations

Additional Functionality Use Cases

- Analytics
- Suggesting near-by attractions to drivers
- Business promotions

c. Plan of Work

Our team plan of work is outlined in the Gantt chart in Figure 11-1. The periods in the chart represent weeks in the Fall 2023 semester. The chart will be updated accordingly as the project progresses.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
SPGMS inital discussion	1	1	1	1	100%
Design review	2	3	2	3	100%
User Experience Design	2	4	2	4	100%
Admin Login Design	4	3	4	3	100%
Database Design	4	5	4	5	100%
Analytics Design	5	5	5	5	100%
Payment Gateway Design	6	5	6	5	0%
First Demo	8	4	8	4	0%
Review Feedback	10	2	10	2	0%
Redesign	10	2	10	2	0%
Develop and Integrate	11	2	11	2	0%
System Testing	13	2	13	2	0%
Document Bugs	13	2	13	2	0%
Deployment	14	1	14	1	0%
Second Demo	15	2	15	2	0%

Figure 11-1: Gantt Chart

d. <u>Breakdown of Development Responsibilities</u>

Team coding responsibilities and status updates are tracked using a Trello Board. Progress thus far is outlined in the table below.

Nicole Brandenburg	● Team meeting organization ● Report formatting ● Front-end coding ○ Login Pages (User and Admin) ○ Admin Panel
Gene Holt	● Front-end coding
Oakley Cardwell	● All back-end coding ○ Database development and integration ○ Server side security configuration ○ MVC architecture for endpoint management ● Trello board updates

References

- Marsic, I. (2012). *Software Engineering* [online]. New Brunswick: Rutgers University. Available from: https://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf [accessed 28 Aug 2023].
- Sharma, A., Kumar, M., Agarwal, S. (2015). 'A Complete Survey on Software Architectural Styles and Patterns', *Procedia Computer Science*, vol. 70, pp. 16-28 [online]. Available from: https://www.sciencedirect.com/science/article/pii/S187705091503183X [Accessed 24 Sep 2023].
- Tran, L., Nguyen K., Choudhury S., Ngo T., Nguyen, D., Xiao, Z., Patel, N. (2019). *Blockchain and Docker Assisted Secure Automated Parking Garage System* [online]. Available from: https://www.ece.rutgers.edu/~marsic/books/SE/projects/ParkingLot/ [Accessed 8 Sep 2023].

Modifications and Improvements

October 16, 2023

Per the instructor's review of Report 2, Part 2, descriptions were elaborated upon for the class diagram and interaction diagrams.

October 9, 2023

Per the instructor's review of Report 2, Part 1, the ERD for the Parking Management System Schema was added to the Data Model and Persistent Data Storage section.

October 2, 2023

"Website" and "Database" were added as separate Actors in the Actors and Goals section.

"UC-4: Daily Commuter Reserves a Parking Spot" was added to the Fully-Dressed Descriptions and System Sequence Diagrams section.

September 25, 2023

Per the instructor's review of Report 1, Part 2, the following modifications were made prior to the submission of the Full Report 1:

- The Use Case Diagram was broken into two separate diagrams, each with a single boundary line.
- The Traceability Matrix was re-formatted to display use cases, requirements, and priority weights in a single, tabular format.