

SMART PARK

“RU Ready to Park Smart?”

Report 1



Github: <https://github.com/swetha-5689/parking-manager>

Group 3

Disha Bailoor, Neha Nelson, Param Patel, Swetha Angara,
Nicholas Meegan, Thomas Murphy, Charles Owen, Jeffrey Samson,
Aniqa Rahim, Brian Ogbebor

Table of Contents

Table of Contents	2
Individual Contribution Breakdown	4
Customer Problem Statement	4
Glossary of Terms	8
System Requirements by Subgroups	10
Customer Registration Subgroup	10
Functional Requirements	10
Non-Functional Requirements	11
User Interface Requirements	11
Managerial / Administration Subgroup	14
Functional Requirements	14
Non-Functional Requirements	14
User Interface Requirements	15
Elevator Operation Subgroup	18
Functional Requirements	18
Non-Functional Requirements	18
User Interface Requirements	19
Functional Requirement Breakdown	22
Stakeholders	22
Users of the Parking Garage	22
Managers of the Parking Garage	22
Actors and Goals	23
Use Cases	25
Casual Description	25
Use Case Diagram	28
Traceability Matrix	29
Fully Dressed Descriptions	30
System Sequence Diagrams	39
Customer Subgroup	39
Managerial / Administrative Subgroup	40
Elevator Operation Subgroup	43
User Interface Specification Breakdown	46
Customer Registration Subgroup	46

Estimated User Effort	48
Managerial / Administrative Subgroup	49
Estimated User Effort	53
Elevator Operation Subgroup	54
Estimated Effort by User	58
Domain Analysis	59
Domain Model	59
Concept Definitions	60
Association Definitions	62
Attribute Definitions	64
Traceability Matrix	65
System Operation Contracts	66
Mathematical Models	69
Poisson Random Process	69
Dynamic Pricing Model	69
Project Size Estimation	71
Project Management by Subgroups	79
Customer Registration Subgroup	79
Managerial / Administrative Subgroup	80
Elevator Operation Subgroup	81
Plan of Work	83
Customer Registration Group	83
Managerial/Administrative Group	84
Elevator Operation Group	85
Product Ownership (from Proposal)	87
References	91

Individual Contribution Breakdown

After consulting the entire group, we have decided that all of the team members have contributed equally to this report.

Customer Problem Statement

Problem Statement

Numerous processes in our daily lives are being digitized at a rapid pace; our everyday experiences continue to be dominated by inefficiencies that can be remedied by modernization. One such experience is finding parking in an urban area. The process of parking or seeking parking is responsible for overflow traffic in streets surrounding the parking garage, wasted staff-hours, and time-consuming commutes. Factors like these make the automation of parking facilities a prime target for the development of efficient digital systems. Automated parking benefits garage owners and customers by optimizing time due to assigned parking spots, allowing advance reservations, avoiding congestion in the garage itself and making a more streamlined parking experience. A computerized garage facility requires fewer employees to achieve business optimization while giving customers an improved experience.

Parking garage construction and usage are rising in step with urbanization throughout the US. Despite this increase in growth, parking garage revenue is expected to be sluggish due to high-interest rates and investor uncertainty as well as consumer sentiments about parking garages. According to IBISWorld, the anticipated parking garage industry revenue will reach 11.3 billion dollars by 2024, only a 0.6% increase in revenue from 2019. The strategy to combat sluggish revenue growth is to not only increase parking garage revenue but to improve user experience by making parking garages more efficient by automating garage processes. Ultimately, the automation of parking garages will lead to less overflow traffic and by extension less time searching for parking spots. This will lead to higher efficiency for parking garage owners by streamlining spot assignments and reducing staff workload and parking garage customers by wasting less time searching for a parking spot and decreasing the pedestrian risk of injury.

In the traditional parking garage paradigm, customers enter the garage and receive a ticket, which will be used once they are finished parking to pay for usage of the spot. The customer then drives through the garage searching for a spot. What if each spot is occupied? The customer is then forced to drive back through the garage, either once again searching for a spot or exiting the garage to look for a spot elsewhere, costing the customer their own time and effort and the garage loses a potential customer. If multiple customers attempt this, it is possible that not only traffic will build up inside the garage itself but into the streets of the city itself, which further

inconveniences customers that are on a tight schedule. Depending on the time of day and day of the week, the percentage of total motorist traffic searching for a parking space in major cities can reach up to 45%, according to a study taken in 2007. The study was conducted at four different times: weekend peak, weekday peak, and two average weekday times (Barter, Reinventing Parking). By creating an automated parking garage, traffic in cities can be significantly decreased as less time is spent searching for a parking spot, freeing up nearly 50% of traffic in cities during busy periods.

As of now, parking garage staff is required to manually check the occupancy status of parking spots throughout the garage. This is accomplished by walking through the garage and making note of what spots are currently being taken up. As customers can leave the garage at any time, this means that either a parking garage staff may miss a departure while still considering that spot as occupied, or the garage must have staff watching and analyzing each parking spot for departures. If there were staff members watching each spot and marking if the spot is currently occupied or not, we would require long staffing hours or a significant amount of staff members per garage. We wish to automate our parking garage process to eliminate excessive employee overhead and increase efficiency of garage operations.

Another concern is liability related to pedestrian collisions that happen in the garage. In traditional parking garages, there is often a shortage of space due to two lanes of traffic, one going upwards and one going downwards. We wish to create a one-way flow of traffic in our garage by implementing an elevator system and assigned parking spots. The elevator will be the only access to upper levels, and each car will have a designated parking spot. This new system will reduce garage traffic and minimize liability associated with vehicles and humans occupying the same space.

Additionally, our parking garage management system does not account for relationship marketing. This is because all the tickets are taken at the front and then the customers park and go on about their lives. There is no way to track recurring customers or reserving parking spots for the customers in advance. We want our new solution to include customer data tracking and usage statistics. From this data, we would like to derive garage volume on specific dates/times and be able to make seasonal or otherwise targeted marketing efforts.

We want our garage system to interface with an easy-to-use mobile app and website. The ideal solution for us is one that allows customers to access their account details and make/alter reservations on their own time, from anywhere in the world. The system will ideally include a feature to gather customer feedback. System administrators should be able to aggregate this data and make actionable changes to the garage's operations to improve customer satisfaction. This feedback can give us

more insight into our shortcomings and help us improve our garage for the consumer, making a more satisfying customer experience.

In order to further improve customer interaction with the proposed system, we would like to have the ability for the customer to use voice recognition to perform most tasks of the software hands-free. This will allow customers who are driving to the garage to safely make a reservation, without needing to take their eyes off of the road in the event another passenger is not present in the vehicle. Additionally, we also wish to offer the same voice recognition software in the elevator. Instead of the user having to reach out of their vehicle's window to enter in their user information in the event their license plate is not recognized or scan a QR code to complete the interaction on their phone, the customer will also have the option to enter in their information through voice.

Presently, our pricing model is static. Meaning we have one price for every minute of every day. As mentioned above, the long term expectation for revenue growth in our industry is flat, yet demand is rising. As parking garage operators, we must develop tactics to combat this stagnant trend. We envision our system, including a dynamic pricing model that evaluates availability based on the number of spots already reserved when the customer checks the availability for a given day. Fewer available places mean higher demand, and the price should adjust accordingly.

Another requirement for our system is that we address a long standing lack of inclusivity for our disabled patrons. Currently, a disabled customer requiring a handicapped parking space has only a few options available on each floor of the garage and has a random chance of finding one convenient to their needs. The new system should inquire whether their car is authorized to park in a handicapped spot and find the closest available.

Finally, we are a future-minded business. There is always the possibility that we expand operations in the future by constructing additional spaces for our current garage or by acquiring another garage facility. With this in mind, we would like the new management system to be flexible. The parking floor plan should be able to add or remove parking spots and floor configurations dynamically. Further, the system should allow for the addition and removal of administrators and inclusion appropriate privilege levels for different employees.

We would like a solution to address as many of the above concerns as possible. Starting from the traffic overflow into the streets by creating a one way direction to guide the customer to their spot on their specific floor. Our customer does not waste any time in searching the parking garage for an open spot or leaving the garage to find parking elsewhere. This elevator system also addresses the problem of blind spots in the parking garages at turning corners, the elevator would lift the customer's car and ramps from the upper levels would assist in getting back to street level. Not only this, we would like our solution to be efficient and low-latency. Given the amount of users potentially

using the application at once, a speedy web service is necessary. This will ensure that the proposed system is faster than having employees walk around the garage to check each of the spots, as well as reduce chances of human error on our employees' part. Overall, the digitalization of a traditional parking garage could drastically improve the commuting experience, quality of life in urban areas and provide an insight into the most profitable business strategy for the company.

Glossary of Terms

This is a list of frequently used terms throughout this report:

Administrator - The administrator is any authorized user of the privileged portions of the system including: Customer data, billing/payments, parking garage rules.

Alexa Skill - Alexa is Amazon's Voice service. Skills are voice-driven Alexa capabilities. Adding Alexa skills brings products/services to life.

Central Server (Database) - The database which stores login information, customer vehicles, information regarding the parking garage and the garage layout.

Confirmed Reservations - Represent registered customers who have made a reservation.

Contracted Customers - Customers/businesses that make a long-term reservation with the garage for a certain number of spaces on a daily, weekly, monthly or annual basis.

Customer - The customer is any user accessing either the online reservation system or entering the parking garage in a vehicle.

Customer Reservation System - Database which stores data regarding customer's reservations.

Dynamic Pricing - Flexible prices for products based on current market demands. Depending on time of day, day of the week, and whether the current day is a holiday or event causes the price to change for the spot.

Elevator/Lift - A compartment used to raise and lower vehicles to different levels of the parking garage. The elevator will also be responsible for taking in user information in the event a license plate is not recognized.

Elevator Console/Terminal - Keypad in the elevator that allows the customer to edit the end time of their reservation or enter their membership number (see below).

Garage Simulation - Accessed through the management portal, the garage simulation is a mock-up of what the SmartPark system will be able to accomplish.

Grace Period - Also known as the holding period, a short amount of time before and after which the customer is allowed to come to the garage or leave the garage.

Ground Level - The bottom-most level of the garage (at street level). Cars will not be parked here, but will rather be used for walk-in customers to make reservations.

Guaranteed Reservations - Allows customers to make a monthly contract with the parking garage for a parking spot or multiple parking spots.

License Plate Reader/Recognition - A device that can read and store license plate number data to keep track of vehicles in the garage. Specifically, the device will check in arriving vehicles and check out departing vehicles.

Management Portal - The location where the garage employee can login and view garage statistics, set dynamic pricing, and change the garage layout.

Membership Number - Unique identifier for each customer that can help find their reservation in case of a misread of license plate.

Monthly Tab / Running Tab - The form of billing that accounts for all transactions (charges, refunds, etc.) over the course of a month with only one net payment is made after the completion of the month.

Occupancy - The maximum number of objects that can fit in a designated area. Max occupancy will occur when all of the parking spots are filled.

Occupied Spots - Spots that have already been reserved by customers or are currently in use.

Overbooking - Accepting more reservations than there is room to account for 'no-show' customers.

Overstay - Currently parked customers who wish to extend their stay beyond the time for which they made reservations, either intentionally or unintentionally.

Passenger Vehicle - A regular-use vehicle that is allowed to park in the garage [no trucks, motorbikes, trailers, etc. are allowed to park].

Premium/Handicapped Spot - Parking spots that a customer can access whether they pay a premium fee if interested in the premium spot or spots for customers with disabilities for handicapped spots.

QR Code - A machine-readable code consisting of an array of black and white squares, used for storing URLs or other information for reading by cameras. The user will be able to scan a QR code in the elevator console in order to enter the information through their phone instead.

Rain Check - A token for customers that arrive with a reservations but the parking garage is full. This will allow the customer to receive the same service they purchased but at a later date.

Registration Plate Number - License plate number; this is what the user will enter when signing up for the Smart Park service. The number will then be entered into the database and will then be recognizable by the license plate sensors.

Registration Time - The times the customer inputs as a boundary to their reservation.

Reservation Confirmation Number - The number a customer receives after they have made a reservation.

Reserved Period - The total time a parking spot is reserved for by a customer.

Sensor - A device that detects or measures a physical property and records, indicates or otherwise responds to it. These sensors will determine if spots are vacant or occupied and update the database accordingly.

Standing Reservations - Recurring reservation from known customer.

Understay - Are customers who arrive on time but decide to leave before their predicted time of departure.

Vacant/Unreserved Spots - Spots that are available to be reserved.

Voice Recognition - It is a computer software program or hardware device with the ability to decode the human voice. This will be used to allow customers to enter their information hands free.

Walk-In - Are registered customers who arrive at the parking garage without a contract or reservation or unregistered customers that can have a reservation be made for them.

System Requirements by Subgroups

Listed below are all the requirements of our proposed solution defined by subgroup and type:

Customer Registration Subgroup

1. Functional Requirements

Requirement Identifier	Size	Requirement
REQ: C-F-1	5 pts.	As a user, I should be able to create an account to store my demographic information for system operation.
REQ: C-F-2	5 pts.	As a user, I should be able to access my account to edit the information provided to ensure that the system has the most up-to-date information.
REQ: C-F-3	5 pts.	As a customer, I should be able to make a reservation.
REQ: C-F-4	5 pts.	As a customer, I should be able to access my previously made reservations and update them for changes in time of reservation, changing the car that will be parking in the garage, etc.
REQ: C-F-5	4 pts.	As a contracted customer, I should be able to set up a weekly, monthly or annual contract for my business' customers.
REQ: C-F-6	3 pts.	As the system owners, we should be able to charge the customer on a monthly basis in a 'running tab' format that can be paid by check or online after the month is complete.
REQ: C-F-7	4 pts.	As a user, I should be able to access this monthly payment information, to view and pay it after the completion of the month.
REQ: C-F-8	3 pts.	As a user, I should be able to pay for an extended grace period while I book my reservation.
REQ: C-F-9	4 pts.	As the system owners, we can charge the customer if they use the half an hour extension to their reservation in their monthly charge.
REQ: C-F-10	2 pts.	As system owners, we can charge the customer for their overstay time at an increasing rate, as informed by the elevator operation subgroup.

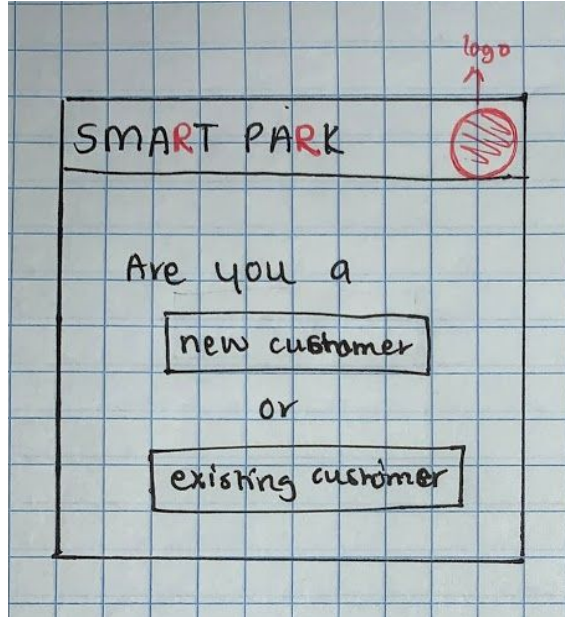
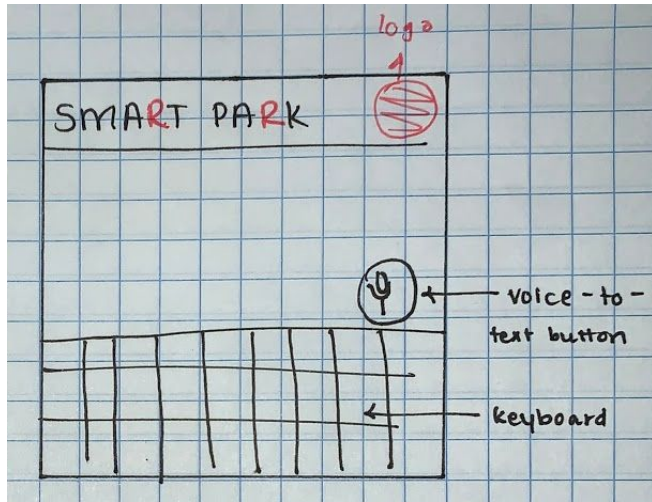
REQ: C-F-11	1 pt.	As system owners, we can coordinate with the managerial team to send out notifications, for any of the following reasons: overstay charges, subsection full, reservation is ending, grace period ending, etc.
REQ: C-F-12	1 pt.	As a user, I should be able to use voice recognition in order to set up a reservation or perform various other actions.

2. Non-Functional Requirements

Requirement Identifier	Size	Requirement
REQ: C-NF-13	5 pts.	As a system, we can check the customer's reservations at the time of booking to ensure that there are no contiguous bookings under their name.
		As a system, we can check every time a user extends his/her reservation that they do not run into a continuous block of bookings after that.
		As a system, every time a new reservation is made check to see that the user making the reservation does not have more than 3 active reservations.
REQ: C-NF-14	2 pts.	As the system owners, we can have the customers, agree to the 'terms and conditions' of our business policies
REQ: C-NF-15	3 pts.	As system owners, we can bill the customer in the event of a 'no-show' to their reservation.
REQ: C-NF-16	5 pts.	As a user, I want the rights to a 'rain-check' credit on my account, if the garage is unable to find a spot for my reservation.
REQ: C-NF-17	3 pts.	As a system, we want to be able to gain customer metric data per month to optimize our overbooking estimation algorithm.

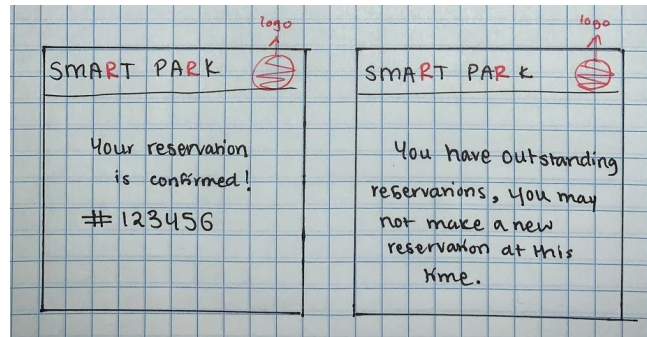
3. User Interface Requirements

Requirement Identifier	Size	Requirement
REQ: C-UI-18	5 pts.	As a customer, I should be able to use the user interface easily to register and make reservations.

		<p>As a customer, the number of data entries I have to enter to register and make a reservation should be minimal on my mobile device.</p> 
REQ: C-UI-19	2 pts.	<p>As a system operator, automatic spot selection will simplify the user interface significantly making it easy to develop and use, provided by elevator operation subgroup.</p>
REQ: C-UI-20	4 pts.	<p>As a customer, I should be able to make reservations using my voice, while driving or in a moving vehicle.</p> 
REQ: C-UI-21	5 pts.	<p>As a system operator, we should let the customer make a reservation and edit the times of their</p>

reservations on the mobile interface with the least amount of keystrokes/clicks for simplicity.

As a system operator, we would let the customer perform sensitive tasks such as editing the account information, making a temporary registration association and accessing their ongoing tab and previous bills on the larger desktop interface.



Managerial / Administration Subgroup

1. Functional Requirements

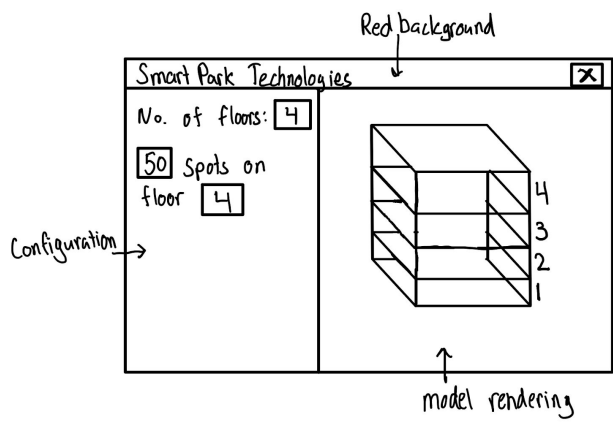
Requirement Identifier	Size	Requirement
REQ: M-F-1	5 pts.	As a parking garage administrator, I will be able to view statistics on historic garage occupancy, revenue, overbookings, overstays, and understays.
REQ: M-F-2	1 pt.	As a parking garage administrator, I will be able to view information on local events that may affect demand.
REQ: M-F-3	5 pts.	As a parking garage administrator, I will be able to change prices for reserved parking as well as fees for overstays.
REQ: M-F-4	3 pts.	As a parking garage administrator, I will be able to change the configuration of the garage; this includes changing the total number of spots, number of decks, and the number of spots per deck.
REQ: M-F-5	2 pts.	As a parking garage administrator, I will be able to configure the average rates of arrivals and departures.
REQ: M-F-6	3 pts.	As a parking garage administrator, I will be able to see customer profiles and print transactions by this customer.
REQ: M-F-7	5 pts.	As a parking garage administrator, I will be able to register to login to my account.
REQ: M-F-8	5 pts.	As a parking garage administrator, I will be able to register and logout of my account.
REQ: M-F-9	4 pts.	As a parking garage administrator, I will be able to view and approve system generated rainchecks.
REQ: M-F-10	5 pts.	As a parking garage administrator, I will be able to view a monthly record of transactions.

2. Non-Functional Requirements

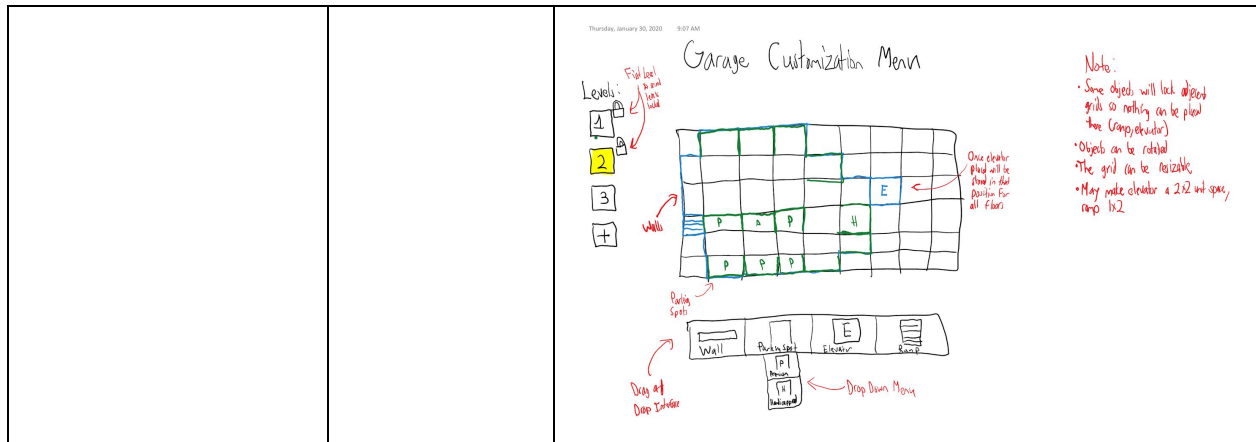
Requirement Identifier	Size	Requirement
------------------------	------	-------------

REQ: M-NF-11	5 pts.	As a parking garage administrator, I will have a login page that verifies my credentials.
REQ: M-NF-12	4 pts.	As a parking garage administrator, I will be able to access a separate interface where I can manage pricing and garage configuration.
REQ: M-NF-13	5 pts.	As a parking garage administrator, I will be able to search and select customer profiles.
REQ: M-NF-14	4 pts.	As a parking garage administrator, I will be able to select different parameters for viewing statistical information.
REQ: M-NF-15	4 pts.	As a parking garage administrator, I will be able to view a monthly summary of the garage's monetary business transactions.

3. User Interface Requirements

Requirement Identifier	Size	Requirement
REQ: M-UI-16	4 pts.	<p>As a parking administrator, I will be able to see a visual representation of the garage for configuration.</p> 
REQ: M-UI-17	4 pts.	As a parking administrator, I will have a dashboard to configure prices and manage or check on customer reservations.

		<div>Smart Park Technologies</div> <div>Reservation Price: 15\$/hr</div> <div>Walk-in Price: 20\$/hr</div> <div>Reservations Made Today: 120 Spots</div>																																
REQ: M-UI-18	5 pts.	<p>As a parking administrator, I will be able to view a visual representation of garage occupancy statistics.</p> <div><div>Smart Park Technologies</div><div><div>Pop-Up Window</div><div><div>Red Background</div><div>Exit button</div><div><div># of Spots Occupied</div><div>Hour</div><div>Graph of Chosen statistic</div><div><div>Week</div><div>Month</div><div>Year</div></div><div>Graph by Chosen measurement of time</div></div></div></div></div>																																
REQ: M-UI-19	4 pts.	<p>As a parking administrator, I will be able to edit system-generated monthly summaries of transactions.</p> <table><tr><th colspan="4">Smart Park Technologies</th></tr><tr><th>Customer Name</th><th>Reservation</th><th>Date</th><th>Price</th></tr><tr><td>Josh</td><td>no</td><td>1/21</td><td>\$15</td></tr><tr><td>Matt</td><td>yes</td><td>12/31</td><td>\$25</td></tr><tr><td>:</td><td>:</td><td>:</td><td>:</td></tr><tr><td>:</td><td>:</td><td>:</td><td>:</td></tr><tr><td>:</td><td>:</td><td>:</td><td>:</td></tr><tr><td>:</td><td>:</td><td>:</td><td>:</td></tr></table>	Smart Park Technologies				Customer Name	Reservation	Date	Price	Josh	no	1/21	\$15	Matt	yes	12/31	\$25	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
Smart Park Technologies																																		
Customer Name	Reservation	Date	Price																															
Josh	no	1/21	\$15																															
Matt	yes	12/31	\$25																															
:	:	:	:																															
:	:	:	:																															
:	:	:	:																															
:	:	:	:																															
REQ: M-UI-20	5 pts.	<p>As a parking administrator, I will be able to design and freely edit the layout of the parking garage.</p>																																



Elevator Operation Subgroup

1. Functional Requirements

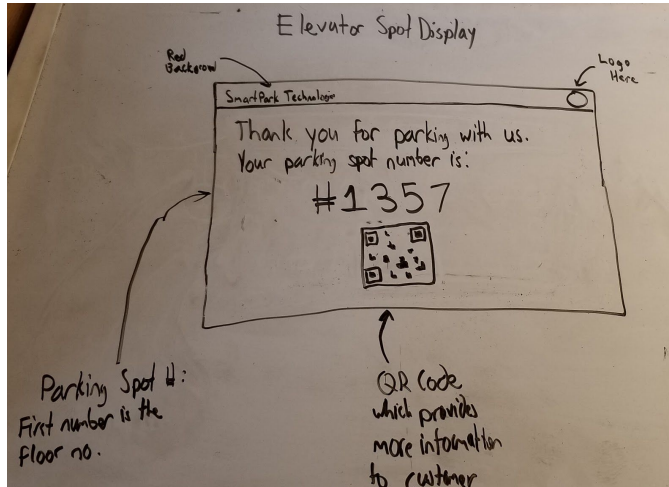
Requirement Identifier	Size	Requirement
REQ: E-F-1	5 pts.	The system will keep track of occupied and available spots.
REQ: E-F-2	5 pts.	The system will scan incoming license plates and reference the customer/reservation database for the next action.
REQ: E-F-3	5 pts.	The system will be able to obtain user membership info from the customer.
REQ: E-F-4	3 pts.	The system will alert customers if the garage is full and issue a rain-check notification to those who had reservations.
REQ: E-F-5	4 pts.	The system will ask users to register on the spot if they have not already done so.
REQ: E-F-6	5 pts.	The system will ask customers if they would like to extend their reserved period if they arrive after their grace period is up and there are vacant spots.
REQ: E-F-7	2 pts.	As the system operator I will allow customers to select a vacant spot of their liking in a given section of the garage as long as it is available.
REQ: E-F-8	5 pts.	Anyone will be able to access ground level parking provided space is available.

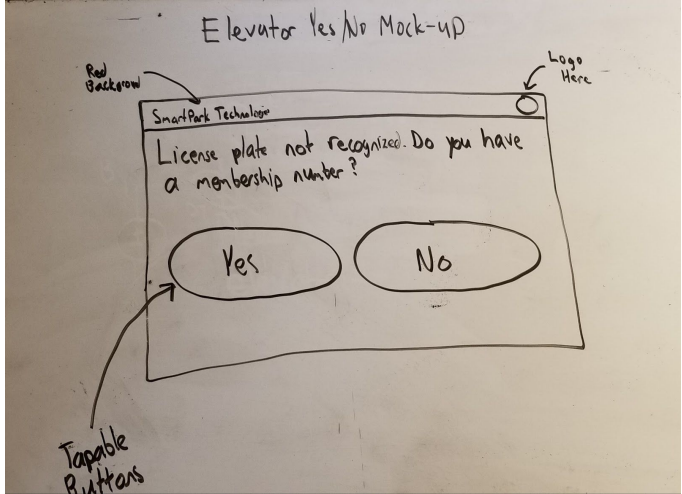
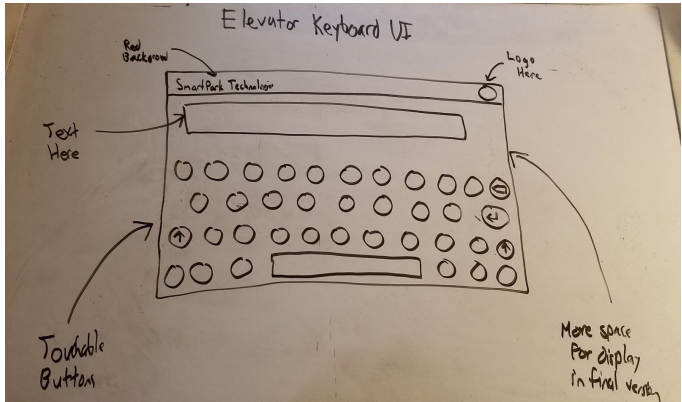
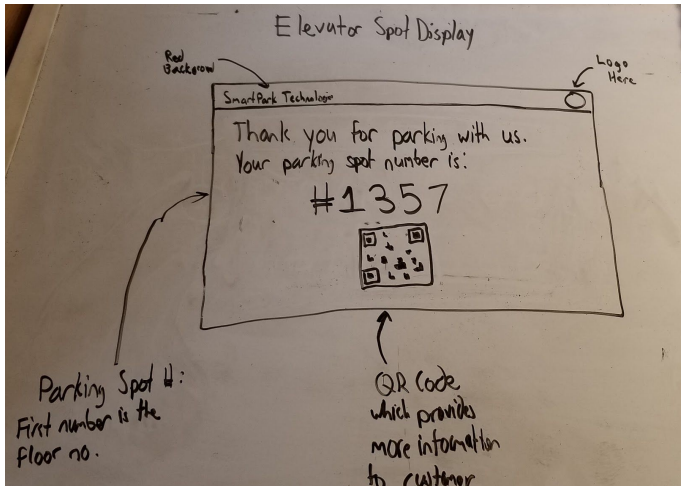
2. Non-Functional Requirements

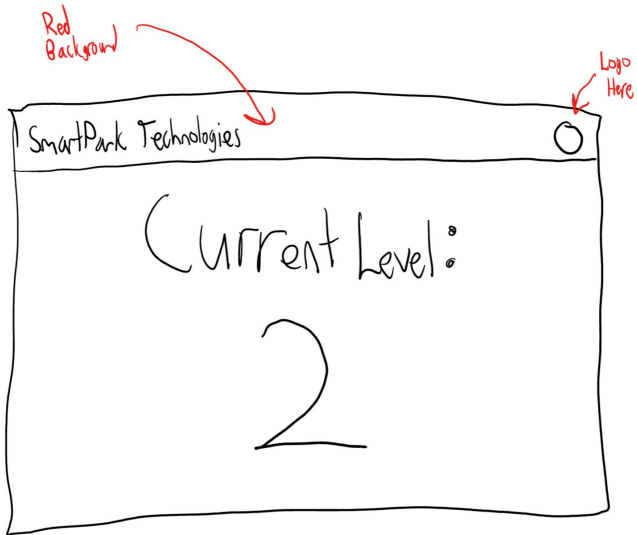
Requirement Identifier	Size	Requirement
REQ: E-NF-9	5 pts.	As the system operator, I will be able to obtain user membership info from the customer.
REQ: E-NF-10	3 pts.	As the administrator, I will be able to access the database in order to account for reservation extensions.
REQ: E-NF-11	3 pts.	As the administrator, I will increase the billing rate for an overstay the longer the duration.

REQ: E-NF-12	3 pts.	As the administrator, I will be able to overbook the parking garage.
REQ: E-NF-13	3 pts.	As the administrator I will give customers a rain check if there are no vacant spots.
REQ: E-NF-14	2 pts.	As the administrator I will be able to turn on/off dynamic pricing.

3. User Interface Requirements

Requirement Identifier	Size	Requirement
REQ: E-UI-15	5 pts.	<p>As a customer, I will be able to view my parking location and parking spot number by the elevator's screen.</p> 
REQ: E-UI-16	5 pts.	As the customer, I will be able to respond yes/no when asked if I have an existing reservation.

		
REQ: E-UI-17	4 pts.	<p>As the customer, I will be able to use an on-screen keyboard in order to enter my account information or reservation number.</p> 
REQ: E-UI-18	3 pts.	<p>As the customer, I will be able to scan a QR Code to get more detailed information about my parking spot.</p> 

REQ: E-UI-19	2 pts.	<p>As a customer, I will be able to see the current garage level I am on through the elevator's screen.</p> 
--------------	--------	--

Functional Requirement Breakdown

Stakeholders

The stakeholders into the project can be broken down into two different sub-groups: the subgroup that is using the parking garage and the subgroup that manages the parking garage.

Users of the Parking Garage

These stakeholders consist of anyone who uses the SmartPark service to make reservations and park in the garage. This can include:

- Customers
 - Returning customers or walk-ins, people who use the garage.
 - Can consist of only one person and/or groups of people, that share the account.
- Business Owners and Employees of that Business
 - May consist of walk-in or registered customers, but will typically be contracted.

Additionally, these stakeholders are contained in a few other sub-categories, such as:

- Walk-In Customers
 - Registered customers who arrive at the parking garage without a contract or reservation.
 - Unregistered customers that can have a reservation be made for them.
- Contracted Customers
 - Customers and businesses that make a long-term reservation with the garage for a certain number of spaces on a daily, weekly, monthly or annual basis.
- Returning/Registered Customers
 - Customers that are registered in the SmartPark database who have either made a reservation in the past or are registered in the system.

Managers of the Parking Garage

These stakeholders consist of any staff members or other teams that work closely with the parking garage. This may include:

- Managers
 - Owners of the parking garage.
 - Will design the garage through the SmartPark graphical interface and monitor the garage's activity through an administrative portal.
 - Will be able to set prices of the garage (suggested based on a pricing model) and view trends such as garage usage, popular parking times, revenue/accounts, etc.
- Employees
 - Will be able to monitor the garage's activity to make sure everything runs smoothly and efficiently.
 - Using a modified view of the garage created by the managers and will be able to view spots that are currently occupied or not.

Actors and Goals

Actor	Role of Actor	Actor's Goals
Customer	Initiating Actor	Register an account via the online portal. Make and cancel reservations to park. Enter the garage to park with as little system interaction as possible.
Employee of Garage/Manager	Initiating Actor	To view customer registration data, reservations, billing information. To be able to reconfigure the available parking spots with a GUI. View garage occupancy statistics and upcoming local events. Edit pricing and fees.
Parking Sensor	Participating Actor (supporting)	To send parking spot occupancy data to the parking spot management system.
License Plate Scanner	Participating Actor (supporting)	To send license plate numbers to the garage entry and exit systems.
Registration Terminal	Participating Actor	Provide an interface for the customer to interact with the registration, scheduling and billing portion of the system.
Elevator Terminal	Initiating + Participating Actor	To display which parking spot the customer should park in. Or to inform the customer they are not authorized to use the elevator by issuing a denial message.
Customer Reservation System	Participating Actor (supporting)	To accept customer login credentials and provide a GUI for creating or cancelling reservations. Also keeps track of the reservations customers have.

Central Server	Participating Actor (supporting)	To store information such as parking statistics, transaction histories, and garage layouts for administrative use.
----------------	-------------------------------------	--

Use Cases

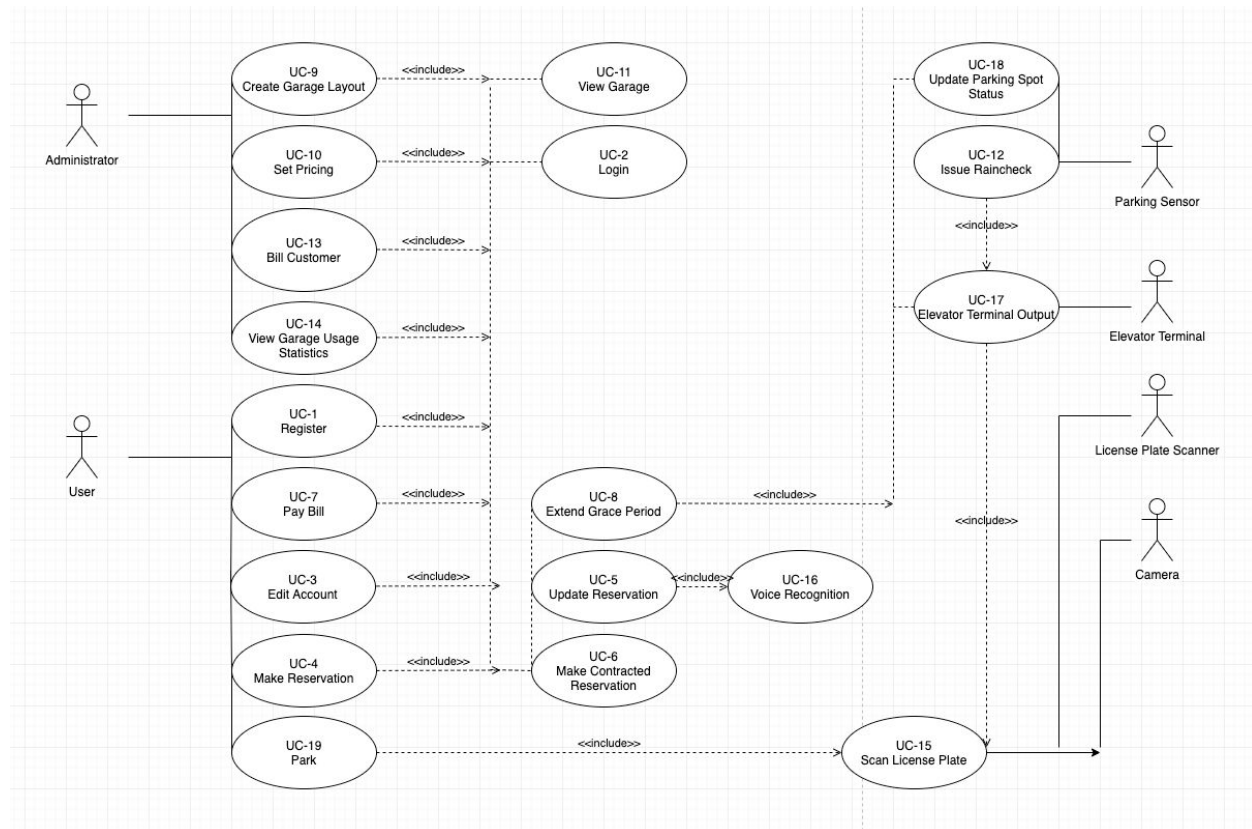
Casual Description

Use Case Identifier	Use Case Name	Description of Use Case
UC-1	Register	The user will be able to register for an account through the SmartPark website. The user will be asked to enter in their email address, a username, a password, and other demographic information.
UC-2	Login	The user will be able to access their SmartPark account using their email address and password or their username and password.
UC-3	Edit Account	After logging into their account, the user will be able to edit or update any information regarding their account. This includes changing their email address, username, password, adding a new registered vehicle or other information.
UC-4	Make Reservation	The user will be able to make a reservation once logged in.
UC-5	Update Reservation	The user will be able to change the date and time of their reservation, or cancel it.
UC-6	Make Contracted Reservation	The user will be able to make a contracted reservation for a parking spot.
UC-7	Pay Bill	The user can pay their bill through the online system.
UC-8	Extend Grace Period	The user can extend their grace period provided there is no conflict with another

		reservation.
UC-9	Create Garage Layout	The administrator of the system will be able to alter the arrangement and availability of parking spots through a GUI “drag and drop” system.
UC-10	Set Pricing	The administrator of the system will be able to set pricing for daily rates and contracted rates. This person can also turn on/off dynamic pricing.
UC-11	View Garage	The administrator and employees will be able to see the current layout of the parking garage, which spots are open/parked, etc.
UC-12	Issue Rain Check	The system shall determine whether an oversold event has occurred and issue a rain check to a user who cannot be accommodated.
UC-13	Bill Customer	The administrator can login to the privileged section of the online portal and bill the customer.
UC-14	View Garage Usage Statistics	The administrator of the system can login to the backend GUI to view usage statistics of the garage.
UC-15	Scan License Plate	The entry and exit cameras will scan the license plates of cars and forward this information to the garage entry and exit subsystems.
UC-16	Voice Recognition	The customer may use an Alexa/Google device to create a new reservation.

UC-17	Elevator Terminal Output	The elevator terminal displays the customer's parking spot if available. Otherwise displays rain-check notification.
UC-18	Update Parking Spot Status	The parking spot is updated to either be occupied when a customer is using the spot or vacant when there is no vehicle in the spot.
UC-19	Park	The customer has initiated the parking sequence by entering the garage.

Use Case Diagram



Traceability Matrix

	UC ID	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16	UC17	UC18	UC19
REQ ID	Cl.	4	4	1	6	6	5	5	2	1	8	5	4	8	8	6	8	3	8	4
C-F-1	2	X																		
C-F-2	3		X	X																
C-F-3	3				X						X						X			
C-F-4	5				X	X	X		X							X	X			
C-F-5	4				X	X	X										X			
C-F-6	3							X			X			X						
C-F-7	2							X			X									
C-F-8	4				X	X		X			X									
C-F-9	4							X			X			X			X			
C-F-10	3							X			X			X						
C-F-11	6								X			X	X	X		X	X			
C-F-12	5				X	X	X									X	X			
M-F-1	5										X				X	X		X	X	
M-F-2	3										X			X	X					
M-F-3	4										X	X	X	X						
M-F-4	3								X		X				X					
M-F-5	2																		X	X
M-F-6	1																X			
M-F-7	3	X	X												X					
M-F-8	3	X	X												X					
M-F-9	2												X	X						
M-F-10	1													X						
E-F-1	2														X				X	
E-F-2	5														X	X		X	X	X
E-F-3	1		X																	
E-F-4	5										X	X			X			X	X	
E-F-5	4	X														X	X			X
E-F-6	3					X	X												X	
E-F-7	4				X	X	X												X	
E-F-8	2																		X	X

Fully Dressed Descriptions

Use Case UC-1: Register	
Related Requirements	REQ: C-F-1, REQ: M-F-6
Initiating Actors	Customer, Employee of the Garage
Participating Actors:	Customer, Employee of the Garage, Database for Customer Information
Actor's Goal	To register for the website by using a unique username, password, email address, and other information
Preconditions	The user attempting to create an account cannot use an email or username that is currently registered in the system.
Postconditions:	The user's login information will be saved into the database.
Flow of Events for Main Success Scenario	<p>→1. The Customer/Employee selects the register account button on the main site page.</p> <p>→ 2. The user is then asked to enter their email address, username, and password.</p> <p>3. The user is brought to a page asking if they want to register a vehicle along with the registration</p> <p>←4. The Database for Customer Information then saves the data regarding the user's account.</p>
Flow of Events for Extensions (Alternate Scenarios):	<p>2A. The user's login information (email, username) is already taken</p> <ul style="list-style-type: none"> -The system prompts the user stating that either the email address or username is already taken -The system asks the user to select a new email address or username

Use Case UC-2: Login	
Related Requirements	REQ: C-F-2, REQ:M-F-7 REQ: E-F-3
Initiating Actors	Customer
Participating Actors:	Customer, Database for Customer Information
Actor's Goal	To access their SmartPark account using their email address and password or their username and password.
Preconditions	The user must have already registered to the website using a unique username, password, email address, and other information
Postconditions:	The user can now make a reservation, edit an existing reservation or edit account information.
Flow of Events for Main Success Scenario	<p>→ 1. The user is asked to enter their email address, username, and password if they have not done so already.</p> <p>← 2. The Database for Customer Information verifies the login credentials.</p> <p>← 3. The user is brought to a page where they can choose one of three buttons; making a new reservation, editing an existing reservation or editing account information.</p>
Flow of Events for Extensions (Alternate Scenarios):	<p>3A. The user chooses to make a new reservation -proceed to UC-4</p> <p>3B. The user chooses to edit an existing reservation -User can cancel reservation or adjust the timing of the reservation</p> <p>3C. The user chooses to edit account information -User can add vehicles, adjust payment method, change login information such as password and email.</p>

Use Case UC-4: Make Reservation	
Related Requirements	REQ: C-F-3, REQ: E-F-1, REQ: E-F-5
Initiating Actors	Customer
Participating Actors:	Customer, Employee of the Garage, Manager
Actor's Goal	To make a reservation in the garage through the online portal
Preconditions	The user must have already registered to the website using a unique username, password, email address, and other information required to login. They must also have a payment method on their account.
Postconditions:	The user will receive confirmation that they successfully made a reservation by receiving a message on the screen they made the reservation on.
Flow of Events for Main Success Scenario	<p>→ 1. The user is asked to enter their email address, username, and password if they have not done so already.</p> <p>← 2. The Database for Customer Information verifies the login credentials.</p> <p>← 3. The system checks which time slots have vacant parking spaces and lists them for the user.</p> <p>→ 4. The user is brought to a page that requires input as to what day the user would like to choose for their reservation.</p> <p>← 5. The system shows the available times for the day that the user selected</p> <p>→ 6. The user chooses a spot to reserve and is asked to confirm that all their information is correct and that they want that spot</p> <p>← 7. The user is shown a message that their reservation was successful as well as a unique identifier of their reservation.</p>
Flow of Events for Extensions (Alternate Scenarios):	<p>1. The user's login information (email, username) is already taken</p> <ul style="list-style-type: none"> - The system prompts the user stating that either the email address or username is already taken - The system asks the user to select a

	<p>new email address or username</p> <p>2. The user is not able to see the available parking spots (System error)</p> <ul style="list-style-type: none"> - User is asked to reload page or an error message pops up <p>3. There are no available spots for the date picked</p> <ul style="list-style-type: none"> - The user is automatically sent to the next available date with open slots
--	---

Use Case UC-7: Pay Bill	
Related Requirements	REQ: C-F-6, REQ: C-F-7, REQ: M-F-3
Initiating Actors	Customer
Participating Actors:	Customer, Manager
Actor's Goal	To pay off any existing bills incurred from reservations, overstays, etc.
Preconditions	The user must already have an account and be logged in with a payment method on their account
Postconditions:	The user will receive confirmation that they have successfully paid their bill and receive a receipt for the transaction which will be emailed to the email associated with their account
Flow of Events for Main Success Scenario	<p>→ 1. The user is asked to enter their email address, username, and password if they have not done so already.</p> <p>← 2. The Database for Customer Information verifies the login credentials</p> <p>← 3. The user is brought to a page that pulls up the bills associated with their account</p> <p>→ 4. The user is able to pay off their bill with either the payment method on file or they can type in a credit or debit card number, expiration date, and cvv.</p> <p>→ 5. The user is asked to check and make sure they want to pay off the selected bill with the payment provided.</p> <p>← 6. The user is shown shown a message</p>

	that they paid their bill successfully and a receipt is sent to the email address they have on their account
Flow of Events for Extensions (Alternate Scenarios):	<p>1. The user's payment method was invalid or expired</p> <ul style="list-style-type: none"> - User is prompted to enter a new payment method to complete the purchase <p>2. The user does not pay the bill within the billing period</p> <ul style="list-style-type: none"> - User is sent a notification that the payment is delayed and that the payment method listed will be used as an alternative backup

Use Case UC-9: Create Garage Layout	
Related Requirements	REQ: M-F-4
Initiating Actors	Manager
Participating Actors:	Central Server
Actor's Goal	To alter the arrangement and availability of parking spots through a GUI image render system.
Preconditions	The user must be logged in and have administrative privileges.
Postconditions:	The garage map will be updated in the reservation management system and displayed in the GUI. The virtual garage layout will accurately represent the real-life garage layout.
Flow of Events for Main Success Scenario	<p>→ 1. The manager opens the GUI of the garage layout by pressing the corresponding button.</p> <p>← 2. The system then creates an image render of the current garage layout.</p> <p>→ 3. The manager adds, deletes, and/or alters the virtual garage layout using mouse and keyboard controls.</p> <p>→ 4. The manager presses the finish button after they are finished making changes.</p> <p>← 5. The system displays a notification that</p>

	the change was successful and displays the new layout.
Flow of Events for Extensions (Alternate Scenarios):	<p>→ 3. The manager presses the cancel button before they are finished making changes.</p> <p>← 4. The system makes no updates to the garage layout and displays the original layout. The system returns to the main menu.</p>

Use Case UC-10: Set Pricing	
Related Requirements	REQ: C-F-6, REQ: C-F-7, REQ: C-F-8, REQ: C-F-9, REQ: C-F-10, REQ: M-F-1, REQ: M-F-2, REQ: M-F-3
Initiating Actors	Manager
Participating Actors:	Database for Pricing, Customers
Actor's Goal	Change the pricing for hourly, overstay, and peak-time parking as well as the cancellation fee
Preconditions	The user must be logged in and have administrative privileges.
Postconditions:	The data will be updated in the pricing table in the database
Flow of Events for Main Success Scenario	<p>→ 1. The Manager enters their desired price for hourly, peak-time, and overstay parking as well as inputs a cancellation fee.</p> <p>→ 2. The Manager then selects their desired start date. Note that the Manager cannot select dates that are in the past.</p> <p>→ 3. The Manager presses the submit button after they are finished making changes.</p> <p>← 4. The system updates the pricing in the database for all reservations for the corresponding dates.</p> <p>← 5. The system displays a notification that the change was successful.</p>
Flow of Events for Extensions (Alternate Scenarios):	<p>→ 3. The manager presses the cancel button before they are finished making changes.</p> <p>← 4. The system makes no updates and the pricing reverts to its original value.</p>

Use Case UC-11: View Garage	
Related Requirements	REQ: C-F-3, REQ: C-F-11, REQ: M-F-3, REQ: M-F-4, REQ: E-F-4
Initiating Actors	Employee of Garage/Manager
Participating Actors:	Central Server
Actor's Goal	The administrator and employees will be able to see the current layout of the parking garage, which spots are open/parked, etc.
Preconditions	The user must be logged in and have administrative privileges.
Postconditions:	Not applicable.
Flow of Events for Main Success Scenario	→ 1. The manager opens GUI of the garage layout. ← 2. The system then creates an image render of the current garage layout. → 3. The manager explores the virtual garage layout using mouse and keyboard controls. → 4. The manager presses the cancel button after they are done. ← 5. The system returns to the main menu.
Flow of Events for Extensions (Alternate Scenarios):	Not applicable.

Use Case UC-14: View Garage Usage Statistics	
Related Requirements	REQ: C-F-3, REQ: C-F-11, REQ: M-F-3, REQ:M-F-4
Initiating Actors	Manager
Participating Actors:	Parking Database
Actor's Goal	To view statistics for revenue, occupancy and overstay for a specific time period.
Preconditions	The Manager must be logged in and have

	admin access.
Postconditions:	N/A
Flow of Events for Main Success Scenario	→ 1. The Manager clicks on the statistics tab. → 2. The Manager enters the desired time period to view statistics. ← 3. The Database retrieves the requested data for the given time slot. ← 4. The system renders the data in a graphical interface.
Flow of Events for Extensions (Alternate Scenarios):	Not applicable.

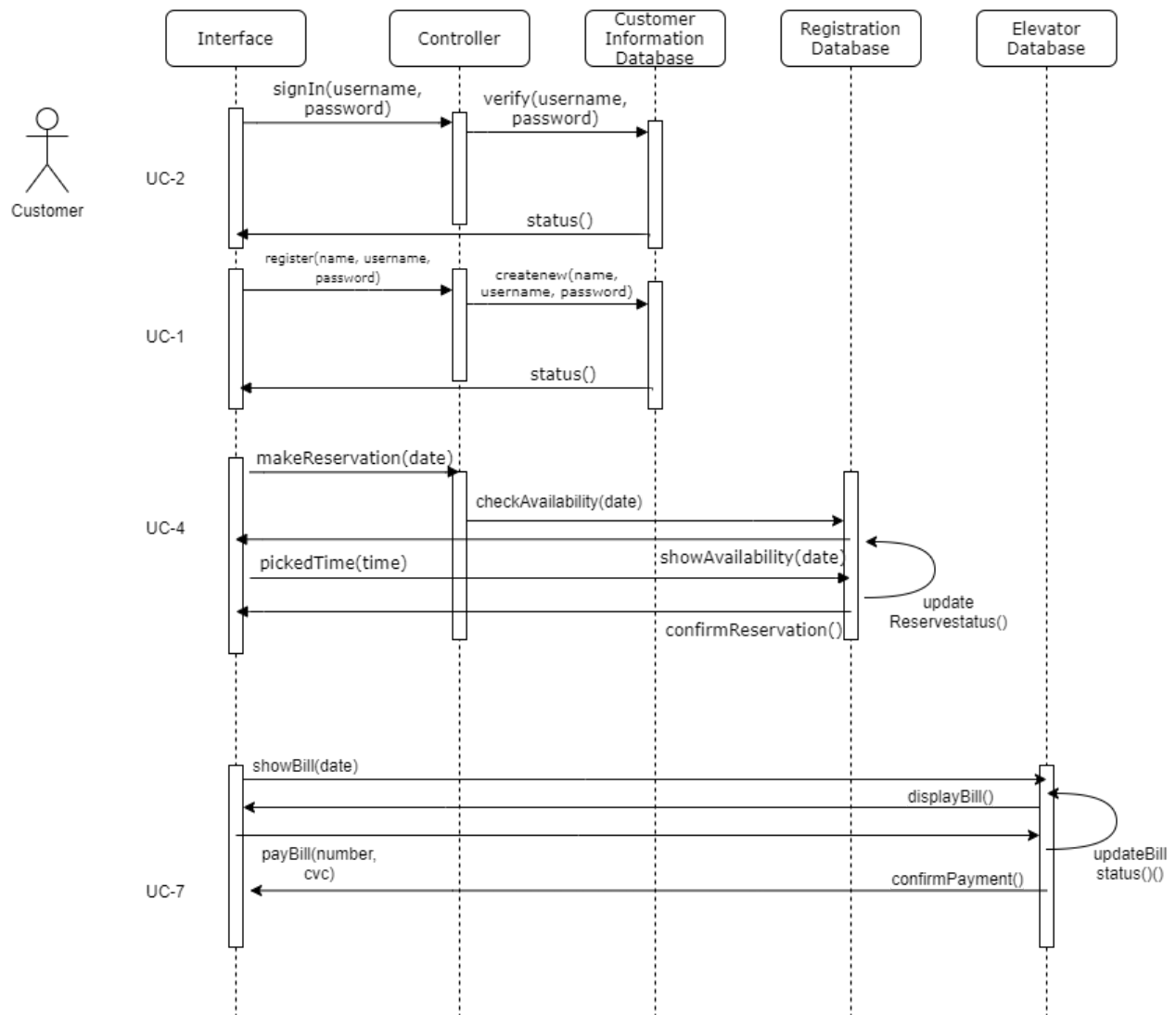
Use Case UC-18: Update Parking Spot Status	
Related Requirements	REQ: M-F-1, M-F-5, E-F-1, E-F-2, E-F-4, E-F-6, E-F-7, E-F-8
Initiating Actors	Elevator Terminal
Participating Actors:	Customer, Employee of the Garage/Manager, Customer Reservation System
Actor's Goal	To update a parking spot's current status as either occupied or vacant.
Preconditions	Not applicable.
Postconditions:	The parking spot's status will be updated as either vacant or occupied.
Flow of Events for Main Success Scenario	→1. The customer's vehicle enters or exits the parking spot. 2. The Parking Sensor will sense the vehicle either entering or leaving the spot. ← 3. The Parking Sensor will update the database which stores information about each parking spot.

Use Case UC-19: Park	
Related Requirements	REQ: M-F-4
Initiating Actors	Elevator Terminal

Participating Actors:	Customer
Actor's Goal	To park in the Parking Garage
Preconditions	The consumer may or may not have a reservation. There needs to be an open spot on the ground level if the customer is a walk-in. There needs to be an open spot in general if it is a car with a reservation.
Postconditions:	If the customer is a walk-in, they will get a ticket and a parking spot on the ground floor. If the consumer already has a reservation, then they are allowed to park if there is a spot. However if they have a reservation, and there is no space they are given a raincheck. If they have walked in and there is no space, they are not given a spot in the parking garage.
Flow of Events for Main Success Scenario	<p>→ The customer enters the parking garage</p> <p>→ The license plate reader reads and either recognizes the plate number from an existing database of reservations or does not find it in the database.</p> <p>→ If it does not find it in the database, it asks for a reservation number. If there is no reservation number then it is treated as a walk in customer.</p> <p>→ If a reservation is found in the system with that number, then the customer is led on to the next set of events to lead the car to a parking spot.</p> <p>→ Otherwise the car is taken to the open spot, if they have a valid reservation.</p>
Flow of Events for Extensions (Alternate Scenarios):	<p>→ If the license plate is recognized, then there is a check done for whether there is an open spot.</p> <p>→ If there is an open spot the customer is led to the spot, otherwise they are given a raincheck.</p>

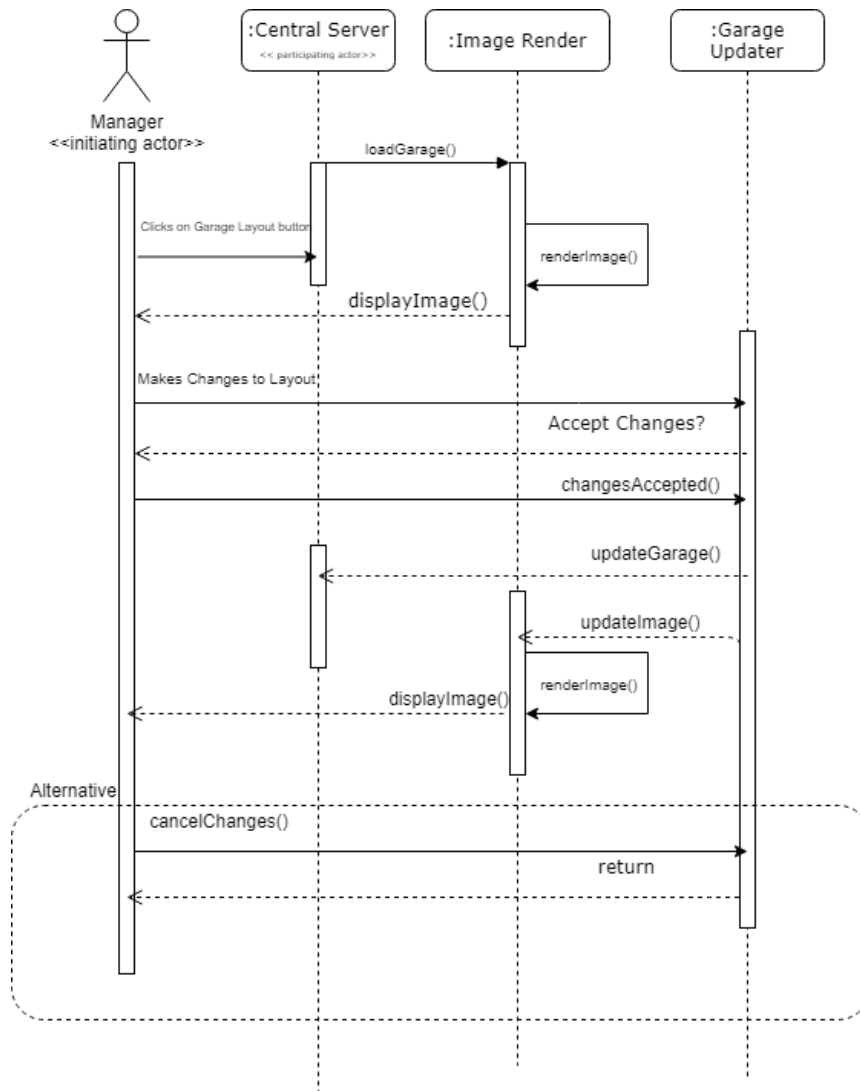
System Sequence Diagrams

Customer Subgroup

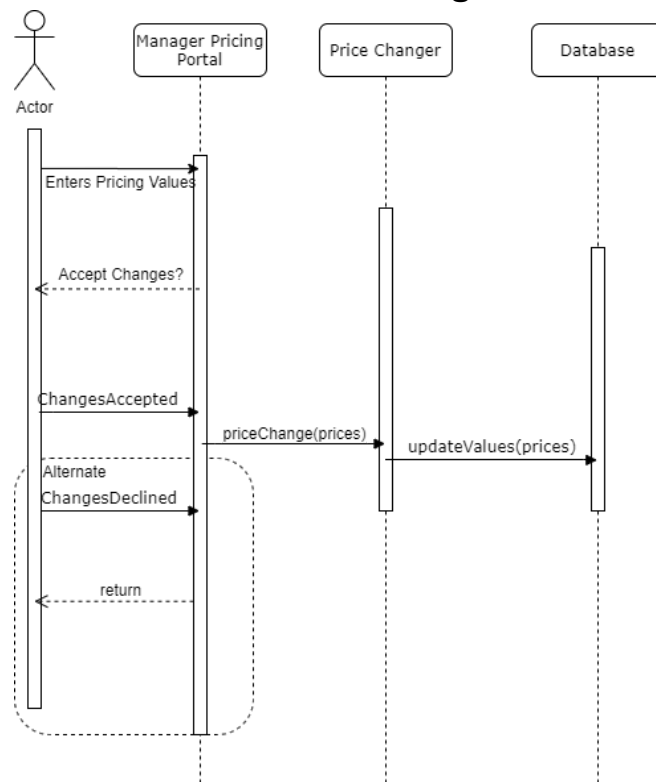


Managerial / Administrative Subgroup

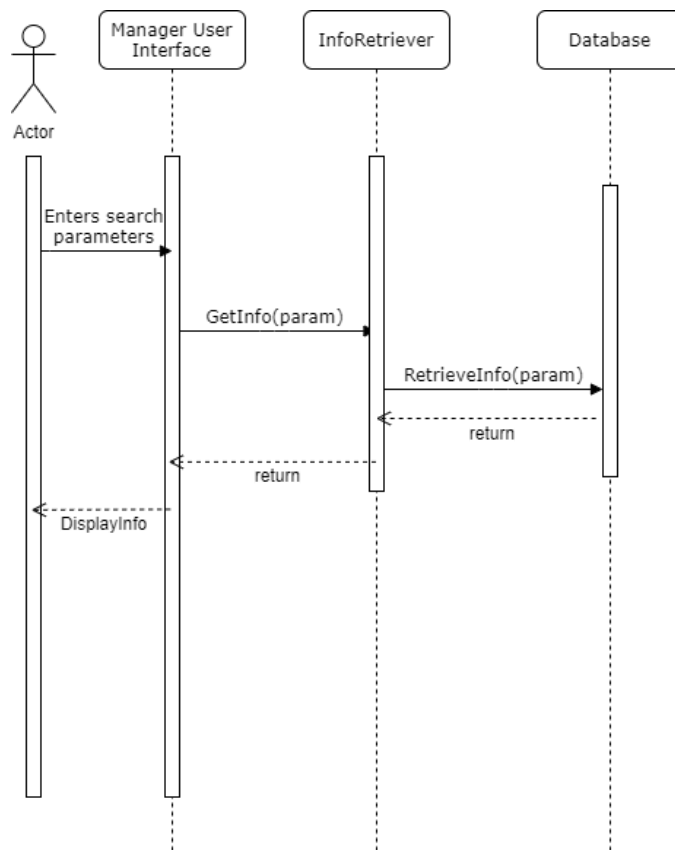
Use Case #9



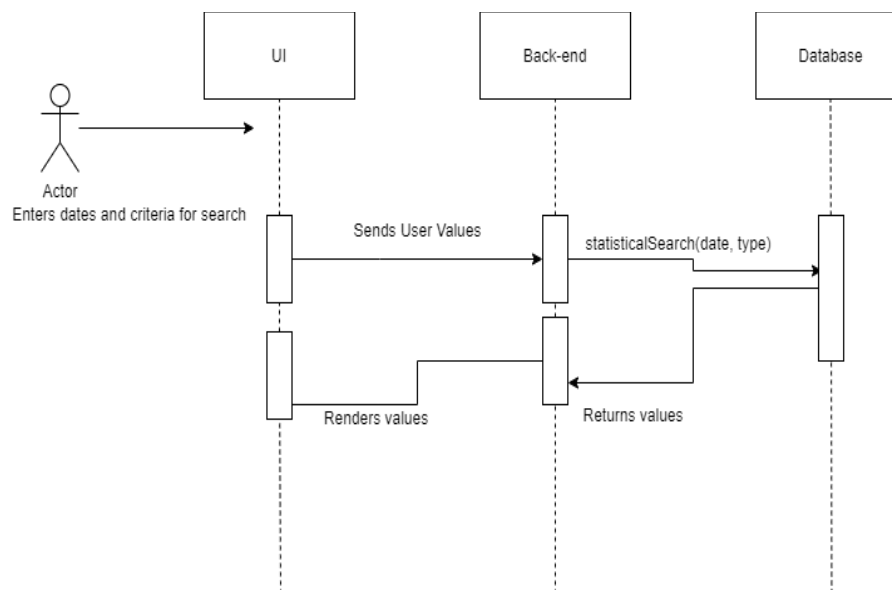
Use Case #10: Set Pricing



Use Case #11: View Garage Information

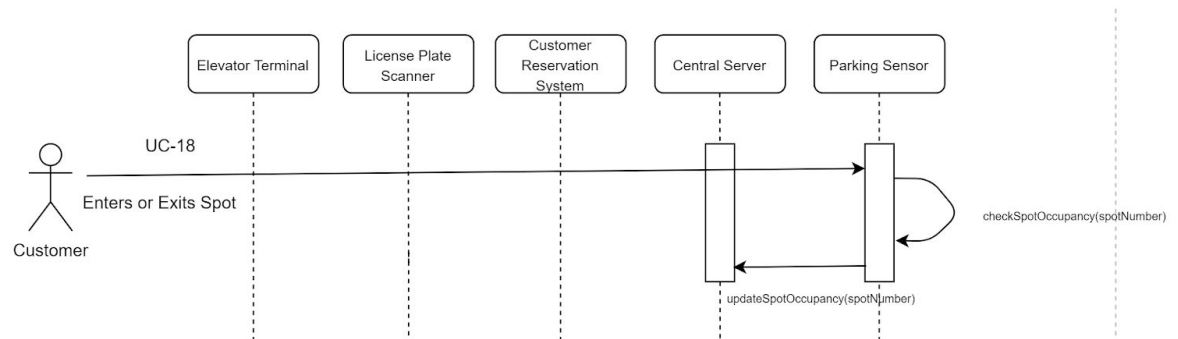


Use Case #14: View Statistics



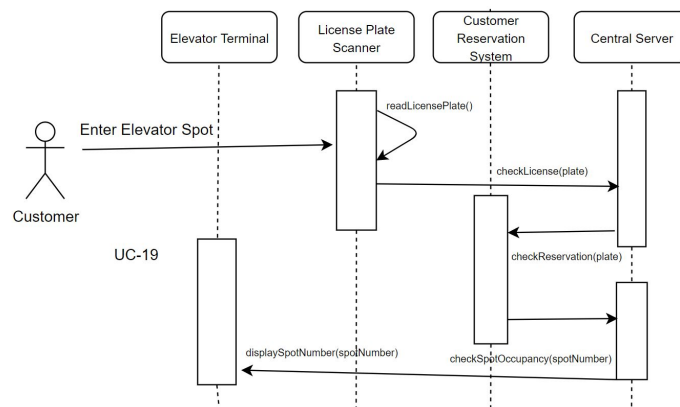
Elevator Operation Subgroup

Use Case #18

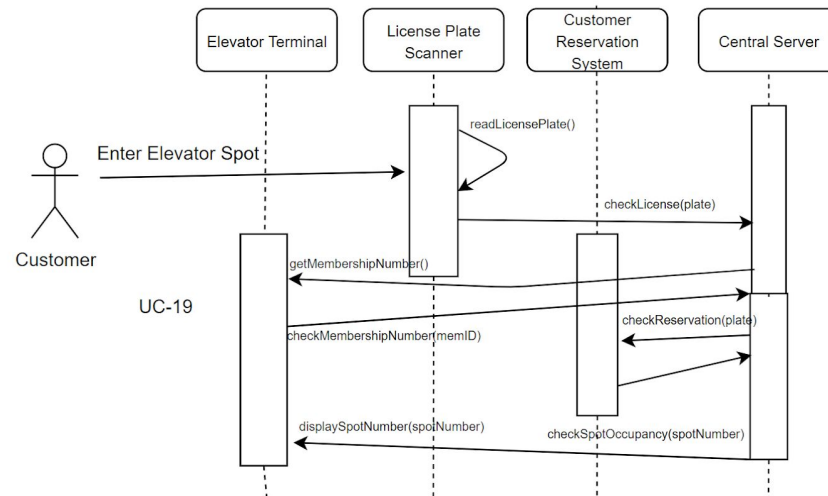


Use Case # 19

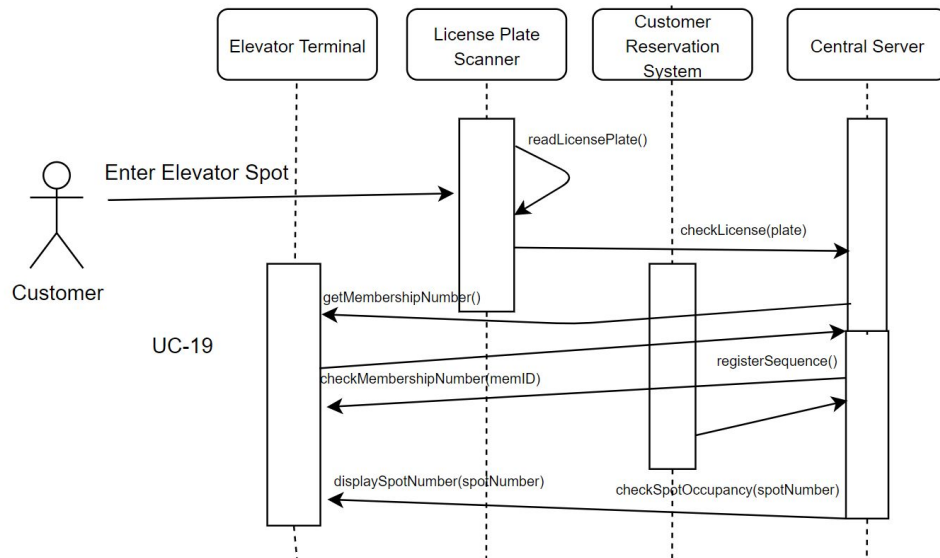
On success:



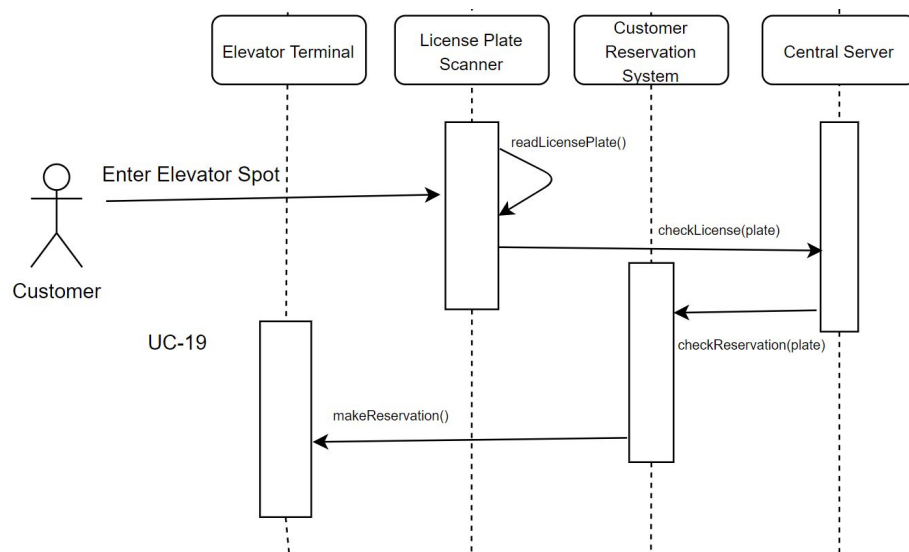
When plate is not recognized:



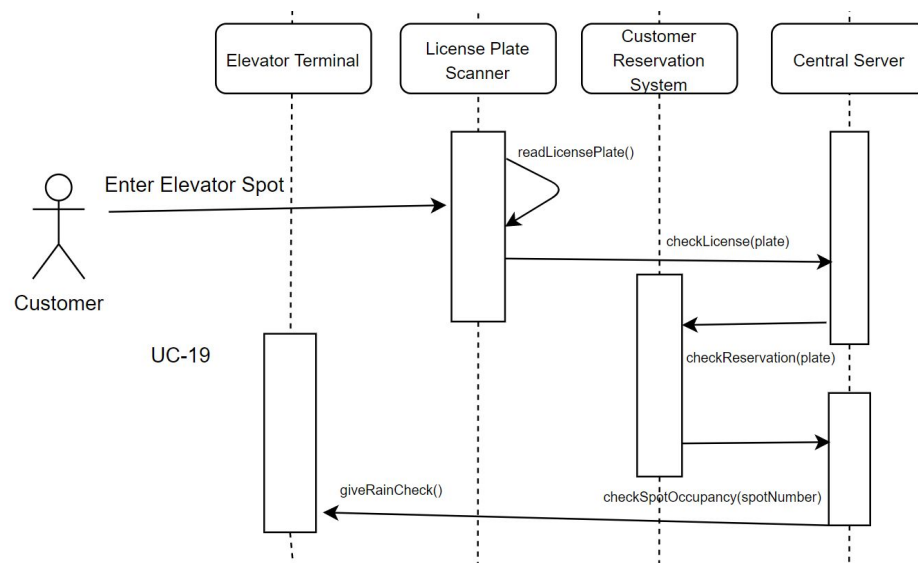
When membership number is not recognized:



When the customer does not have a reservation:



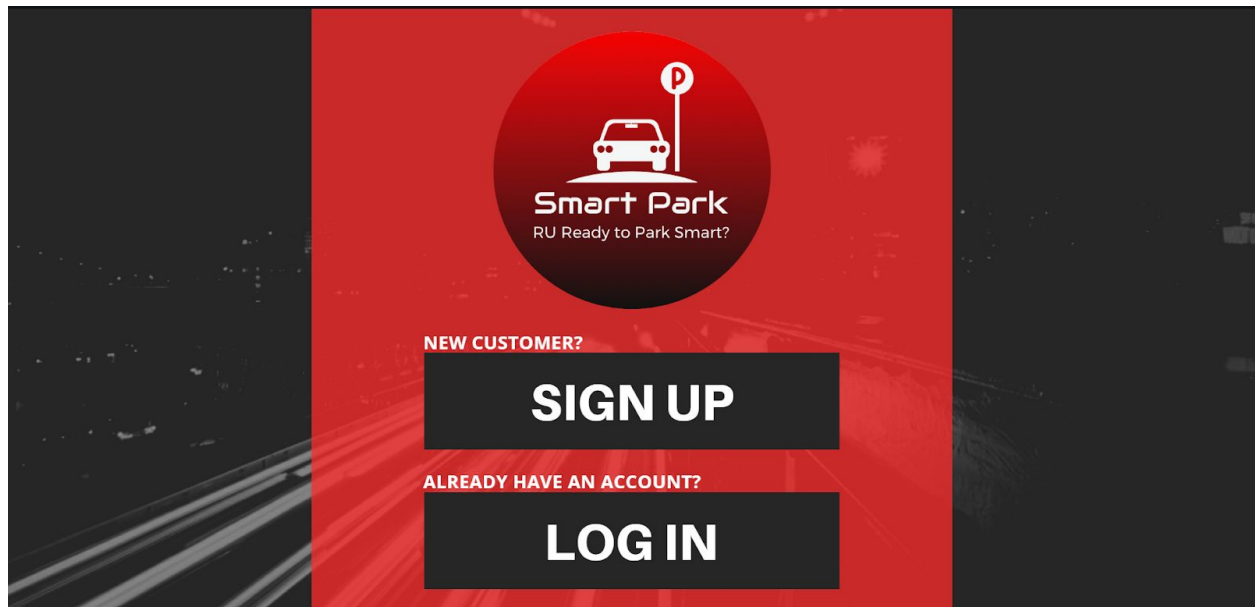
When the garage is full:



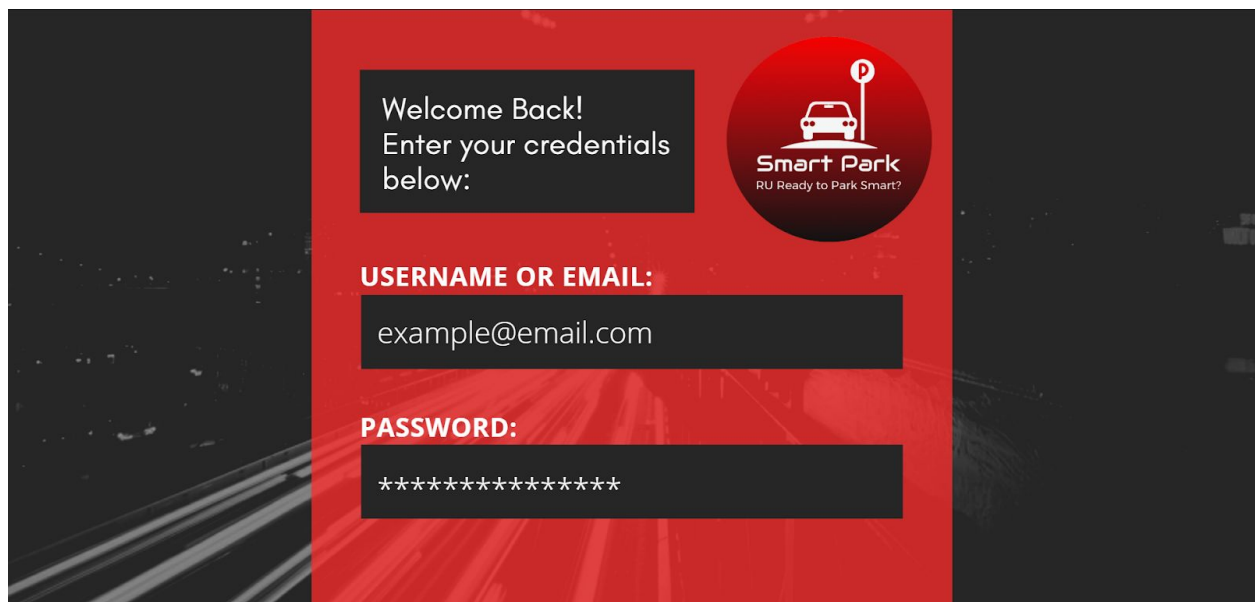
User Interface Specification Breakdown

Customer Registration Subgroup

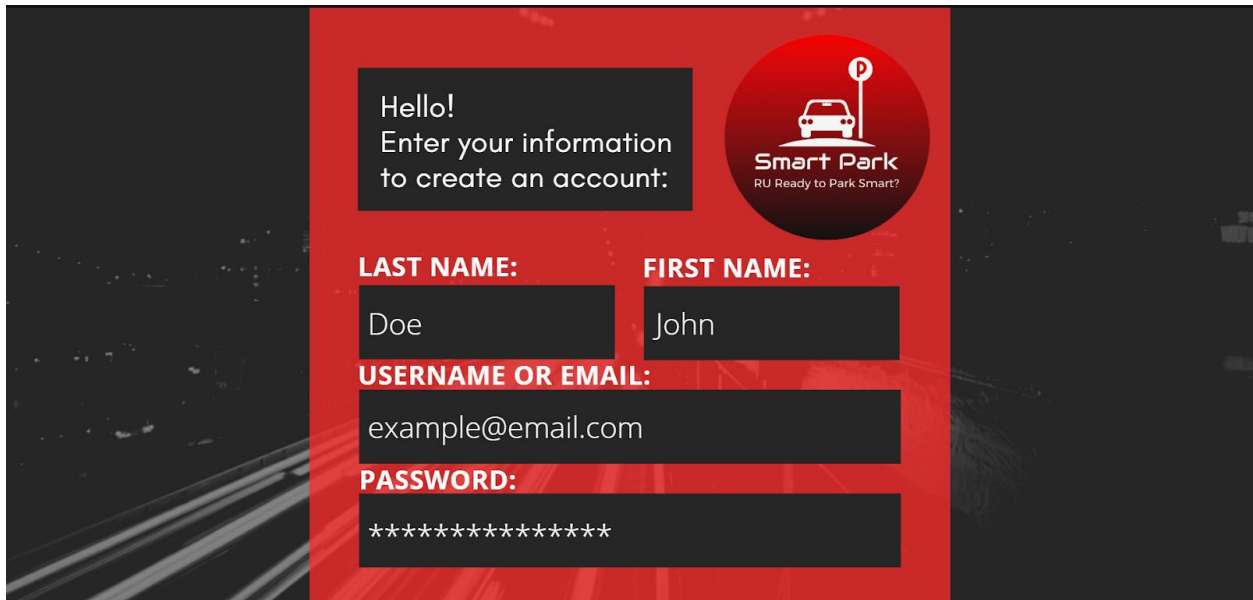
This is the first screen of the website, the user will be prompted to pick one of the options depending on if they are an existing or new customer.



If the user is an existing customer and clicked the 'Login' button on the prior screen they will be guided to the Login screen. This screen will ask for the user's Username or Email and their account's corresponding Password. This will ensure the identity of the customer logging in.



If the customer is a new customer and clicked on the 'Sign up' button on the prior page. They will be directed to this screen which will ask for their First and Last Name, email and password. This information will then be stored in the customer information database that can be accessed whenever the user logs in again.



The image shows a sign-up screen for 'Smart Park'. It has a red background with a dark grey sidebar on the left and right. In the top right corner, there is a circular logo with a car icon and the text 'Smart Park' and 'RU Ready to Park Smart?'. The main content area is a dark grey box with white text. It says 'Hello! Enter your information to create an account:'. Below this, there are four input fields: 'LAST NAME:' with 'Doe', 'FIRST NAME:' with 'John', 'USERNAME OR EMAIL:' with 'example@email.com', and 'PASSWORD:' with '*****'.

Hello!
Enter your information
to create an account:

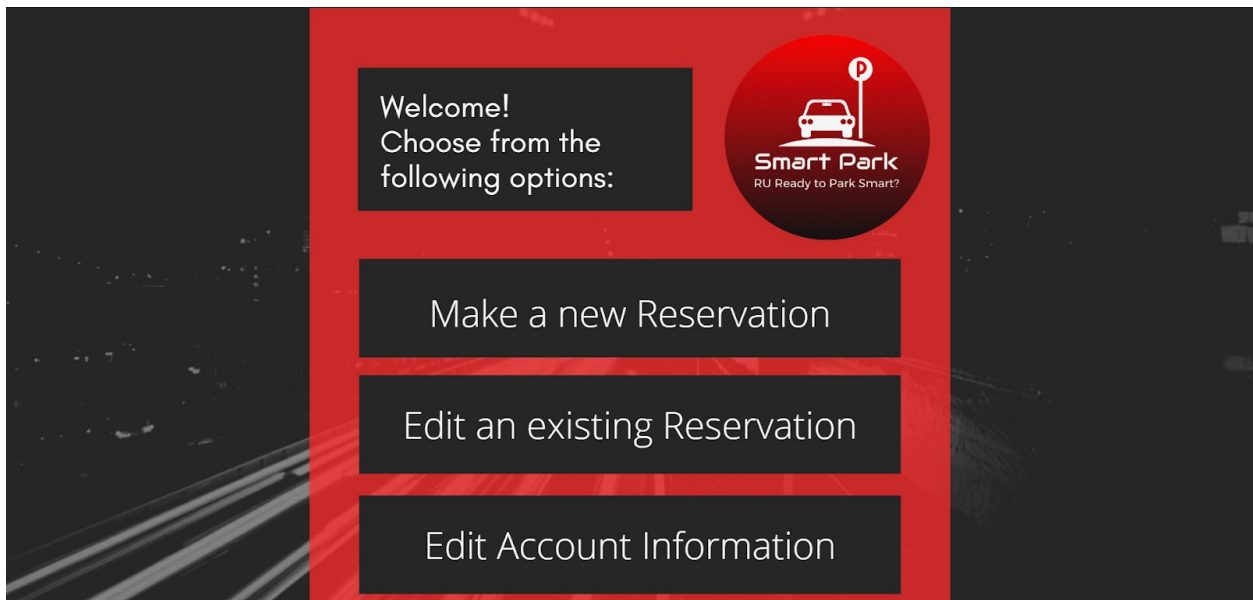
LAST NAME:
Doe

FIRST NAME:
John

USERNAME OR EMAIL:
example@email.com

PASSWORD:

After the customer accesses the system either from logging in or signing up for an account they are directed to the page below that asks them to pick from one of the following tasks; to make a new reservation, edit an existing one or edit their account.



The image shows a welcome screen for 'Smart Park'. It has a red background with a dark grey sidebar on the left and right. In the top right corner, there is a circular logo with a car icon and the text 'Smart Park' and 'RU Ready to Park Smart?'. The main content area is a dark grey box with white text. It says 'Welcome! Choose from the following options:'. Below this, there are three buttons: 'Make a new Reservation', 'Edit an existing Reservation', and 'Edit Account Information'.

Welcome!
Choose from the
following options:

Make a new Reservation

Edit an existing Reservation

Edit Account Information

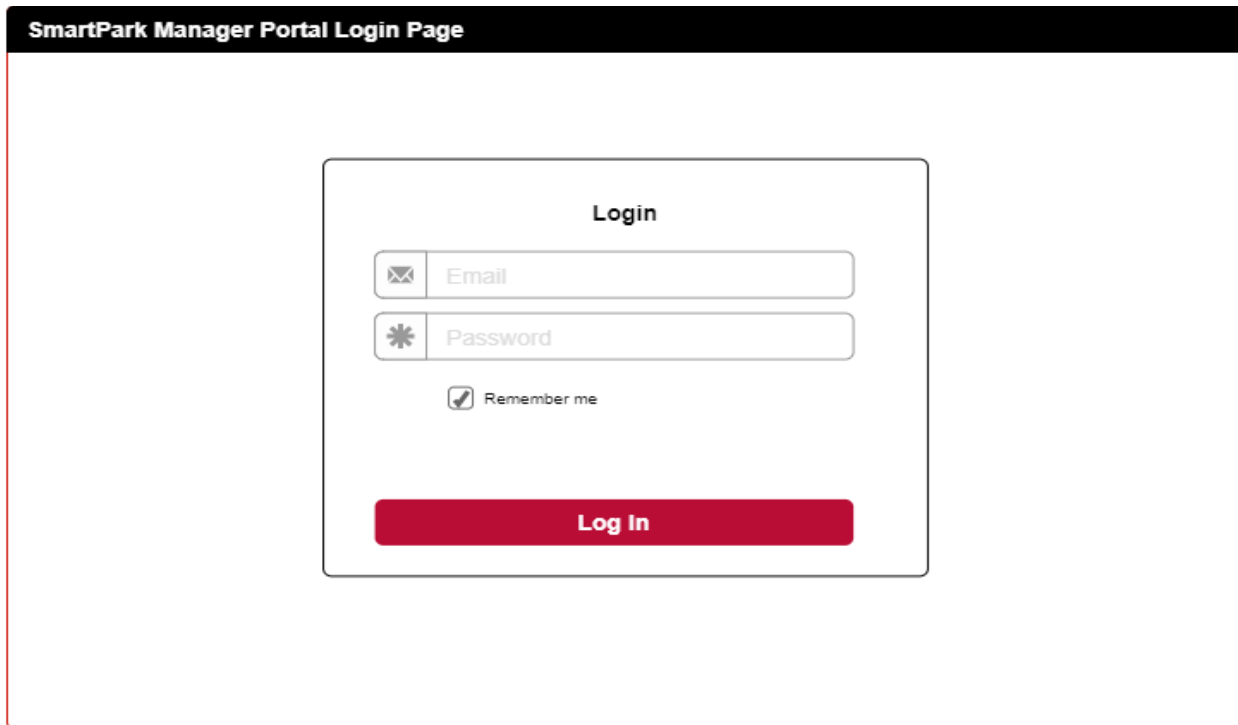
Estimated User Effort

The user's estimated effort would depend on whether they are an existing or new customer. This interface is designed to be a clean and intuitive representation. The user would need a maximum of 2 keystrokes to get to their required task. It can also be extended to add upcoming reservations on the left hand side column and a notifications bar on the right side column.

The 'Make a New Reservation' button takes the user to a screen where the user would pick the type of reservation they want to make and then pick the date(s) they wish to park there. The 'Edit an existing Reservation' button takes the user to a screen that displays all the available reservations that are on record with the SmartPark system. The 'Edit Account' button would take the user to a screen that would let them edit their account or view their monthly bills.

Managerial / Administrative Subgroup

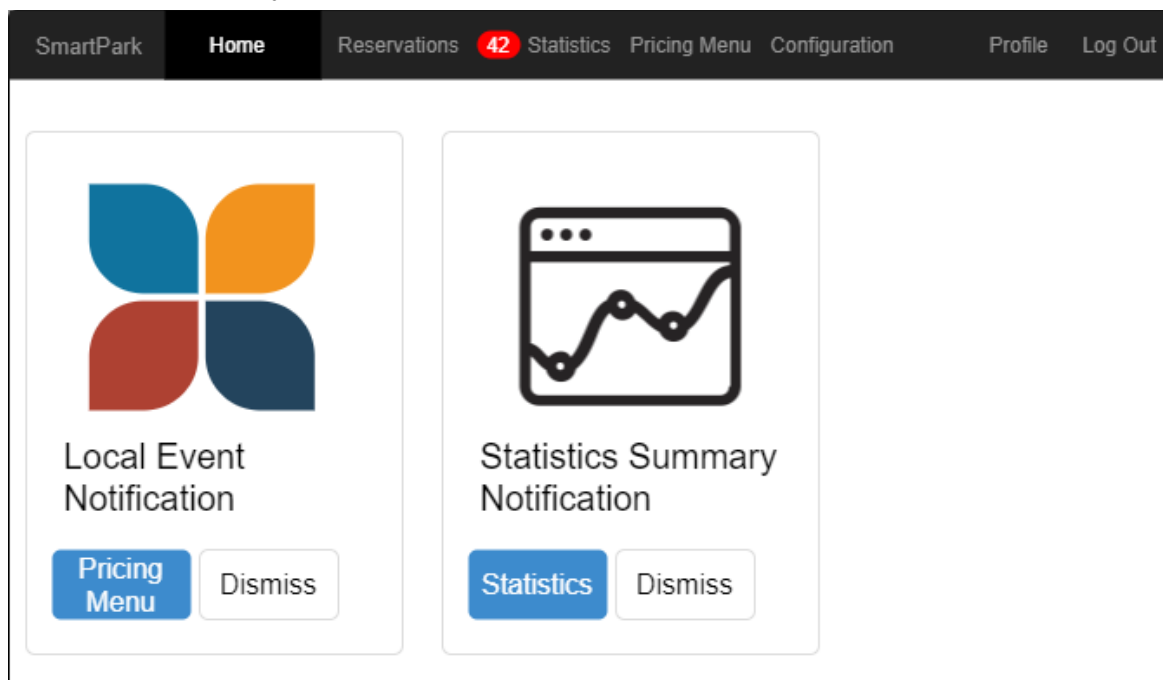
The user, an employee with administrative access, first enters their credentials into the SmartPark Manager Login Page. If the Email or Password is incorrect, the system will not allow the user to proceed and will indicate that the credentials are incorrect.



The image shows the SmartPark Manager Portal Login Page. It features a central login form with the following elements:

- Title:** Login
- Email Field:** A text input field with an envelope icon on the left and the label "Email".
- Password Field:** A text input field with a star icon on the left and the label "Password".
- Remember me:** A checkbox with a checkmark icon and the text "Remember me".
- Log In Button:** A red button with the text "Log In".

Next, the user will enter their home page. The home page will have cards that display information about any local events.



The image shows the SmartPark Manager Home Page. It features a navigation bar at the top with the following items:

- SmartPark
- Home (active)
- Reservations
- 42 (notification badge)
- Statistics
- Pricing Menu
- Configuration
- Profile
- Log Out

The main content area displays two cards:

- Local Event Notification:** Features a logo with four colored squares (blue, orange, red, dark blue). Below the logo is a "Pricing Menu" button and a "Dismiss" button.
- Statistics Summary Notification:** Features a line graph icon. Below the icon is a "Statistics" button and a "Dismiss" button.

The user can then click on one of the tabs on the navigation bar to complete whatever tasks they have in mind. The first tab is the Reservation tab. The reservation tab allows the user to view existing reservations by customer. The user can then select if they want to cancel the reservation or bill the customer.

The screenshot shows the SmartPark application interface. The top navigation bar includes 'SmartPark', 'Home', 'Reservations' (with a red badge '42'), 'Statistics', 'Pricing Menu', 'Configuration', 'Profile', and 'Log Out'. On the left, there are two tabs: 'Reservations' and 'Current Overview'. The main content area displays a table of reservations with columns: Customer Name, Reservation Time, Reservation Type, and Actions. The table contains two rows: Ross Gellar (8:00 AM - 5:00 PM, Subscription) and Chandler Bing (9:00 AM - 7:00 PM, One-Time). Each row has 'Cancel' and 'Bill' buttons in the Actions column. Two modal dialogs are shown on the right. The first dialog, 'Cancel Reservation?', has a red header and text stating the reservation will be cancelled with a refund. It has a red 'Cancel' button and a 'Don't Cancel' link. The second dialog, 'Send Bill?', has a yellow header and text asking for confirmation to send the bill. It has an orange 'Send' button and a 'Cancel' link. Arrows indicate the flow from the 'Cancel' and 'Bill' buttons in the table to their respective modal dialogs.

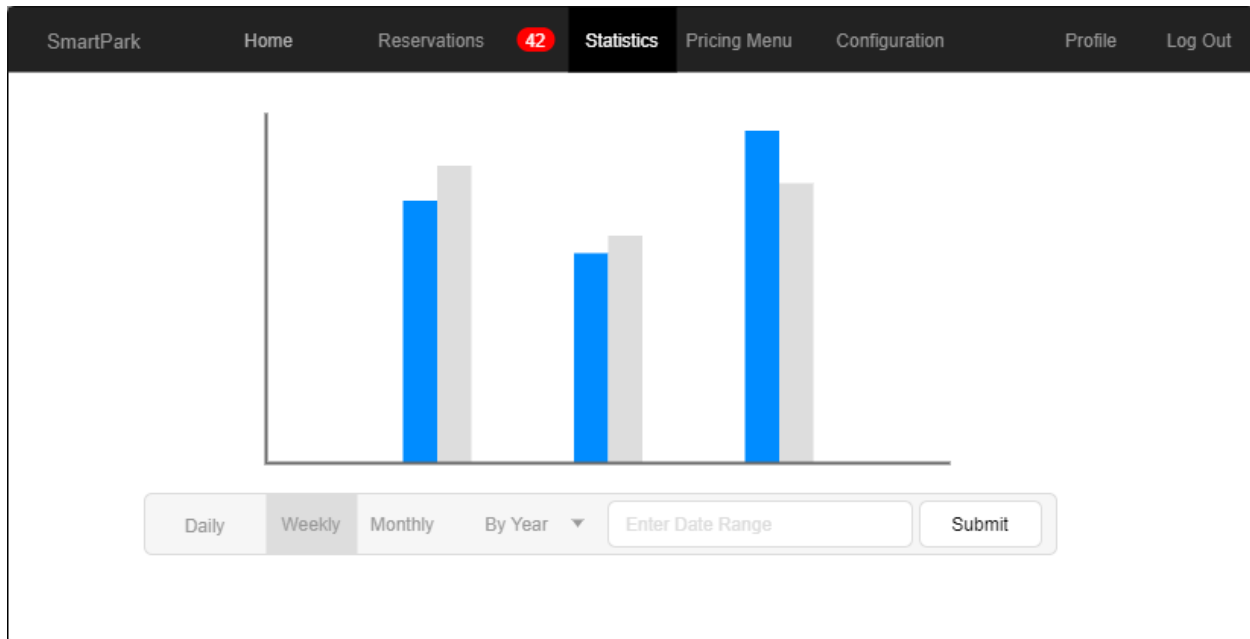
Customer Name	Reservation Time	Reservation Type	Actions
Ross Gellar	8:00 AM - 5:00 PM	Subscription	Cancel Bill
Chandler Bing	9:00 AM - 7:00 PM	One-Time	Cancel Bill

The user can then select the Current Overview tab on the left to view current garage status.

The screenshot shows the SmartPark application interface with the 'Current Overview' tab selected. The top navigation bar is the same as the previous screenshot. The left sidebar now has 'Reservations' and 'Current Overview' tabs, with 'Current Overview' being the active tab. The main content area displays a table of garage spots with columns: Floor, Spot, Upcoming Reservations, and Current Status. The table contains two rows: Floor 1, Spot 200 (8:00 PM - 11:00 PM, Occupied) and Floor 2, Spot 200 (None, Free). Each row has a status button in the Current Status column. A pagination bar at the bottom shows page numbers 1 through 9, with page 2 selected.

Floor	Spot	Upcoming Reservations	Current Status
1	200	8:00 PM - 11:00 PM	Occupied
2	200	None	Free

When the user navigates to the statistics tab, they can view statistics for revenue, occupancy and overstay for a chosen time period.



In the pricing menu, the user can input values for hourly, overstay, and peak-time parking as well as configure the cancellation fee. The user can then select the effective date for these price changes. To complete their changes, the user presses the submit button.

The screenshot shows the 'Pricing Menu' tab in the SmartPark application. The navigation bar at the top includes 'SmartPark', 'Home', 'Reservations' (with a red badge showing '42'), 'Statistics', 'Pricing Menu' (the active tab), 'Configuration', 'Profile', and 'Log Out'. The main content area is titled 'Pricing Adjustment Menu' and contains the instruction 'Enter desired values and select effective date.' Below this, there is a table with four rows for pricing adjustments:

Hourly Parking Price	\$.00
Overstay Rate	\$.00
Cancellation Fee	\$.00
Peak Time Rate	\$.00

To the right of the table is a calendar widget for October 2014. The calendar shows the days of the week (Mo, Tu, We, Th, Fr, Sa, Su) and the dates. The 24th is highlighted in blue. At the bottom right of the form are two buttons: 'Cancel' and 'Submit'.

The configuration tab allows the user to edit the number of spots and floors available in their garage. A rendering would be shown on the right after the user selects their desired values. If the user decides to edit the number of spots, the bottom left dropdown will appear and the user can edit spots on a specific floor. If the user decides to edit the available floors the top right dropdown will appear. The user then selects if they wish to add a floor or remove a floor. If they remove a floor, the bottom right dropdown appears and the user can select which floor to remove. If they add a floor, the text entry box appears. After the user has made their desired changes, the user presses the submit button.

SmartPark
Home
Reservations
42
Statistics
Pricing Menu
Configuration
Profile
Log Out

Edit Garage Configuration

Edit

Edit Available Floors
Edit Available Spots

Spots

Select Floor

Enter Number of Spots

Floors

Add


Remove

Edit Floors

Add A Floor
Remove a Floor

Remove Floor

Enter Number of Floors

Render of Parking
Garage Appears
Here


Cancel
Submit

Generated Summary

The monthly revenue summary is automatically generated. It displays peak hour, number of reservations, and revenue by day. A user can enter a date to view the details for that day.

SmartPark	Home	Reservations	42	Statistics	Pricing Menu	Configuration	Profile	Log Out
-----------	------	--------------	----	------------	--------------	---------------	---------	---------

Monthly Revenue Summary			
Search...			Go!
Date	Revenue	Number of Reservations	Peak Hour
Tuesday, January 28, 2020	\$1345.00	29	11:00 AM - 12:00 PM
Wednesday, January 29, 2020	\$1625.00	35	1:00 PM - 2:00 PM

← Older	Newer →
---------	---------

Estimated User Effort

The number of mouse-clicks to accomplish any of the user's tasks is relatively low. The reservation tab displays data by search and filters. This requires some data entry from the keyboard. Any of the actions the user wishes to take in the reservations tab requires 2-3 mouse clicks. The statistics tab may require user entry to view data for a specific time period. Daily, weekly, and monthly statistics can be reached in one click. To view data over the course of a year, the user must select the year from the dropdown, requiring an extra click. The pricing adjustment menu requires data entry and one click. In the final implementation, we may be able to shorten this by providing suggestions based on previous entries. The garage configuration tab requires between 2-3 clicks and some data entry, but includes a rendering which may take extra time to load.

Example: View Revenue and Occupancy Statistics for one week: 2 clicks

1. Click the Statistics tab
2. Click the Weekly button

Example: Search and Cancel a Customer's Reservation: 3 clicks, 1 entry

1. Click the Reservation tab
2. Enter the Customer's name
3. Click Cancel.
4. Confirm Cancel.

Elevator Operation Subgroup

The following is a step-by step description of how the user will interact with the elevator terminal when going to park. The potential flow of choices are described below. This flow of choices is not an exhaustive list.

Scenario 1: Has reservation, space available, and the license plate can be read.

The user enters the garage.

The user drives into the elevator and the system identifies a valid reservation.

The license plate scanner successfully reads the user's license plate.

The in-elevator display notifies the driver of the floor and spot designator assigned to the user.

The user exits the elevator at the correct floor and parks.

Overall user input: 0

Scenario 2: Has reservation, oversold lot.

The user enters the garage.

The user drives into the elevator and the system identifies a valid reservation.

The in-elevator display notifies the customer that no spots are available, issues rain-check.

The user exits the elevator and exits the garage.

Overall user input: 0

Scenario 3: Has reservation, space available, but license plate not recognized.

The user drives into the elevator.

The in-elevator display notifies the user that the license plate cannot be read and so a reservation cannot be found.

The in-elevator display requests the user input their login information. (Yes/No interaction and entering information via keyboard, two screens)

The user enters their login information and a reservation is found.

The in-elevator display notifies the driver of the floor and spot designator assigned to the user.

The user exits the elevator at the correct floor and parks.

Overall user input: 2 screens of user interaction: a yes/no interaction and entering information through the keyboard.

Scenario 4: Does not have reservation, space available, license plate is recognized.

The user drives into the elevator.

The system successfully scans the license plate but the user does not have a reservation.

The system prompts the user for a reservation/membership number in case it was missed. (One screen of keyboard interaction with the user)

The system prompts the user if they would like to still park (with additional charges applied). (One screen of Yes/No interaction with the user)

The user selects a section of the garage to park in or exits the elevator. (One touch screen interaction with the user)

Overall user input: 3 screens of interaction: a touch screen interaction, a yes/no interaction and entering information through the keyboard.

Scenario 5: The user does not have an account.

The user drives into the elevator.

The in-elevator display notifies the user that the license plate cannot be read and so a reservation cannot be found.

The in-elevator display requests the user input their login information. (One screen of keyboard interaction with the user)

The user does not have user information.

The user is prompted to register for an account (One screen of keyboard interaction with the user)

The system prompts the user if they would like to still park (with additional charges applied).

(One screen of Yes/No interaction with the user and one touch screen interaction with the user)

The user selects a section of the garage to park in or exits the elevator.

Overall user input: 4 screens of interaction: Two keyboard interactions, one yes/no interaction and one touch screen interaction.

Certain scenarios can be intertwined with one another, therefore some situations may have a higher amount of user effort required than the effort values listed above.

Two walkthroughs of potential scenarios encountered are displayed below using the mock-up user interfaces. The user interfaces may not be representative of the final design.

When a vehicle is not present in the elevator, the default screen is displayed below. The placeholder text is an editable field where the manager can input the garage name.



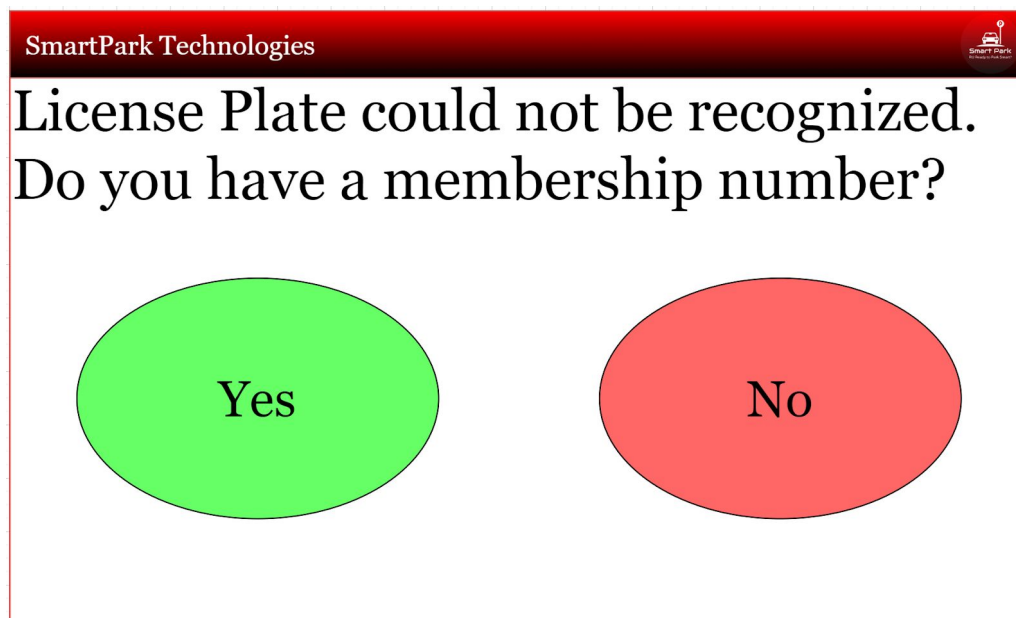
In the event that the user's license plate is recognized, then the user's parking spot is displayed and a QR code containing more information about his or her spot is available to be scanned.



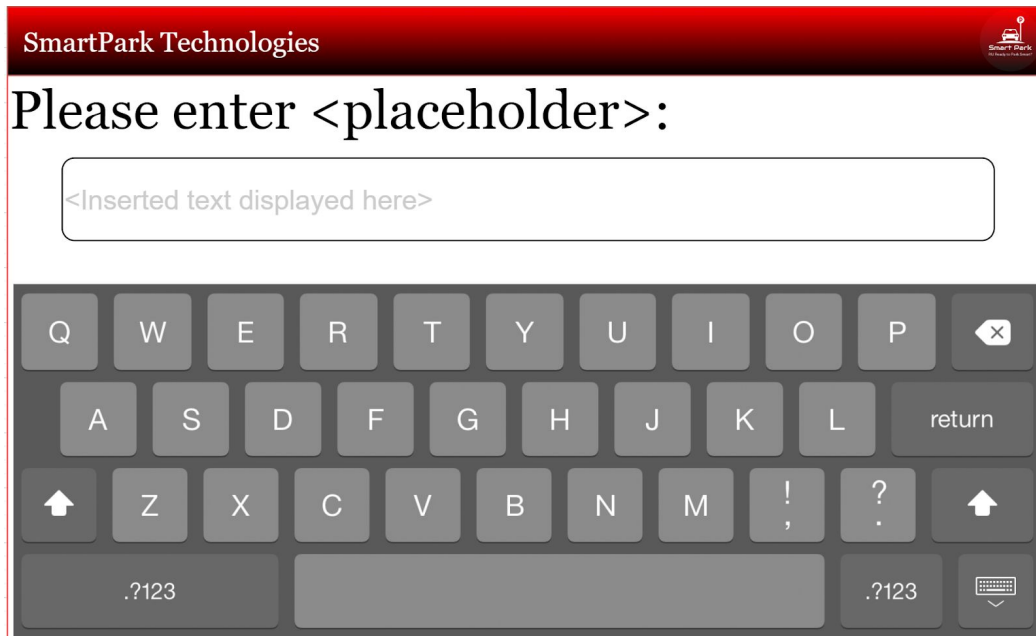
Following a successful scan of the license plate and a reservation is recognized, the current floor of the parking garage is displayed to the user. When the elevator stops, the current level and the floor number will turn green to let the user know it is safe to depart the elevator.



In the event the user's license plate could not be recognized, the elevator prompts the user for his or her login information through a yes or no button press. If the user selects the wrong option, a back button on the subsequent screen will be available for the user to redo the selection. The buttons will be activated through a touchscreen.



If the user needs to enter his or her login information, the user is brought to a screen with a simple keyboard interface. The user will enter his or her information and will continue to the next screen, which may consist of a yes or no question or the option to select a parking section (see customer user interface section). When a membership number/reservation number needs to be entered, a number pad interface will be displayed instead.



Estimated Effort by User

The effort exerted by the user in the elevator terminal ranges vastly depending on how much interaction the user has done with the SmartPark system before entering the elevator. The elevator's user interface is set up in such a way that interaction with the user can be done very simply: two touch buttons for yes and no can easily allow for the elevator control system to make decisions on what to do next in order to minimize user input.

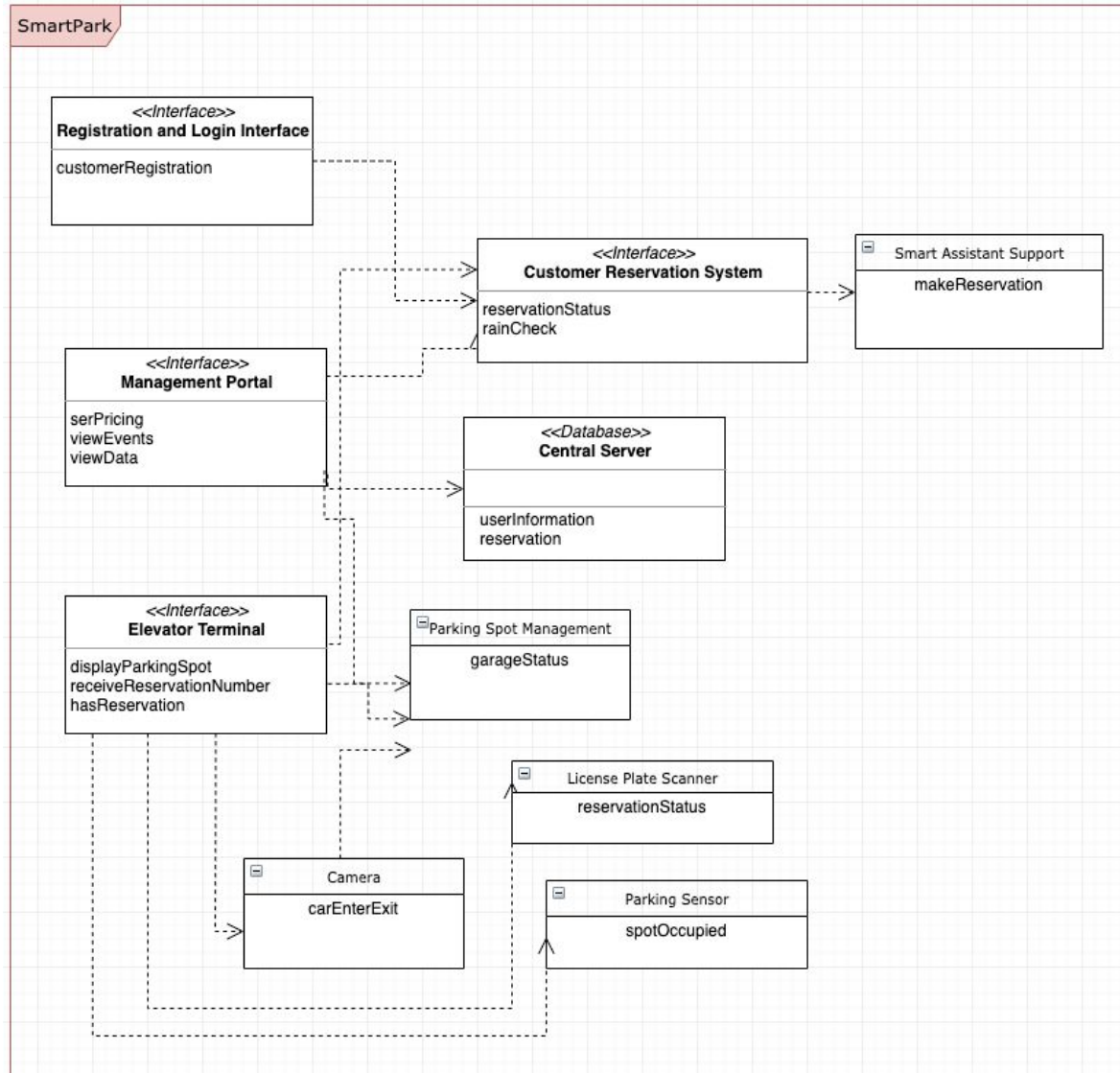
The ideal user scenario is if the user's license plate is recognized and the user has already made a reservation, or the garage is full, however the former is much more ideal for the user than the latter. Both of these situations require zero input from the user: the user only needs to drive up to the elevator and the elevator does all of the work getting him or her to the requested parking zone. This scenario takes the most preparation beforehand, however this saves the most time in the elevator.

The worst case scenario is the situation where the user's license plate is not recognized and the user does not have an account in the system. This would ultimately lead to the most button presses for the user, which would vary depending on how long the user's login information is. In this scenario, it may be important to consider the case of making the user exit the elevator to then create an account and a reservation in the ground floor terminals. The issue is, however, that this may cause congestion in the garage as the user would be forced to back up out of the elevator.

Domain Analysis

Domain Model

The domain model was created by using the use cases and requirements of the parking garage specifications. The functional requirements and actors were analyzed and then transformed into different concepts for the parking garage. With this vast amount of concepts, we ensure that no one concept has too many responsibilities.



Concept Definitions

Concept Identifier	Responsibility Description	Type	Concept
DC-1	<ul style="list-style-type: none"> Sends information regarding a parking spot's status to the parking management system. Determines when a vehicle is using a parking spot or not. Parking spot status is transmitted as a boolean (false for a vacant spot, true for an occupied spot). 	D	Parking Sensor
DC-2	<ul style="list-style-type: none"> Displays the customer's parking spot to the user if the license plate is scanned correctly or after receiving the user's reservation number through the terminal. Communicates with the customer reservation system to determine whether or not the customer at the elevator has a reservation or not. Allows the user to enter in their login information to confirm a reservation number through the registration and login interface. 	D	Elevator Terminal
DC-3	<ul style="list-style-type: none"> Keep track of when vehicles enter and exit the garage. Sends data to the central server. 	D	Camera
DC-4	<ul style="list-style-type: none"> Scans the license plate of the vehicle entering the elevator. Sends license plate information to the customer reservation system to see if there is a match and therefore a reservation. 	D	License Plate Scanner
DC-5	<ul style="list-style-type: none"> Stores user login information, vehicles registered, and garage information. Communicates between the customer reservation system, registration and login interface, parking spot management system and management portal. 	D	Central Server (Database)
DC-6	<ul style="list-style-type: none"> Communicates with the elevator terminal to send information regarding a customer's reservation. Checks the time of the reservation and whether the customer is in the grace period or not. Determines if rain checks are to be issued to the customer by communication with the management 	K	Customer Reservation System

	portal and parking spot management system.		
DC-7	<ul style="list-style-type: none"> Allows the customer or the employee to register for the first time to the SmartPark website or login if already registered. Communicates with the central server which stores all the login information. Allows the customer to register a new vehicle, check the garage status, display bill up to the current date and pay his or her bill. 	K	Registration and Login Interface
DC-8	<ul style="list-style-type: none"> Holds the status of all of the parking spots in the garage. Communicates with the central server and updates the customer reservation system when spots are filled. Spot availability is also sent to the management portal 	D	Parking Management System
DC-9	<ul style="list-style-type: none"> Allows the manager to set pricing, enable dynamic pricing, design the garage using the graphical interface, view garage statistics and local events 	D	Management Portal
DC-10	<ul style="list-style-type: none"> Allows the user to enter in information without using a keyboard, such as reservation times and login information. 	K	Smart Assistant Support

Association Definitions

Concept Pair	Association Description	Association Name
Parking Sensor \longleftrightarrow Parking Management System	Parking sensor determines whether or not a parking spot is occupied or not. This information is then sent to the parking management system where it is stored until the next update.	Update Parking Status
Elevator Terminal \longleftrightarrow Customer Reservation System	The customer reservation system checks to see if the vehicle in the elevator has a reservation. The information is then sent to the elevator terminal and proceeds depending on if the customer has a reservation or not.	Notify About Reservation
Elevator Terminal \longleftrightarrow Registration and Login Interface	If the customer did not have a registration, then the customer will have to provide his or her account details. This is done through the registration and login interface which is used by the elevator terminal.	Elevator Login
Camera \longleftrightarrow Central Server (Database)	The camera watches the vehicles entering and exiting the garage as well as a timestamp of when they arrived and left. This information is transmitted to the central server to be stored and used for the employees of the garage.	Store Vehicle Data
Central Server (Database) \longleftrightarrow Management Portal	Data is extracted from the database and is displayed to the management portal as charts and graphs. Additionally, the database can be accessed by the managers to manually add, delete, or sort information.	View Statistics
License Plate Scanner \longleftrightarrow Customer Reservation System	The vehicle's license plate is scanned and is then cross-checked against the customer reservation system.	Check Reservation
Customer Login System \longleftrightarrow (Database) Customer Registration Information	The login information provided by the user is checked against the database to see if the credentials provided are correct to give them access to their account.	Check Login Credentials
Customer Reservation System \longleftrightarrow Managerial Database	To give the customer the option to view their current billing cycle from the first day of the month to the current day.	Check Bill

Management Price Tool ↔ Central Server (Database)	The Management Price Tool requests the database to update the prices for hourly rates and fees	Set Pricing
Management Garage Configuration Tool ↔ Central Server (Database)	The Management Garage Configuration Tool requests the database to make updates to the number of spots or number of floors	Configure Garage
Management Events Tool ↔ Central Server (Database)	The Management Events Tool retrieves information on local events from the Database	View Local Events

Attribute Definitions

Responsibility	Attribute	Concept
RC1 Know whether a car is parked in the spot or not.	spotOccupied	Parking Sensor
RC2 Display on screen which parking spot the customer should use.	displayParkingSpot	Elevator Terminal
RC3 Display accepts customer reservation numbers via GUI.	receiveReservationNumber	
RC4 Display walk-in or Reservation status.	hasReservation	
RC4 Takes photos of entering and exiting license plates.	carEnterExit	Camera
RC5 Determines whether or not an inbound car has a reservation.	reservationStatus	License Plate Scanner
RC6 Stores user information.	userInformation	Central Server (Database)
RC7 Stores reservation information.	reservation	
RC9 Checks time of customer reservation against current status.	reservationStatus	Customer Reservation System
RC10 Issues rain check if the garage is full.	rainCheck	
RC11 Accepts customer data for registration of an account.	customerRegistration	Registration and Login Interface
RC12 Maintains occupancy status of all spots in the garage.	garageStatus	Parking Management System
RC13 Allows administrators to set pricing policies.	setPricing	Management Portal
RC14 Allows administrators the ability to view local events	viewEvents	
RC15 Allows administrators to view historic occupancy data	viewData	
RC16 Allows customers to make reservations using voice interface.	makeReservation	Smart Assistant Support

Traceability Matrix

	UC ID	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16	UC17	UC18	UC19
DC ID	Ct.	3	3	3	6	5	4	3	5	3	2	5	6	4	3	3	4	5	4	7
DC1	2																		X	X
DC2	3				X								X					X		
DC3	5							X			X				X				X	X
DC4	4				X						X				X					X
DC5	19	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DC6	13	X	X	X	X	X	X	X	X				X	X			X	X		X
DC7	13	X	X	X	X	X	X	X	X				X	X			X	X		X
DC8	9					X		X	X		X	X	X		X			X	X	X
DC9	6									X	X	X	X	X	X					
DC10	4				X	X	X										X			

System Operation Contracts

Operation	Register
Precondition	The user must have already registered to the website using a unique username, password, email address, and other information. <code>isRegisteredUser(loginName) == false</code>
Postcondition	The user's login information will be saved into the database. <code>customerRegistration(loginName) == success</code>

Operation	Login
Precondition	The user attempting to create an account cannot use an email or username that is currently registered in the system. <code>isRegisteredUser(loginName) == true</code>
Postcondition	The user can now make a reservation, edit an existing reservation or edit account information. <code>hasUserAccess(loginName) == true</code>

Operation	Make Reservation
Precondition	The user must have already registered to the website using a unique username, password, email address, and other information required to login. They must also have a payment method on their account. <code>hasPaymentOnFile(loginName) == true</code> <code>isRegisteredUser(loginName) == true</code>
Postcondition	The user will receive confirmation that they successfully made a reservation by receiving a message on the screen they made the reservation on. <code>sendResConfirmation(loginName)</code>

Operation	Pay Bill
Precondition	The user must already have an account and be logged in with a payment method on their account. <code>hasPaymentOnFile(loginName) == true</code>
Postcondition	The user will receive confirmation that they have successfully paid their bill and receive a receipt for the transaction which will be emailed to the email associated with their account

	sendPaymentConfirmation(loginName)
--	------------------------------------

Operation	Create Garage Layout
Precondition	The user must be logged in and have administrative privileges. hasAdminAccess(loginName) == true
Postcondition	The garage map will be updated in the reservation management system and displayed in the GUI. The virtual garage layout will accurately represent the real-life garage layout. displayGarage() updateGarage(currentLayout)

Operation	Set Pricing
Precondition	The user must be logged in and have administrative privileges. hasAdminAccess(loginName) == true
Postcondition	The data will be updated in the pricing table in the database. updatePricing(pricingOptions)

Operation	View Garage
Precondition	The user must be logged in and have administrative privileges. hasAdminAccess(loginName) == true
Postcondition	None

Operation	View Garage Usage Statistics
Precondition	The Manager must be logged in and have admin access. hasAdminAccess(loginName) == true
Postcondition	None

Operation	Update Parking Spot Status
Precondition	None
Postcondition	The parking spot's status will be updated as either vacant or occupied. spotOccupied == true if car in spot spotOccupied == false if car not in spot

Operation	Park
Precondition	<p>The consumer may or may not have a reservation. There needs to be an open spot on the ground level if the customer is a walk-in. There needs to be an open spot in general if it is a car with a reservation.</p> <p>hasReservation(licensePlate) == true Prompt appropriately in the elevator GUI.</p> <p>hasReservation(licensePlate) == false Prompt appropriately on ground floor GUI.</p>
Postcondition	<p>If the customer is a walk-in, they will get a ticket and a parking spot on the ground floor.</p> <p>If the consumer already has a reservation, then they are allowed to park if there is a spot. However if they have a reservation, and there is no space they are given a raincheck. If they have walked in and there is no space, they are not given a spot in the parking garage.</p> <p>garageFull == true Issue rain check</p> <p>isWalkin(licensePlate) Initiate walk-in policy.</p> <p>elevatorPrompt(licensePlate)</p>

Mathematical Models

Poisson Random Process

The mathematical model used to simulate cars leaving and arriving at the garage will be based on two Poisson random processes: one to simulate arrivals and one to simulate departures. The probability of seeing n arrivals in a time interval Δt is:

$$Pr(n) = \frac{e^{-\lambda \Delta t} (\lambda \Delta t)^n}{n!} \text{ and } E\{n\} = \lambda \cdot \Delta t.$$

We can use the same model to simulate the departures.

To simulate the time between arrivals, we can generate an exponential random numbers $rx(u) = \frac{-\ln(u)}{\lambda}$, where u is a unit of one hour and λ represents the average number of arrivals per hour. Then, we can perform the arrival simulation as follows:

1. Change the status of one spot from “Available” to “Occupied.” If there are no available spots, mark as an “Overbooked” event.
2. Generate a random value rx as defined above. After a time $t(rx) = 60rx$ minutes, perform step 1 again.

The same process can simultaneously be applied to departures.

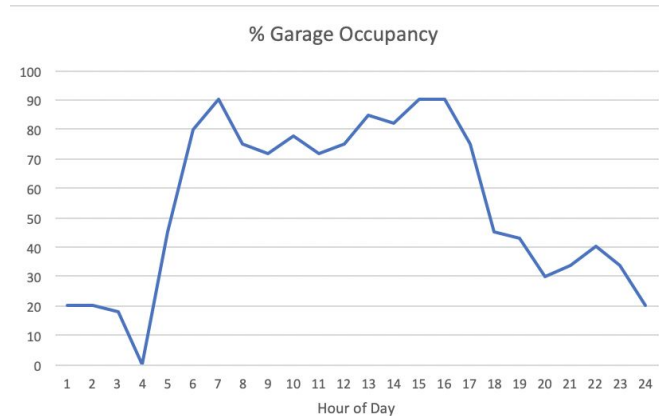
Dynamic Pricing Model

Dynamic pricing is the practice of basing the price of a product on a non-static set of factors for the purpose of maximizing revenue. In the case of a parking garage an owner is dealing with a finite resource, parking spots, from which to extract value. In this section we explore a model to increase revenue based on traffic volume and parking statistics data from Uber.

The Uber model shows strong correlation, 93%, between the volume of traffic in a major city and the associated parking density for the given time period. Because our garage and its location are theoretical we are forced to make assumptions regarding the density in our own garage. We will assume the location of our garage is in a major metropolitan area and located in a high demand zone. Our model further assumes a 90% occupancy rate during the peak weekday times and follows the Uber model for decrease in occupancy as a function of the outflow of traffic from a metro area. Therefore, we peg a 90% garage occupancy rate with the peak traffic in Uber’s data set and allow the garage’s occupancy rate to fluctuate as the traffic data fluctuates.

A study of the price elasticity of demand for parking garages shows that “When parking demand in a specific area exceeds parking supply (resulting in a parking occupancy of close to 100%)” (Lehner, Peer 4). We use this research along with our previously stated assumption to justify no change in customer demand when using a dynamic pricing model whose maximum price threshold does not exceed a factor of 1.4 times the original static price.

To avoid deterrence of customers during low demand hours, our pricing model will only “go live” when the garage parking density is projected to be greater than 60% (minimum occupancy threshold of dynamic pricing) .



Administrator defined factors:

Base Rate: base

Minimum occupancy threshold for dynamic pricing: minThresh

Maximum occupancy threshold for dynamic pricing: maxThresh

Maximum Base Rate Multiplying Factor: baseMult

Dynamic factors:

Hours of parking desired: hrsParked

Current garage occupancy percentage: currOccPercent

$$base * hrsParked * \left[1 + \left[(currOccPercent - minThresh) * \frac{1 - baseMult}{maxThresh - minThresh} \right] \right]$$

The simple linear nature of the price model is its greatest strength. This model can be easily understood by garage owners and lends itself to simple market experiments. For example, one could take any arbitrary consecutive time period, split it in two and compare revenue performance by varying administrator defined factors to influence price/revenue/demand outcomes.

Static Model Revenue

Assumed parking spots: p=100.

Traditional flat rate: \$6/hr.

Billable hours on a traditional weekday: 1313

Static Model Revenue: \$7878

Dynamic Model Revenue

Assumed parking spots: p=100

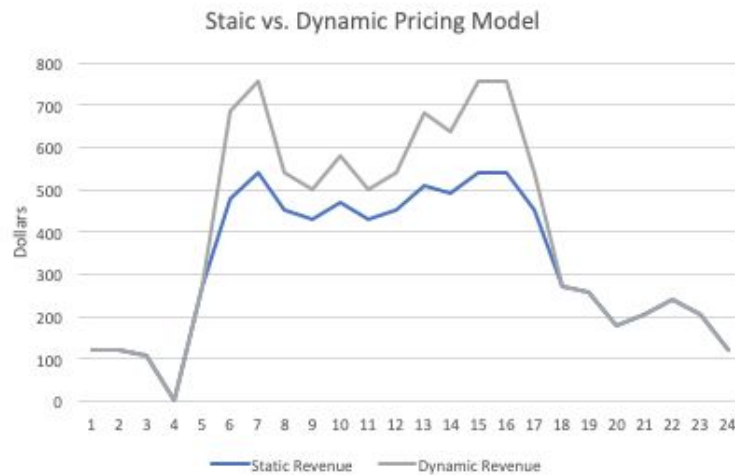
Minimum Charge Threshold: 60% Occupancy

Maximum Charge Threshold: 90% Occupancy

Maximum base rate multiplying factor: 1.4

Billable hours on a traditional weekday: 1313

Dynamic Model Revenue: \$9564.88



Project Size Estimation

For each use case specified in the report, the complexity of implementation was analyzed and was given a specific weight value. A brief description of what would need to be implemented for each use case is provided and an explanation on the complexity value chosen.

Use Case Identifier & Name	Relevant Characteristics and Steps Needed for Implementation	Complexity Description and Reasoning	Weight
UC-1: Register	The register use case would require a user interface to make an account for the parking garage system and add relevant information such as name, phone number, email, cars to register and payment information. The actors included in this use case would be the customer and the registration terminal. The customer must be able to provide their information for the new account to be registered and the registration terminal would accept the new information and save it so that the user can log in with their newly	Average In this use case, the customer would add all the information that the registration terminal asks for and upon successful adherence to the required information, the terminal would save that information to the central server. The worst case would involve the customer entering a piece or multiple pieces of information incorrectly (either incorrect format or just wrong entirely) and needing to ask the customer to re enter their information. This makes this case	10

	created credentials.	average because of both of these scenarios.	
UC-2: Login	The login use case involves the user inputting their credentials for their account in order to gain access to their reservations, account balance, and other information. The actor involved would be the customer and the participating actor would be the central server. The customer would input their email and password and upon verification through the central server, they will be granted access to their account or to make a reservation.	<u>Average</u> In the best case, the user inputs their information correctly and they will be able to access their account. In the worst case, the user inputs their email and password but either their email or password is incorrect and they will be prompted to enter in their correct credentials. If they forget their password, they need to use their email to be able to have a link to make a new password sent to them. The best case would take only 1 step to complete	10
UC-3: Edit Account	The edit account use case would require successful login from the customer and then a page that allows the customer to change elements of his account such as the payment information, email address, password, etc. If the customer inputs the new information correctly then the new information is then updated for their account. The actor would be the customer and the participating actors are the central server and the registration terminal.	<u>Average</u> The best case for this use case would be that the user successfully logs in to their account and they are able to input the correct information into the terminal and that information is saved into the server. The worst case is that the information provided is incorrect and the user must be advised to re input the information that needs to be changed. The best case would take 2 steps since the user needs to input the new info and then the new info needs to be verified.	10
UC-4: Make Reservation	This use case requires a successful login from the customer and then selecting	<u>Complex</u> This use case is given the complex weight because of	15

	<p>“Make a New Reservation” from a list of options. This brings the user to a page that requires them to pick the dates they would wish to park. Then these dates are cross checked to the database to check for availability. If these dates are available then the user is prompted to pick the times between which they would like to park. If there is no availability then the user is shown the next available days they can park. Once the user picks a time then the Central Server can confirm the reservation and then a confirmation to the customer.</p>	<p>the amount of communication and checking that it needs to go through for completion. It checks twice to the reservation system and asks the user for input in between. This transfer in communication can lead to numerous errors and user input failures. Which makes accounting for all mistakes by the user and backend very tedious.</p>	
UC-5: Update Reservation	<p>This use case requires a successful login from the customer and then selecting “Edit an Existing Reservation” from a list of options. This brings the user to a page with all of their current reservations (3 or less). They can then select one reservation, to change the date, time, or vehicle on the reservation. The Central Server can then check the availability and update the reservation and send a confirmation to the user.</p>	<p>Complex This use case is given the complex weight because of the amount of communication and checking that it needs to go through for completion. If a user decides to change the date or time of the reservation, the reservation system and central server need to be checked to see whether the user can make these adjustments. The correct transfer of communication and data is essential for the parking garage to function optimally. Additionally if a user is changing the vehicle that data has to be saved in the user's profile. Thus making this a complex use case.</p>	15

UC-6: Make Contracted Reservation	This use case requires a successful login from the customer and then selecting "Make a New Reservation" from a list of options. Then the user can select a specific option to "Make a Contracted Reservation". Then the User can pick the during they want to reserve a certain number of spots for. The Central Server will check the availability and confirm the status with the user.	<u>Complex</u> This use case is given the complex weight because of the amount of communication and checking that it needs to go through for completion. For this process to work efficiently the central server and reservation systems have to work together to make sure there are no clashes in timing between the users so users can get the spots they desire. Making sure there are no errors will be a challenging task.	15
UC-7: Pay Bill	This use case requires a successful login from the customer and then selecting "Edit my account" from a list of options. The User can then click "View my Bill ". After that the user can then pay their bill by then putting in their credit card and cvv number.	<u>Average</u> This use case is given average weight because the payment information has to be saved on the user profile. There is not as much communication between systems. However it is essential to protect this data and make sure its is processed correctly making this case an average weight.	10
UC-8: Extend Grace Period	The user must be successfully logged in and in either the "Make a Reservation" or "Edit an Existing Reservation" sections. This is where an additional option to pay extra for an extended 30 min, 1 hr or 2 hours onto their current grace period. This then updates their account bill and the Central Server with the extended	<u>Simple</u> This use case is given simple weight because the central server only needs to be checked once to see if grace period can be extended. Since payment information is already saved in the user profile this makes this user case simple.	5

	time.		
UC-9: Create Garage Layout	The manager must be logged in to access the Configuration page. The manager can then use the graphical interface to configure the layout of the garage.	<u>Complex</u> This use case requires the implementation of a graphical interface that can render a model of the garage based on user values. Thus, this is a complex actor interacting with a system that requires heavy internal processing to render the garage model.	15
UC-10: Set Pricing	The manager must be logged in to access the pricing page. They can then enter their desired values for hourly and overstay parking.	<u>Average</u> The use case requires interaction with text and numerical fields. This is an average actor interacting with a system that would require about 4-5 steps in the case that the manager changes the value.	10
UC-11: View Garage	The parking administrator must be logged in to view the garage page. They can then enter parameters for which spots they would like to access.	<u>Simple</u> The use case involves an average actor with 1-2 steps of interaction with the system. The system would then retrieve and display the data as the actor requested.	5
UC-12: Issue Rain Check	The parking administrator must be logged in to issue a raincheck to a customer. In this case, an overbooking situation must have occurred. The administrator can then issue a rain check to the affected customer.	<u>Average</u> The system would first notify the manager of the overbooking situation. In this case, the actor has average complexity as they are interacting with simple data fields. The administrator then selects the customer and issues the rain check, a process that requires 4 steps.	10

UC-13: Bill Customer	The parking administrator must be logged in to generate a bill based on the customer's activity. The administrator would select the customer and the time period for which the system generates a bill.	<u>Simple</u> The administrator, in this case an average actor, locates the customer for which they would like to generate a report. The system then retrieves and displays the data requested.	5
UC-14: View Garage Usage Statistics	The parking administrator can view garage statistics by logging in and navigating to the Statistics tab. They can enter parameters for the system to query the database and present visually.	<u>Average</u> The use case is initiated by a parking administrator that interacts with basic data fields. The system then retrieves the data and displays the information in the format requested. This process would be greater than 4 steps depending on the parameters selected.	10
UC-15: Scan License Plate	The license plate scan use case would require no user interface as the scanner would cross check the license plate with the database. Two actors are involved: the customer reservation system and the central database. The former is to determine if the license plate has a reservation, while the latter checks to see if the license plate is in the database in the first place. To implement, only a check between the database and the customer reservation system needs to be made.	<u>Average</u> In the best case for success, the amount of steps for the scan license plate would take around three steps (scan the license plate, check the database, check the reservations). In the worst case, this would also take three steps: the license plate is scanned, the license plate is in the system but no reservation was made or the license plate. The best case for success is two steps: The license plate is scanned and does not exist in the database.	10
UC-16: Smart Assistant Support	The system would require the login information to be shared with the smart assistant. The user could	<u>Complex</u> This use case requires interaction between multiple actors. The user	15

	then use the voice assistant to create a reservation.	initiates the use case by asking the voice assistant to make a reservation. The voice assistant would then need to interface with the Reservation system.	
UC-17: Elevator Terminal Output	This will be the UI in the elevator. The initiating actor being the elevator terminal, will have access to the databases which belong to the participating actors. It will take the user, another participating actor, get their license plate and check the spot assigned to that license plate. If there is no spt assigned to the license plate, then the system will treat the user as a guest and check the entire database for any vacant spots. If the user is registered to a parking spot, the system will check if their spot is vacant. If it is not vacant, the system will issue a raincheck to the user. If there are vacant spots, no matter the scenario the user will then go on to park. All of this information will be displayed in the UI.	Average The amount of work that needs to be done varies depending on the scenario. In the best case, the system will simply find the user's assigned spot and make sure that it is vacant. In the worst case, the system will have found the assigned spot to be taken, and additionally will have scanned through the lot and found no vacant spaces. Whatever the case is, the system will then have to output this information as a UI. It will therefore need to access the database. This part is not very complex, it is a simple get data method.	10
UC-18: Update Parking Spot Status	When the participating actor, the user, parks in a vacant spt, the status of the spot goes from vacant to occupied in the database. When the user exits the parking spot, the status of the parking spot changes from occupied back to vacant.	Simple This is a very linear program. When a user parks in a vacant spot, it's status goes from vacant to occupied. When a user leaves, its status goes from occupied to vacant. There is not very much thought and detail that needs to go into this.	5

UC-19: Park	<p>The user will enter the garage. The user's license plate will be detected with UC-15, and it will be ran in the system in UC-17 to determine if the license plate is registered or not. After the steps in UC-17, the user will either park, or leave either with or without a raincheck. If the user parts, UC-18 will be invoked.</p>	<p><u>Average</u> This has to take into account many of the same scenarios as UC -17. Additionally, this will detect when a user has entered the garage, which will require UC-15. if it is registered in the system. This adds more complexity, because not only does this need to simulate a camera, but there will also need to be checks run in the database to verify if the license plate is actually in the system. In the UC-17 cases where there are vacant spaces in the parking lot, UC-18 will then be implemented.</p>	10
-------------	--	---	----

Project Management by Subgroups

Customer Registration Subgroup

- Neha Nelson
- Brian Ogbemor
- Param Patel

Within our subgroup, we divided the tasks of this report equally amongst ourselves, with writing out the requirements, ranking their priority and defining our glossary terms. We also assisted in writing the customer problem statement by providing statistics and cross-checking the sources. We also helped in the overall development of the report by completing the outline, formatting the document, and proof-reading and reviewing the submissions from the other subgroups.

At our first meeting, we began to earlier define all of our requirements from the project descriptions and we learned of some overlap in the defined requirements and communicated these to the other subgroups. We organized a Discord call, which is the voice and text chat app that we use to communicate, and learned of other requirements that the other subgroups needed from us. By doing this we were able to draw up a flowchart of the potential interfaces that we would need as a group. Using this flowchart and reading the project description gave us insight on how to collaborate and understand the requirements of how our group fits in with the other subteams.

Along with the complete group discord chat, the subgroup also has a separate channel, or sub text thread, that we use to discuss smaller scale decisions about our system. We plan to meet every week for an overview of the completed work as well as a time to pitch suggestions to make our work easier. This week after looking over the report requirements we were able to help outline the document, set up tables for the requirements and create the identifiers, which allowed for a clearer distinction on the type of requirement and what subteam would be implementing it. Overall, we found that working in subgroups is very effective for the group size and tasks designated. It will allow us to discuss amongst ourselves and decide whether we need the help of an additional subteam before bringing a problem to share with the others.

Within our subgroup, we divided the tasks of this report equally amongst ourselves, with arranging the use cases and cutting them down. We also assisted in making the traceability matrix that contained the cross-section of the requirements and use case. We also helped in the overall development of the report by completing the outline, formatting the document, and proof-reading and reviewing the submissions from the other subgroups.

After drafting up some use cases, we realized that there would be many areas of overlap in our project, namely that one use case can test for a multitude of requirements. We decided to implement the most important ones for the first round of demo because they cover many requirements and ranked higher in priority weights than the other ones listed. This was done through the previously mentioned Discord Chat that we are all a part of. It allows us to easily access other subgroups' chats and them to drop in on ours for input. We used this to rank the requirements and use cases to fully dress up within each team's requirement list. We also delegated the fully dressed up use cases writeups and UI designs between all of us.

Lastly, the UI designs were a culmination of all three groups. It was decided that the colors would remain the same as our original logo and that the subteams would have access over what their subsystem could look like as long as the themes were cohesive enough to pass as one set. We were allowed to give and get input from the other teams for our design as well as theirs. This lets us all work independently yet efficiently.

Managerial / Administrative Subgroup

- Jeffrey Samson
- Aniqah Rahim
- Swetha Angara

The manager group has tasked itself with the integration of the relational financial databases with the administration's front-end interface. With the support of the elevator group, we will aid in the software debugging that comes with integrating back-end code with SQL. This interface the group plans to build will allow the managers to configure the parking garage map, to observe how the current garage occupancy is growing over time, and to analyze various business statistics displayed in either a graphical or spreadsheet format. We will perform multiple tests on the system to ensure that the manager access portal and the financial data banks are both secure and reliable.

There is a team Google Drive where all of our project reports are uploaded in specific folders. There will be a separate Manager folder for any code or documents a member of this group creates. There is also a GitHub repository where every group's code is uploaded for all other team members to see and access. With this, we have set up a system such that once a person in *any subgroup* is done with their part, or is struggling, that person uploads the code to the GitHub repository and can ask for help in the voice/messaging Discord server. Any available group members will try to assist said person with their portion of the project. There is also a line of communication between the project team members in the messaging app GroupMe.

In addition to weekly team meetings over Discord voice-chat, the manager group has organized a weekly, in-person group meeting. This is where we divide the subgroup's responsibilities equally, not only so that everyone can contribute to the project, but also so that each sub-group member can work on the tasks that most closely align to his/her interests and proficiencies. It is also a time to express concerns about the project, and to share any coding breakthroughs, project ideas, or to suggest a sub-group collaboration on a certain part of the project. We have no sub-group nor project team leader, so every big decision is put to a majority vote.

We have divided tasks evenly so that each of us can work concurrently on the document. We ensure that everyone is aware of their tasks by communicating on Discord. Together, we decided which use cases were the most important and required dressed descriptions and UI specifications.

We have started learning React, node.js, and Express.js to start creating our website and preparing for the first demo. After collaboration with the other teams, we will decide which features should be implemented by the first demo.

Elevator Operation Subgroup

- Disha Bailoor
- Thomas Murphy
- Charles Owen
- Nicholas Meegan

The elevator group will be responsible for a combination of front end and back end interfacing. The elevator's front end interface will be responsible for a variety of tasks such as notifying customers regarding what spot they will be parking in, providing information about how full the parking garage is, and allowing the user to enter their login information when their license plate is scanned, to name a few. On the back end of the interface, the elevator will be responsible for keeping track of vacant and occupied parking spots, updating pricing when overstay occurs, and controlling the database, among other responsibilities. We will work closely with the manager and customer groups to provide necessary information from the database to ensure each group's success. Each member of the elevator subgroup will work on the database, using SQLite, as well as other necessary JavaScript code for the elevator to function properly.

The elevator group will have a user interface similar to the customer group's since some functions such as registration and user notification will be shared by the elevator and customer user interface. The architectural requirements for the elevator group will be different from the customer group; the elevator user interface (UI) will be a touchscreen on the inside panel of the elevator while the customer group's UI will be implemented through a smartphone or personal computer. The elevator group must figure out the best way to create a simple user interface that conveys all of the data from the other groups; intergroup communication is critical. The GitHub repository can be accessed by all members of the project in order to effectively stay consistent with each other. Project members will make commits through Git Bash only after they are sure that what they are adding is consistent with the other groups.

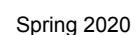
Weekly meetings are organized for the elevator group, in addition to larger meetings with the rest of the teams every other week. The meetings within the elevator group are primarily conducted in person, however a voice call over Discord may also be made. When meetings take place between all of the groups, a voice call over Discord is made. The information that will be conveyed to the user via the elevator will come from both the customer and the administrator, so it is essential that we stay in frequent communication with them. Additionally, we will have two subgroups within this group. One group will focus on the parking garage camera, keeping track of the license plate numbers that enter and exit the garage. The other group will deal with the occupancy/vacancy statistics of the elevator, as well as keep track of which license plate/customer has entered or exited the elevator.

Within the group we divided the tasks equally across the board. We made a checklist for the use cases, the graphs and the descriptions that we needed to do as part of the elevator subgroup and assigned all tasks uniformly across the table. We have been communicating through Discord with other sub teams to talk about the nitty gritty of the project and to ensure cohesiveness.

We are currently in the process of learning how to utilize Express.js, node.js and React JS for the project as a whole. We as a team have decided to use these two to primarily code and implement our project, therefore all team members have started working on gaining expertise in the same.

We as the elevator group have had to figure out the different cases of how the consumer would come to take the elevator. We have had to consider the people who are allowed to use the elevator and others that are not allowed to take the elevator for example the walk in customers. We talked about a user friendly UI that would be handy and that would convey clear and adequate information to the consumer inside the elevator.

The functionality of the elevator is also being discussed as the project is taking its shape. We will need to figure out how we will implement the physical presence of the elevator. We will either treat the elevator as an individual object, or make the elevator out of several different objects that serve a certain function. Either way, the elevator will have boolean statements and vehicle status reports associated with its software implementation. This will perform tasks such as confirming the presence of a vehicle in the elevator, which license plate has entered or exited, and whether or not the vehicle detected is in the system.



Elevator Operation Group

The elevator group will work on two facets of the project at once. The work will be divided into two; one part being the administrator portal and interactive garage layout, the other being the GUI and operation of the elevator business policies.

We anticipate a significant learning period associated with the dynamic layout interface and will plan our time accordingly. Since the elevator operation largely involves interaction with a user interface, all members of the group must gain an understanding of React.js. The first group will need to focus on several of the use cases that involve the administrator's databases as participating actors. The latter group will focus more on integrating the dynamic price model into our UI. This will require a very solid knowledge of React.js. We will definitely need time to learn React while also balancing writing weekly reports. The elevator group will continue to have discord calls, discussing progress made on the code written by each teammate and what portion of the elevator control to focus on next.

Since the elevator control is quite extensive, the elevator subgroup will be broken up further into two sub-teams in the short term: one sub-team working on the demonstration aspect of the project and the other focusing on the elevator interface. The focus for the demonstration group will be simulating the parking process: having a fictitious car drive up to the elevator and waiting for the interaction to proceed. The primary focus in the short term for the elevator interface group will be to have a basic elevator interface working properly. This includes scanning the customer's license plate and communicating with the customer reservation system to check if the license plate has a reservation, displaying the spot to park in, and updating the parking spot status accordingly. The basic implementations of these will be completed in the short term, with further refinements of these operations throughout the entirety of the project's lifespan.

In the long term the elevator operations subgroup will focus on adding in various novel thoughts and ideas into the project. The most extensive task the elevator group will be faced with is implementing the drag and drop interface for the creation of the garage. This will require learning and understanding the interactive.js or React DnD library and determining how limits will be set on garage creation. Lastly, the elevator group will focus on implementing the ability for the elevator group to give rain checks to customers when the garage is full, connecting everything to a database to store all of the necessary information and setting up the interaction between the other subgroups. Essentially, the short term and long term goals of the project can be broken down as follows below:

1. Short Term Goals

- a. Complete the parking simulation, where a vehicle with a specified license plate will "drive" up to the elevator terminal and an interaction between the two will occur.
- b. Design the elevator control terminal, consisting of scanning the license plate of the user and informing the user where to park.
- c. Connect the elevator control terminal to the central server, customer reservation database and login interface to complete the elevator terminal's operations.

3. Long Term Goals

- Prepare initial interfaces for Demo after Spring Break
- Determine how to implement the drag and drop garage customization interface.
- Implement the ability for the user to obtain rain checks when the garage is full.
- Implement premium and handicapped parking spots.
- Set up interaction between the two other subgroups.

					23-Feb-20							1-Mar-20							8-Mar-20							22-Mar-20							29-Mar-20							5-Apr-20							12-Apr-20						
	Task Name	Duration	Start	ETA	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S							
1	Complete Execution of Project	~	2/2/2020	5/7/2020																																																	
2	Engineering	~	2/2/2020	5/7/2020																																																	
3	Complete the Parking Simulation	7 Days	2/24/2020	3/1/2020																																																	
4	Design the Elevator Control Terminal	7 Days	3/2/2020	3/8/2020																																																	
5	Connect Elevator Terminal to Central Server	6 Days	3/9/2020	3/14/2020																																																	
6	Prepare Initial Interfaces for Demo	7 Days	3/22/2020	3/29/2020																																																	
7	Implement Drag/Drop Garage Customization Interface	7 Days	3/30/2020	4/5/2020																																																	
8	Implement User Rainchecks	5 Days	4/5/2020	4/10/2020																																																	
9	Implement Premium and Handicapped Spots	5 Days	4/10/2020	4/15/2020																																																	
10	Set Up Interaction with other Subgroups	5 Days	4/15/2020	~																																																	

Product Ownership (from Proposal)

Member	Role Played in Project/In Sub-Group
Disha Bailoor	<p>I am a member of the Elevator Sub-Group. I will be assisting in developing the back-end of the Parking Garage Application. We will mostly be working with Java/JavaScript in collaboration with the code written by the Customer and Manager Sub-Group. I plan to also assist with error handling and incorporating that in my code. I can also automate testing for the project in Python if needed. As I develop my SQL database skills in my CS class this semester, I will be applying that to code and debug our database side portion of the project. Lastly, I will be working with other teams offering and receiving constructive feedback on how to improve my work.</p> <p>We have decided to code in Node JS and React JS for the backend and the frontend of the project. I have downloaded them both and have started playing around with learning JavaScript so we can start implementing our ideas for this project. Because I have all the members' emails, I usually am the one who sends out emails for the group and takes notes on the concerns of the group and drafts the emails to the Professor and the TA. If there are questions and concerns regarding the group work and flow, we usually work together, have discord calls to figure them out. I have helped out in proofreading and working on the Used Cases, and other descriptions for the project. I will also be working on the backend to improve the app/website that we will be making for the project.</p>
Neha Nelson	<p>I am a member of the Customer Group. I will assist in developing front-end JavaScript/HTML/CSS code for an application that allows customers to create an account and input their information. I believe my knowledge with React can help build the front-end section and my skills in C++ to help the back-end as well. I have also used Google APIs to help create the software that can help find nearby garages and potentially also to find directions to the garage in question.</p> <p>I have installed React and am working on learning more about bootstrap and SQLite. So far our subgroup has kept up with the reports and communicates daily through discord if any questions arise.</p>
Param Patel	<p>As a member of the Customer Sub-group, I can be an asset for my C/C++ background for backend and complement it with the skills that I will learn over the course of the semester in Java and SQL. I can also contribute to the interface end with HTML and other design components in graphics software like Illustrator and Muse/Dreamweaver.</p>

	<p><i>I have installed React and Bootstrap and started the opening webpage for the website. By using React we can ensure that the website can also be mobile ready. I will continue to work on the Login and Sign Up pages and link them to other pages made by my team through the Navigation Bar. For the future, I will continue to further my knowledge in React, Bootstrap and SQLite and their interactions to add to our project. Our subgroup has kept up with the reports and coding tasks by delegating tasks and messaging each other with both compliments and criticism.</i></p>
Swetha Angara	<p>I am a member of the Manager Group. I have started working on the front end of the management portal using React and Bootstrap. So far, I have worked on creating the navigation bar and adding routes for accessing the different portions. For the back-end, we have decided to use node.js and Express.js. For the rest of the semester, I will work on using these technologies to interact with our database. I will improve my SQL/Database skills to support the transactions such as viewing customer profiles, monthly contracts, as well as payments. I will also assist the team in learning the skills necessary to complete the additional features such as Smart Assistant Support.</p>
Nicholas Meegan	<p>I am a member of the Elevator Group. The Elevator Group consists of mostly back-end code, such as determining if spots are occupied or not, reading the license plates of vehicles, and controlling the elevator operations, with a simple user interface at the front. This will be used as a way for the customer to communicate with the system in the garage and determine where they will be parking. These functionalities will be implemented through React, Express.js and Node.js. Some of this information is passed over to the customer and manager interfaces. Regarding the sub-group, the role I will likely play is working on the back-end code for the elevator. and some html for the web page. I will also help design some of the interface the users see using Photoshop if necessary. Additionally, as I learn more about databases and SQL through another course I am taking this semester, I hope to assist my group in developing a fast and efficient database to store all of the data necessary. When focusing on the long term goals, I will help implement the drag and drop garage customization feature using interactive.js. This will then be integrated with the management group and will serve as the basis for the garage layout and determining which spots are occupied. I will be in close communication with the manager group when implementing this feature. As of writing this report, I have downloaded both React and Node.js. I am beginning to program the elevator screens and the scripting which transfers between the screens of the elevator</p>

	terminal. In the reports, I have focused mostly on writing the use cases and the concepts for the elevator group.
Thomas Murphy	As a member of the elevator group with knowledge of C++ and Java, I will use my knowledge of object-oriented programming to help create a realistic simulation of an elevator. This includes the properties of the elevator, its interface, and whether or not a vehicle is present or not. On the database side of things, I also have experience with SQL which can help. I will help figure out an efficient way to organize the customer's data given the information we have about their vehicle registration and parking confirmation number. I am also willing and able to learn anything I need to about JSP and web servers in order to complete the project.
Charles Owen	As a member of the garage team, my focus will be on the Entry and Exit cameras and their interaction with the customer database. I will ensure that the vehicle simulation-GUI works seamlessly to start the parking garage chain of logical events. I will develop and communicate the API requirements of my sub-system to the rest of the team. I will also take ownership of the dynamic pricing model. Our model will track real-time parking availability in the garage. I will be responsible for developing the algorithm to support this scheme. The algorithm will have to accept input in the form of parking availability and respond with an appropriate price based on availability.
Jeffrey Samson	I am a member of the Manager Group. I'll be contributing to develop front-end code that will enable the owners of parking garages to upload the schematics of their parking garage into the app for customers to reserve parking spots and for the owners to manage and observe how their business is going. Additionally, we'll be working closely with the Elevator and Customer groups to develop a convenient payment system and receive the correct parameters and details about a customer's car including license plate number, make and model etc.
Aniqa Rahim	I am a member of the Manager Group. I have been an equal contributor to the writing of the first report. I have attended and plan to attend all of the weekly team and subgroup meetings. I am currently learning Javascript through React, Express.js, and Node.js with the aim of making sure the SmartPark website provides a satisfactory user experience. The different pages of the website will be divided amongst the Manager Subgroup so that we have something to prepare for the future project demo. I also plan to use my newly acquired SQL knowledge that I will

	gain throughout this course, for the backend production of relational databases for the garage administration's calendar system. I will also create the administration's user interface for the calendar system.
Brian Ogbebor	I am a member of the Customer Group and so far I have provided information on the use cases for the group in a few key areas such as the Project Size Estimation table in this report and Fully Dressed Descriptions in Report 2. I am also learning React so I can fully help Neha and Param in making a captivating website for customers to use. The interface will be easy to navigate and cater to those who use technology all the time and even those that do not. In addition to that, my knowledge of SQL may come in handy since we need to save customer's information in a database or central server for other subsystems to use when they need to access it. My group members and I discuss what needs to be done on a daily basis and this aids to how we complete our desired sub assignments on time and in an orderly fashion.

References

Customer Problem Statement

- McCoy, USA Today, [“Drivers spend an average of 17 hours a year searching for parking spots”](#)
- Alexa Skill, [Who is Amazon Alexa and what is an Alexa skill?](#)
- Pedestrian Injuries in Parking Garages, [Pedestrian Injuries in Parking Lots and Garages: Get Fair Compensation](#)
- IBISWorld, [“https://my.ibisworld.com/us/en/industry/81293/industry-at-a-glance”](https://my.ibisworld.com/us/en/industry/81293/industry-at-a-glance)
- Reinventing Parking, reinventingparking.org/2013/10/is-30-of-traffic-actually-searching-for.html
- Project Description, [Software Engineering Course Project Parking Garage/Lot](#)
- Pedestrian Safety, [Pedestrian Safety | Motor Vehicle Safety](#)
- Pedestrian Statistics, [2017 Data: Pedestrians](#)

Mathematical Model

- Nature.com - [City-scale car traffic and parking density maps from Uber Movement travel time data](#)
- Four Week MBA - [Dynamic Pricing: Is The Price Tag Legacy Coming To An End?](#)
- Price Elasticity of Parking (PDF) [The price elasticity of parking: A meta-analysis](#)

Plan of Work

- On the map: product roadmap templates and tips - <https://blog.asana.com/2018/08/product-roadmap-tips-templates/>
- Gantt Chart - <http://www.ganttchart.com/>
- Asana.com - <https://asana.com/resources/gantt-chart-basics>
- Charts and Diagram Templates - draw.io