Rutgers University

Auto Park

Parking Garage Automation

Group 3
Final Report

Author:	*Email:*
Chunhua Deng	chunhua.deng@rutgers.edu
Corey Chen	cc1437@scarletmail.rutgers.edu
Jonathan Garner	jonathan.garner@rutgers.edu
Ridhima Sakhuja	rs1425@rutgers.edu
Siddharth Musale	siddharth.musale@rutgers.edu
Siyu Liao	siyu.liao@rutgers.edu
Xianglong Feng	xianglongchunyi@gmail.com

December 9, 2018

Contents

List of Figures

1 Summary of Changes

- Dynamic update method for single price-based demand function
- 5-Modules based dynamic pricing algorithm
- Ease out quart function for price-based demand function in peak time
- Ease out quart function for price-based demand function in adjustable time
- New demand function optimization base on time and price
- Final benefits optimization.
- Included the transition matrix in Section 9 that was missing from Report 2
- Wrote explanations about the fields required for each class in Section 9.3
- Elaborated more on the component-based architectural style in Section 10.1
- Updated the UI images in Section 12

2 Customer Statement of Requirements

2.1 Problem Statement

The majority of modern parking garages have inherent flaws that cause them to be less efficient than what they could be. A recent study indicates that drivers spend an average of 17 hours per year looking for parking and approximately $97 per driver1. This situation is only exacerbated during peak hours, when a high influx of cars causes congestion in parking garages, resulting in more time and money wasted on looking for parking. Many drivers are also unsatisfied with the flat rate pricing system, where they are charged a fixed hourly rate regardless of the time of day. Additionally, a majority of parking garages do not have a system that lets their customer keeps track of how long their car has been parked for and how much they will have to pay when they leave. Along with this, parking garages also lack a security system that helps ensure the safety of the customers' cars. In metropolitan cities, parking garages are often located in unsafe areas, where anyone can enter and leave the garage regardless of whether they are a paying customer of not. This smart-garage proposal contains ideas to improve customer experience of the parking garage, save wasted time in the garage by looking for empty spots, and boost revenue for the garage.

2.2 Proposed Solution

In order to provide a more streamlined process for our customers, our plan is to implement an automated system that will allow customers to view and reserve available spots using a visual interface hosted on a website. This will give customers a simplified view of available parking spaces that is easily accessible on both computers and mobile devices, and allow them to reserve or change spots at any time. Customers will be able to reserve spots for a specified check in time and reservations for an occupied spot will open once the parked patron leaves the spot. There will be long term term parking as well as short term reservations which will allow customers with a variety of needs to utilize the parking garage.

Our proposed system includes multiple novel ideas that have not yet been implemented in previous automated parking garage projects. The first of these is a visual display on the website that will allow customers to view and select spots on a simplified map in order to check which spots are currently

open or to place reservations. This feature will primarily be available to customers who come in off the street for ad-hoc parking through the use of a screen at the check-in area of the garage that will display available spots and allow customers to select the spot they want. Another new idea is our smart pricing system, which will continuously update the cost of spots based on multiple factors (time of day, amount of reservations, etc). Along with these, we are proposing a traffic monitoring system that keeps track of how many cars are driving to their spots on each level of the garage and limits the number of active cars per floor in order to cut down on traffic within the garage. Another newly proposed feature is an improved security system that will limit garage access to parked customers only in an effort to increase the feeling of safety within our customers. A final feature we are proposing is the use of a unique alphanumeric security codes that customers receive either via email or SMS in order to enter or exit the garage. This guarantees that only customers with security codes can re-enter the building, protecting the customers' vehicles from non-customers. All the features mentioned above are further elaborated on down below.

Due to the fact that ad-hoc parking is a major component of a parking garage's income, there will be a limited number of reservable spots. These spots will be separated by floor or section in the garage. This means that even if reservations are fully booked, patrons will be able to still drive up and park if there are ad-hoc spaces available. The price of parking in these spots will be determined by the smart pricing system (explained more below) which will generate more revenue.

Another major component of streamlining the parking garage is reducing traffic inside and allowing customers to easily find their spot and park. When a customer enters the garage for ad-hoc parking, they will be shown the same display that is available on the website via a monitor and prompted to choose an available spot using the visual interface. This system will display the relative location of the selected spot on the visual interface, which will cut down on the time it takes for drivers to locate their spot. To further cut down on traffic in the garage, drivers will have to get to their assigned floor using a 1-car sized elevator. This elevator will scan the license plates of each car that uses it and communicate with a system that will keep track of which cars are currently on which floors, and whether or not they are parked. Each spot also contains a sensor which can tell whether or not a car is occupying the spot, and report that information back in order to tell whether or not the car that is active on that floor has parked yet. As soon as it is determined

that the car on a specific floor has parked, the next car will be allowed in to locate their spot and park. This system aims to improve a customer's ability to find their spot and park in it.

This proposal also includes a smart-pricing system that entices more customers to use the garage. This pricing system we propose includes incentives for customers to register for membership of the garage by offering special deals such as reduced fees for longer term members. This pricing system dynamically adjusts the hourly rate of parking so that the price of parking spots goes up as the demand for them grows and diminishes as the parking garage is utilized less. This is intended to help make more profit during rush hours as well as maximizing customers during the other times of the day. However, we also offer the fixed rate strategy for someone preferring traditional parking billings. We can give fee estimations and save money for customers by helping them choosing fixed rate or dynamic rate.

Our proposed system will also consist of a notification system that will regularly update the customer regarding their status via text message or email. Upon arrival, once the driver has parked their car at the designated parking lot, this system will send a message confirming that they have parked in the correct location and a ticket that will have a confirmation code that will be required to leave and enter the parking garage. The ticket will also include information such as the driver's arrival time and the parking lot number. After every hour, the notification system will send a text message to the customer, letting them know how long it has been since they have left their car at the parking garage. Not only will this allow customers to keep track of their parking spot, but it also be useful in trying to ensure that drivers with reservations do not overstay their allotted time.

Another concern of many customers is the security of their car when parked inside a garage. In an effort to cut down on potential crime or theft in the garage, a system will be put into place that will only allow verified customers in and out of the garage on foot. When a customer parks their car, they will be sent a notification through the previously mentioned system containing a unique confirmation code. In order to unlock the pedestrian doors and enter or exit the garage, users will be required to enter their access code. This system aims to enforce a feeling of safety and security in consumers. The main goal of this proposed project is to optimize our current parking garage system and turn it into an automated, web-based system. Doing so will allow us to cut down on many operating costs, such as payroll for check-in area attendants who will no longer be of use. This

system will also allow us to view and change current reservations, as well as adjust the pricing of spots manually in order to capitalize on rush hours and bring in more revenue. However, these features will strictly be available to administrative users. The typical customers who will be parking in the automated garage will fall under two general categories: those who plan to park long term (consumers who may be on vacations, business meetings, etc.) and those who plan to park short term (visitors in the city, commuters, etc). The customers we are targeting are those who are willing to possibly pay a higher price for parking for the added convenience of being able to view and reserve spots online. Many commuters in busy cities would be willing to pay higher premiums for the security, efficiency, and convenience our automation provides.

In terms of metrics that will measure the success of the automated garages, we will be looking mostly at revenue brought in by the smart pricing system and the average number of spots being used. In addition to these quantitative metrics, we will also track the number of users who register for membership on the website and how many users use the website reservation system in order to show whether or not customers are taking advantage of the available features.

There are three overarching goals that this system is trying to achieve. Firstly, there will be a website that interacts with the database in order to provide services to members and non-members. This will include reservation services and deals for members of the garage as well as a user interface where ad-hoc customers can select preferable spots to park. In addition, this system will attempt to maximize profits through a dynamic, smart-pricing algorithm based on the occupancy percentage of the garage and time of day. Lastly, the system will also introduce another algorithm that minimizes traffic blockage through the whole garage by bottle-necking the number of cars that are searching for parking spaces on each floor.

Functional Features (Summary) - Users will be able to:

1. View and reserve spots online
2. Select spots for ad-hoc parking
3. Locate their spots using a visual interface
4. Cancel reservations or switch spots at will

5. Receive notifications such as warnings based on time spent parking or confirmation of parking in the correct spot
6. Enter or exit the building using a unique confirmation code

2.3 Novelty

We investigated the report of previous works including the one in Spring 2012 and the one in Spring 2013. To have a clear overview of those works and to distinguish ours contribution from existing works, we will give a summarization of those works and compare with them. Following are features listed in the report of Spring 2012:

1. Place online reservation in the form of a website
2. Website Registration
3. Security guard helps customer pay bills and exit the garage
4. Notify user that they are not identified
5. Notify operator if sensors not working
6. Park with rented or borrowed vehicles
7. Garage remodeled such that an elevator can lift vehicles to different decks
8. One-way entering and existing system to avoid congestions
9. camera based license plate recognition software to track vehicles as they enter and exit the garage
10. the license plate must be recognizable for elevator to work
11. Allow walk-in customers
12. Support confirmed reservation (no credit card) and guaranteed reservation (with credit card)
13. 1 hour grace period for customers not showing up, after which spots will be unreserved

14. Floor sensors for detecting occupancy
15. Elevator will lift up vehicles to the correct spots

Besides, following are the features listed in the report of 2013.

1. It has an online reservation system.
2. It implements a license plate reader reader.
3. It allows customers to create accounts to store personal information for fast parking
4. The system has three types of account, which is customer account, manager account, and employee account
5. It can notify potential customers if there are available parkings spots and for how long they can park
6. It broughts cars to a checkpoint and take pictures of the car before it enters the garage and right as its about to leave for security reason.
7. It can automatically detect frequent customers by reading license plates.
8. It integrates a simulator to test the system
9. It deals with overstate problem.
10. It encrypts its user data
11. It backs up its account information, parking data and daily reports once a day

When compared with existing works, the main novelty of this project are listed as follows:

1. Smart Pricing System: We charge users accordingly in order to attract users and provide for their convenience. For example, we charge higher when less available spots and charge lower when more available sports. **We use both models from existing published paper [2] and propose our model according to the real-world data from Seattle government.**

2. Spot viewing interface: This interface is primarily meant for the ad-hoc customers of the garage. Instead of just assigning spots to a customer, customers will be able to view an interface and select spots based on their personal preference instead of getting forced into a spot they may or may not want.

3. Different Congestion Control: Our system will work in a way such that there will only be one vehicle parking in to its spot concurrently on each floor. This reduces the amount of traffic on each floor and helps us to ensure that people occupy their specified parking spaces.

4. Automatic Security: After customers park in their specified spots, they will be sent an alphanumeric code via email or SMS that allows them to access the parking garage. The code can be used to both enter and exit the garage. This guarantees that only customers and people travelling with customers can go in and out of the garage, guaranteeing the security of their vehicles at least against non-customers.

3 Glossary of Terms

- Account - Holds all of a member's information, such as username and contact information

- Ad-Hoc Customers - Customers who drive in off the street for unplanned or unreserved parking

- Administrative User - A high-level user who will have permissions on the website to log-in and view or change management information such as reservations, pricing, etc.

- Check-In Area - The first area customers encounter upon entry into the garage with gates for entry

- Confirmation Code - Code sent to verified customers after parking to allow exit and reentry into the parking garage

- Customer - Person who enters the garage with the intent of parking in a spot (with or without reservation)

- Drivers - Customers in cars that are currently parking or driving in the parking garage
- Elevator - Platform used to raise cars to upper floors
- License Plate Scanner - Device which reads the license plate of a car waiting to enter the elevator
- Member - Customer who has registered on the website and is now a member with login information Notification - Alert or message sent through text or email to a customer
- Occupied Spot - Parking spot which currently has a car parked in it
- Parking Spot Number - Unique number which identifies a spot and its relative location Pedestrian Doors - Allow foot traffic in and out of the parking garage; can only be unlocked and used with a valid confirmation code
- Registration - The process by which a customer becomes a member
- Reservation - An agreement that the garage will hold a spot for a customer who intends to park there
- Reserved Spot - Spot which a customer has reserved with the intention of parking in it
- Sensor - Device in each parking spot that determines the presence of a car in that spot
- Smart Pricing System - Automatically determines the prices of reservations and ad-hoc parking spaces based on an algorithm and relevant information
- Ticket - Document containing a customer's parking and reservation information
- Traffic Control System - Keeps track of which floor has active drivers, and the spots in which these drivers belong in order to confirm that customers park in the correct location

- Username - Unique identifier created by a customer for themselves during the registration process
- Vacant Spot - A spot not currently occupied by a car and not currently reserved
- Verified Customer - Customer who has parked their car in the correct spot, as determined by the sensors and traffic control system
- Visual Interface - Graphical interface in the form of a simple map that displays the location of parking spots and whether or not they are occupied
- Website - Used by administrative users to view information and by customers to view spots and make reservations

4 System Requirement

4.1 Enumerated Functional Requirements

Identifier	**P.W.**	**Description**
REQ-01	3	The system will scan license plates
REQ-02	3	The system will use the license plate number to recognize customers
REQ-03	5	The system will show a real-time, visual display of the number of vacant, occupied, and reserved spots in the parking garage.
REQ-04	5	The system will keep track of the number of occupied and vacant spots in the garage at all times.
REQ-05	4	The system will allow customers to create an account on the website and register with their information.
REQ-06	4	The system will allow members to make reservations for a specified time block at the parking garage.
REQ-07	5	The system will allow ad-hoc customers to choose their parking spot.
REQ-08	5	The system will implement a smart pricing system to charge the customers accordingly.

REQ-09	4	The system will send a notification to the customer once the customer has parked to letting them know if they have parked in the correct spot
REQ-10	4	The system will also send a ticket to the customer with the customer's parking time, parking spot number, and an alphanumeric confirmation code.
REQ-11	3	The system will regularly notify the customers of how much time has elapsed since they parked in the garage.
REQ-12	3	The system will allow customers to extend their stay if there is enough space in the parking spot.
REQ-13	4	The system will allow customers with reservations to pay for their extensions via mobile
REQ-14	2	The system will allow employees to manually input information in case of system failure
REQ-15	4	The system will check for reservations via the license plate number
REQ-16	5	The system will allow customers to input the confirmation code when they come back to the garage in order to gain entry to their car
REQ-17	3	The system will not allow any other car to enter the floor if there is a car driving on that floor.
REQ-18	5	The system will send a notification via text or email confirming a registered customer's reservation.
REQ-19	2	The system will not allow more than 50 reservations to be made a time.
REQ-20	2	The system will charge the customers cancel fee if they cancel within one hour
REQ-21	5	The system charge according to different time and parking demand
REQ-22	4	The system list details of pricing for different time slots

4.2 Enumerated Non-Functional Requirements

Identifier	**P.W.**	**Description**

REQ-23	3	An elevator system will be used to take customers up to their designated floor. The elevator will only allow the customer to get on the designated floor if there is no other car driving around on that floor.
REQ-24	5	In order to register, the customer will have to give their first name, last name, license plate number, a cell phone number, email address and credit card information.
REQ-25	4	Customers will also need to create a password for their account.
REQ-26	4	Customers will need to input an username and password to login and access a preexisting account.
REQ-27	4	Members will be allowed to update their account information and add an unlimited number of credit cards and license plates to the account.
REQ-28	5	Sensors will be placed on each floor to ensure that the customers have parked in their designated spot.
REQ-29	3	The confirmation code customers receive once they have parked will allow them to enter and exit the parking garage to ensure safety of the cars.
REQ-30	4	Members with reservations may make changes to their reservations up to 12 hours prior to their reserved time.
REQ-31	4	Each time a customer makes a change to their reservation, a confirmation email should be sent out to them via their preferred method of communication.
REQ-32	4	During registration, the customers should be asked if they prefer to receive emails or text messages. Whatever method they choose will be the main method of communication while they are in the parking garage as well.
REQ-33	5	Ad-hoc customers will be allowed to choose their parking spot through the real-time visual display of the parking garage.
REQ-34	3	Ad-hoc customers will be notified upon arrival if there are any available parking spots.
REQ-35	5	The elevator system will have two elevators- one will take customers up and one will take customers down.

REQ-36	3	Customers should receive all notifications about parking, registration/reservation confirmations, etc. within a minute.
REQ-37	2	The system will adjust to different parking garages with different layouts.
REQ-38	5	A simulation/virtual parking garage will need to be created in order to ensure that the system works effectively under real world conditions. The simulation will test that the customers can register for an account and login properly, and will also test that they can make reservations via the website. The simulation will also need to ensure that the features included to ensure safety and traffic control are working properly as well.
REQ-39	5	The system will charge higher rates when there is higher demand and fewer when lower demand.

4.3 On Screen Appearance Requirements

Identifier	**P.W.**	**Description**
REQ-36	3	Allow user to input their username and password to log in to their account
REQ-37	4	Consumers will be able to do a guest log in for non members by inputting an email and name
REQ-38	2	On account page, users will be able to click a button to go to reservation page
REQ-39	2	On account page, users will be able to click a button to go to account management page
REQ-40	2	On account page, users will be able to click a button to go to reservation cancellation page
REQ-41	3	On account page there will be a list of nearby activities and their locations
REQ-42	3	Spot management page will have map of garage to help users pick and navigate to their spot
REQ-43	2	Users will be able to input calendar date and time for reservations.
REQ-44	5	Users will be able to scroll through garage map pages.

4.4 Smart Pricing Requirements

Identifier	P.W.	Description
REQ-52	4	Support fixed-rate charging.
REQ-53	5	Support dynamic rate charging.
REQ-54	3	Extract hourly information from Seattle dataset.
REQ-55	3	Extract weekly information from Seattle dataset.
REQ-56	3	Extract monthly information from Seattle dataset.
REQ-57	3	Extract yearly information from Seattle dataset.
REQ-58	4	Discard useless information from Seattle dataset.
REQ-59	5	Get the average prices for different locations.
REQ-60	5	Interpolate the hourly information with LS algorithm.
REQ-61	5	Use demand-price functions to adjust the demands according to different prices.
REQ-62	5	Decrease the peak-hour demands by increasing the prices.
REQ-63	5	Support the manager overwrite the suggested prices.
REQ-64	5	Decrease the price of the daytime except the rush hours to increase revenue.
REQ-65	5	Customers can get an estimated prices before parking.
REQ-66	5	Repeat the result of some papers to verify the smart pricing system.
REQ-67	4	Large Data Volume Processing
REQ-68	4	Billing to the garage system
REQ-69	4	Dynamic Rate shown to user via web page

5 Functional Requirements Specification

5.1 Stakeholders

This system is primarily intended for implementation in current garages to help garage owners increase efficiency and profits. This system will also be beneficial towards people who can utilize or maintain the system. Below is a list of possible stakeholders in the system.

- Parking Garage Owners
- Business Enterprises

- Customers: ad-hoc and walk-ins

5.2 Actors and Goals

Actors	**Goals**	**Use Cases**
Parking Interface	To display for customers on the website to allow them to reserve spots in the garage ahead of time	UC-2,UC-3,UC-4,UC-6
Parking interface	To display for ad-hoc customers to choose vacant spots to park in once they enter the garage	UC-2,UC-4,UC-6, UC-16
License-Plate Reader	To read the license plates of vehicles and passes it to the system	UC-9,UC-10
System	Update customer information and payment data	UC-2,UC-11,UC-14, UC-15
System	Update parking information in the garage	UC-2,UC-15,UC-16
Customer	To register in the system	UC-1
Customer	To review the available parking lots	UC-2
Customer	To enter the garage	UC-9
Customer	To leave the garage on foot	UC-8
Customer	To leave the garage by car	UC-10,UC-14
Customer	To make reservations on the website.	UC-1,UC-2,UC-3
Customer	To cancel his reservation	UC-7
Customer	To do Ad-hoc parking while choosing their spot	UC-2, UC-4
Customer	To update or modify account information	UC-11
Customer To receive notifications	To park in the garage UC-15	UC-5 Customer
Garage Owner	To choose a pricing strategy	UC-13
Customer	To pay bills for the parking service	UC-14
Garage Owner	To analyze statistics of the usage of the garage	UC-1,UC-3,UC-4, UC-13

Garage Owner	To keep track of parking spots in garage	UC-2,UC-15
Garage Owner	To manually input information in case of system failure	UC-13
Sensors	To monitor traffic on each floor	UC-5,UC-6
Sensors	To check to make sure customers have parked in the correct spot	UC-5,UC-6
Elevators	To bring customers to their designated floors	UC-5,UC-9
Elevators	To bring customers to the ground floor once they are leaving the garage	UC-9,UC-10
Security System	To ensure only customers and the people they are traveling with can enter or exit the garage	UC-8, UC-9,UC-10
Security System	To send customers an alphanumeric code they will need to enter the garage along with periodic updates about time elapsed	UC-10,UC-15

5.3 Use Cases

5.3.1 Casual Description

- UC-1: Register- to register an account on the website
- UC-2: View Spots - to view available and unavailable spots in a selected time window
- UC-3: Reservation - to save a spot reservation online ahead of time
- UC-4: Ad-Hoc - to select a spot at the garage with no reservation
- UC-5: Park - to park a driver's car
- UC-6: Switch spots - to change a reserved or selected spot before parking
- UC-7: Cancel reservation - to cancel a reservation before parking
- UC-8: Unlock door - to unlock and gain access to the pedestrian door to exit or enter the garage on foot
- UC-9: Enter- for a driver to enter the garage in their car

- UC-10: Exit - for a driver to exit the garage in their car
- UC-11: Update information - to change account information on the website
- UC-12: Set pricing - to override smart pricing options or set smart pricing options
- UC-13: Manual Input - to input information directly in case of system failure
- UC-14: Payment - to pay bill at the end of parking
- UC-15: Track Time- to track how much time has elapsed since parking
- UC-16: No Availability- to notify ad-hoc customers in case there is no availability in the garage

5.3.2 Use Case Diagram

Customer:

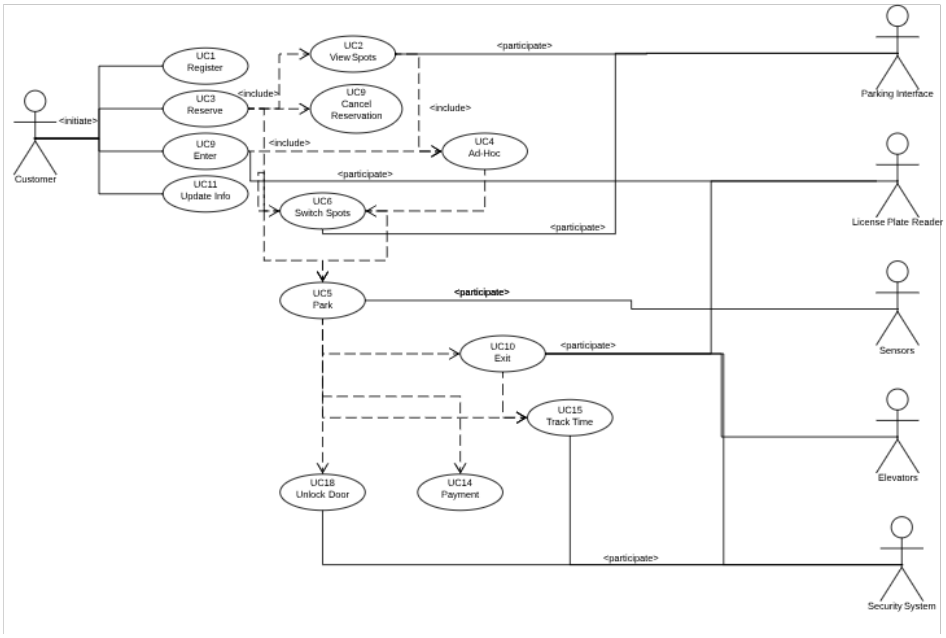


Figure 1: Customer use case.

The following diagram shows how the previously stated use cases interact with the customer and the actors. UC1,UC3, UC9 and UC11 are all first tier use cases as they interact directly with the customer. Since customers will be allowed to view all the available spots, UC2 (view spots) is directly linked to the parking interface.Through reservations, the customer will also be allowed to switch spots. Additionally, UC9 will be linked to license plate reader as upon entrance, the licence plate reader takes a photo of the license plate.

Once the customer has entered the garage, the customers will park in their designated spot, which will be monitored by the sensors, and will leave the garage. Entering and exiting the garage after the customer first parks will be monitored by the security system, as the in order to be allowed to exit and then reenter the garage, the customers will have to input the alphanumeric code they had received into an interface. Customers will also be required to pay for their spot once they have parked and the system keeps track of how much time has elapsed. This ensures that customers are not going over their allocated parking time.

Owner:

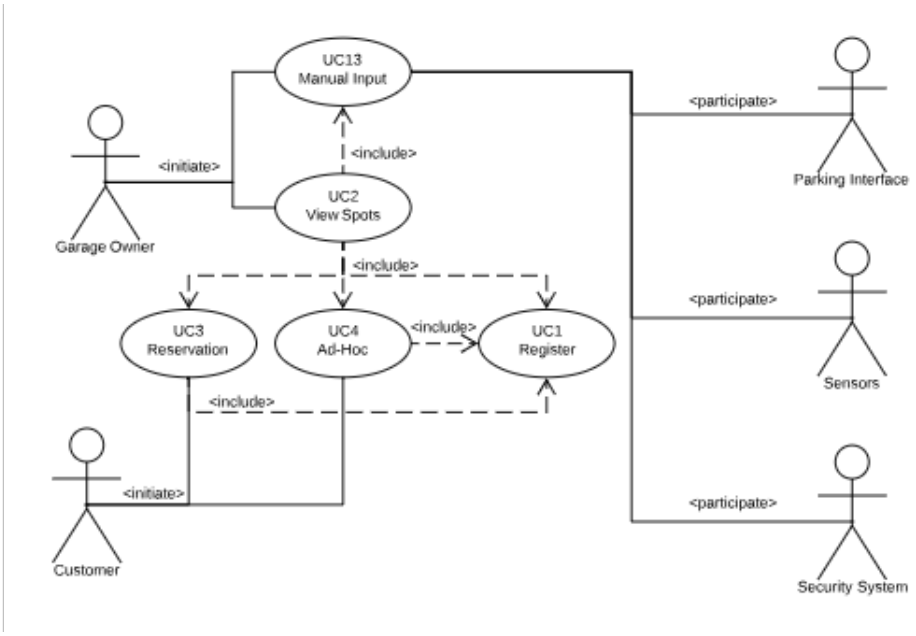


Figure 2: Owner use case.

The following use case diagram show the previously stated use cases interact with the owner of the garage along with the participating actors. In cases of system failure, the garage owner will have to manually update the system to show all occupied spots and the current prices, which is why it directly connects with UC13 (manual input) and UC2(view spots). For this reason, UC13 directly interacts with the parking interface actor, sensors, and security system. The view spots use case also connects to the reservation, ad-hoc, and register use cases which directly interacts with the customer.

5.3.3 Traceability Matrix

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16
REQ-1	3									X							
REQ-2	3			X	X					X				X			
REQ-3	5		X	X	X									X			X
REQ-4	5		X														X
REQ-5	4	X										X					
REQ-6	4	X		X								X				X	
REQ-7	5		X		X												
REQ-8	5												X		X		
REQ-9	4					X											
REQ-10	4					X											
REQ-11	3					X										X	
REQ-12	3					X										X	
REQ-13	4													X	X		
REQ-14	2												X				
REQ-15	4			X													
REQ-16	4								X	X	X						
REQ-17	3					X	X										
REQ-18	5			X													
REQ-19	2			X													
REQ-20	2							X									
REQ-21	4												X		X		
REQ-22	3							X									
MAX PW		3	4	4	3	5	2	2	1	4	5	3	4	2	5	4	2
TOTAL PW		20	15	23	25	17	3	5	10	10	4	15	20	10	23	10	10

Figure 3: Traceability matrix.

The following traceability matrix was derived from the use cases and the functional requirements. The priority weights for each requirement and use case was alloted based on the complexity of the requirement/use case and how long it will take to implement it. The priority weights were given on a scale of 1 to 5.

5.3.4 Fully Dressed Description & Sequence Diagram

Use Case UC-1: Register
Related Requirements: REQ-05, REQ-18, REQ-24, REQ-25, REQ-27, REQ-32, REQ-38
Initiating Actor: Customer
Actor's Goal: To register an account on the website
Participating Actors: Customer, System
Preconditions: The system will request the necessary information from the customer on the registration page.
Postconditions: The customer's account will be stored within the database which will be backed up regularly
Flow of Events for Main Success Scenario: $-> 1$. Customer accesses the website and chooses to "Register" an account < -2. The system returns a page that requests the necessary fields $->3$. The customer enters information into the required data fields < -4. The system checks the username and email against the database. a. If the username or email is not unique, return to step 3 to alter login information. b. If the username and email are both unique, the system takes the information and enters it into the database. the system notifies the user for successful registration.

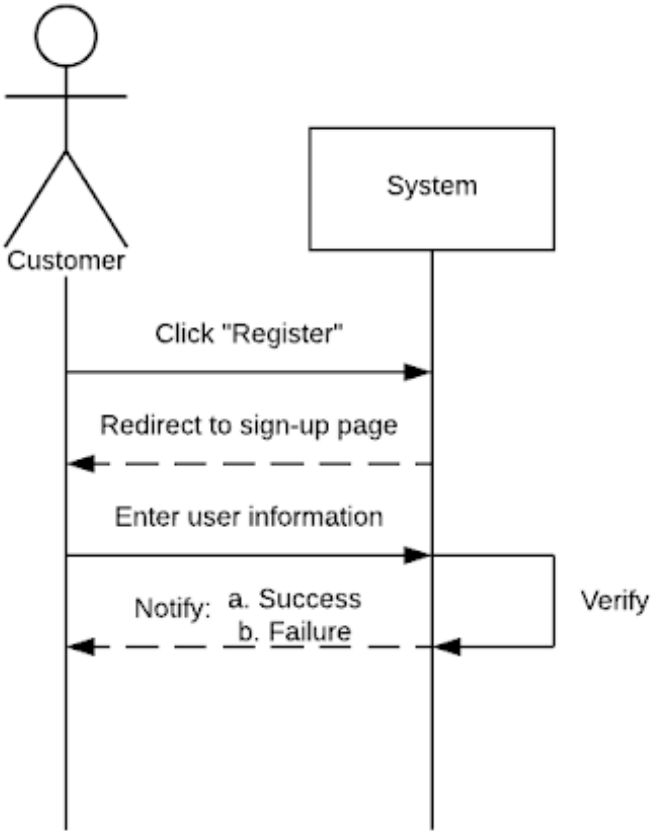


Figure 4: Register sequence.

Use Case UC-3: Reservation
Related Requirements: REQ-03, REQ-04, REQ-06, REQ-13, REQ-15, REQ-18, REQ-19, REQ-30, REQ-36, REQ-38, REQ-40, REQ-43
Initiating Actor: Customer
Actor's Goal: To reserve parking spots at the garage in advance
Participating Actors: Customer, System
Preconditions: The spots that are being reserved must be reserved at least a day in advance.
Postconditions: The customer's reservation will be stored in the database.
Flow of Events for Main Success Scenario: $->$1. Customer logs into his account and clicks "Reserve a Spot". a. As a result of a failed login, the customer repeats step 1. b. If the login is successful, continue to step 2. $<-$2. The system returns a page that requests the time block for the customer's reservation $->$3. The customer enters information into the required data fields (date, time block) $<-$4. The system checks the database of reserved spots during that time frame a. If there are no available reservation spots during the specified time frame, the system will inform the customer that there are no available spots to be reserved. (no reservation is made: flow ends here) b. If there are available spots during the desired time frame, the customer will be presented with a UI of the available spots that they can select from. (continue to step 5) $->$5. The customer selects a parking spot to reserve $<-$6. The system enters the reservation into the reservation database. A message is displayed to show that the reservation went through.

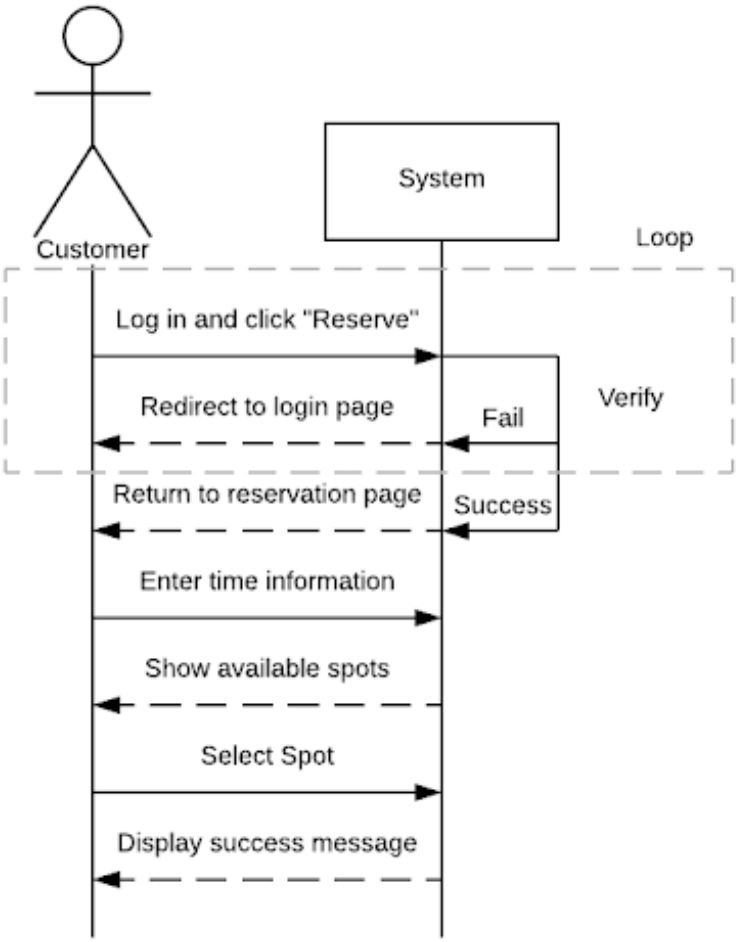


Figure 5: Reserve sequence.

Use Case UC-4: Ad-hoc
Related Requirements: REQ-03, REQ-04, REQ-07, REQ-09, REQ-24, REQ-33, REQ-34
Initiating Actor: Customer
Actor's Goal: To obtain a parking spot at the garage without any reservation
Participating Actors: Customer, System
Preconditions: N/A
Postconditions: The customer will be able to park in the garage
Flow of Events for Main Success Scenario: $->$1. The customer arrives at the parking garage and chooses the "Ad-hoc" option of parking. $<-$2. The system displays a real-time interface of the current parking situation of the garage (vacant and occupied spots) a. If there are no vacant spots, the customer has to leave b. If there are vacant spots, the customer can continue to step 3. $->$3. The customer can select a vacant spot to park in. $<-$4. The system requests some basic information from the customer (name, license plate, contact information, and credit card information) $->$5. The customer enters all the necessary information. $<-$6. The system checks the validity of the information. a. If the information does not fit the requirements (specifically credit card number), loop back to step 6. b. If the information fits the requirements, the system enters the customer's information into the database. The customer's spot now shows up as "occupied" on the real-time interface, and the customer can proceed with the parking process.

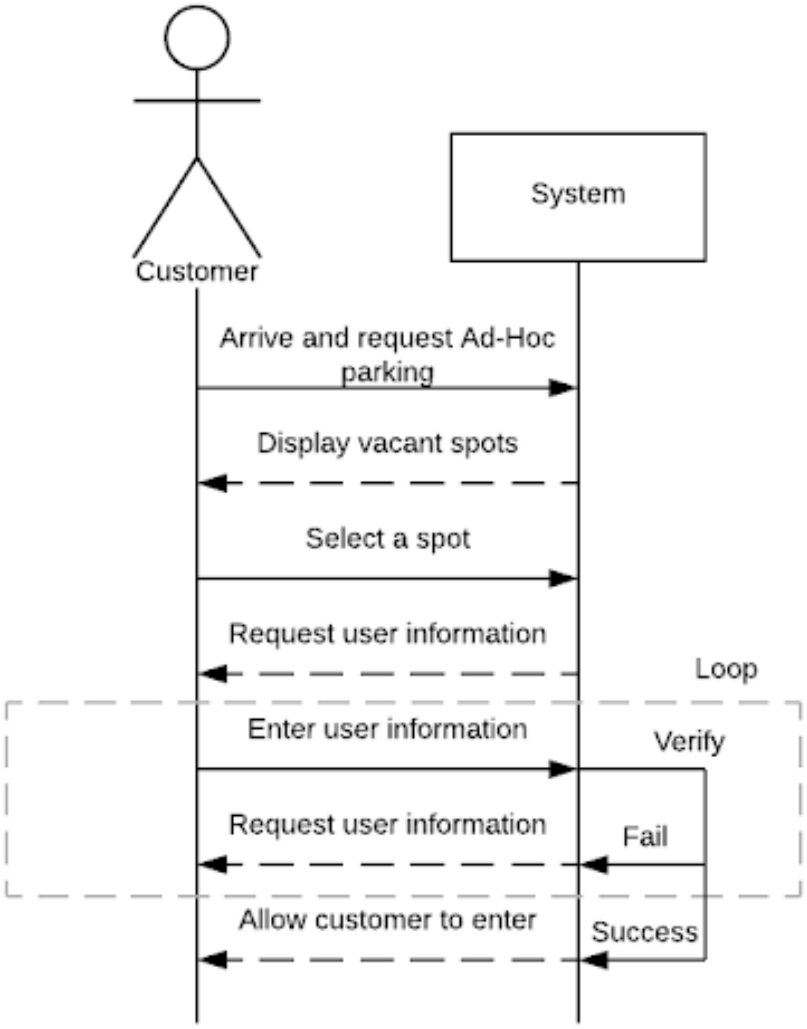


Figure 6: Ad-hoc parking sequence.

Use Case UC-14: Payment
Related Requirements: REQ-03, REQ-04, REQ-06, REQ-13, REQ-15, REQ-18, REQ-19, REQ-30, REQ-36, REQ-38, REQ-40, REQ-43
Initiating Actor: Customer
Actor's Goal: To pay the bill based on the system information.
Participating Actors: The security system
Preconditions: The customers registered and entered to park the car.
Postconditions: The security system will let the customers out.
Flow of Events for Main Success Scenario: $->$1. Customer logs into his account and clicks "Leave and Pay". a. As a result of a failed login, the customer repeats step 1. b. If the login is successful, continue to step 2. $<-$2. The system returns a page that shows the total money to be paid. $->$3. The customer enters information into the required data fields (date, time block) $<-$4. The system check the payment information and deduct it from the bank. a. If failure, return to 2 b. If success, turn to 5. $->$5. The customer receives the receipt. $<-$6. The secure system will let the customer out.

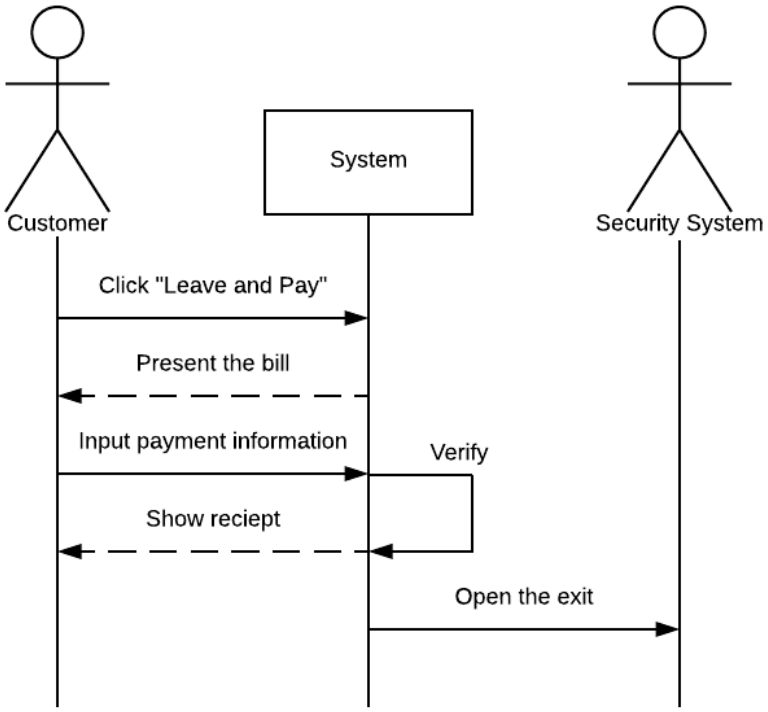


Figure 7: Leave and payment sequence.

6 Effort Estimation

6.1 Actor Classification

UAW: 3(3) + 3(2) + 3(1) = 18. The complexity of the unadjusted actor weights is determined by user interface design, and the weights are based off the complexity.

Actor Name	Description of relevant characteristics	Complexity	Weight
Parking Interface	Customers interact with the parking interface through a graphical display on the website or the on-site monitor	Complex	3

License Plate Reader	The scanner interacts through a camera device and software API that allows license plates to be read	Simple	1
System	Updates customer information and parking information in their respective databases via manual input by the user or automatic input from other modules	Average	2
Customer	Customer can interact with the system through a graphical interface to view spots and change options	Complex	3
Garage Owner	Garage owner can also interact with the system via the graphical interface, but most interactions are simple text-based inputs	Average	2
Sensors	Returns a value to determine whether or not a car is parked in a respective spot	Simple	1
Elevators (Traffic Control System)	Allows customers to reach the desired floor in their cars, while also receiving data from other modules to limit traffic within the garage	Simple	1
Security System	Creates and manages an authentication code for customers and requires text-based input from customers to use the doors	Average	2
Smart Pricing System	Charge customers according to current demand and learn to self-adjust future price	Complex	3

Table 6: Actor Classification

6.2 Use Case Classification

UUCP: 3(15) + 7(10) + 7(5) = 150.

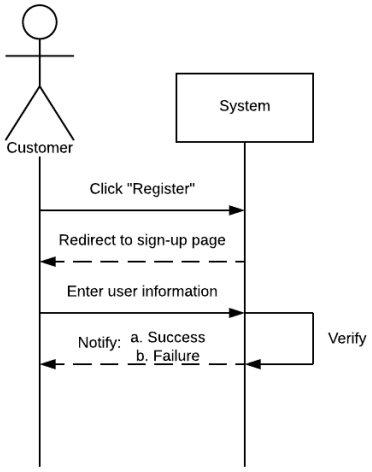
The complexity of the use cases is determined by the number of participating actors and the number of steps it takes to get to the success scenario. The higher the number of participating actors and number of steps, the higher the complexity. The weights are based off the complexity.

Use Case	**Description of relevant characteristics**	**Complexity**	**Weight**

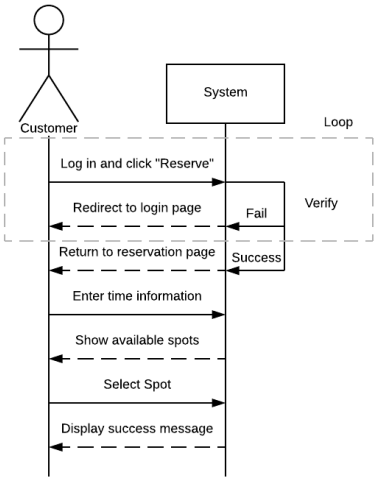
UC-1: Register	Simple user interface. 2 Participating actors. 4 step success scenario	Average	10
UC-2: View Spots	Graphical interface. 2 Participating actors. 2 step success scenario	Average	10
UC-3: Reservation	Graphical Interface. 3 Participating actors. 6 Step success scenario	Complex	15
UC-4: Ad-Hoc	Graphical Interface. 3 Participating actors. 6 Step success scenario	Complex	15
UC-5: Park	No interface. 2 Participating actors. 2 Step success scenario	Simple	5
UC-6: Switch spots	Graphical interface. 2 Participating actors. 3 Step success scenario	Simple	5
UC-7: Cancel Reservation	Graphical interface. 2 Participating actors. 2 Step success scenario	Simple	5
UC-8: Unlock Door	Text-based interface. 3 Participating actors. 6 Step success scenario	Average	10
UC-9: Enter	Simple interface. 2 Participating actors. 3 Step success scenario	Simple	5
UC-10: Exit	Simple interface. 2 Participating actors. 2 Step success scenario	Simple	5
UC-11: Update Info	Text-based interface. 3 Participating actors. 6 Step success scenario	Average	10
UC-12: Set pricing	Text-based interface. 3 Participating actors. 8 Step success scenario	Average	10
UC-13: Manual Input	Text-based interface. 2 Participating actors. 6 Step success scenario	Simple	5
UC-14: Payment	Simple interface. 2 Participating actors. 6 Step success scenario	Average	10
UC-15: Track Time	No interface. 2 Participating actors. 4 Step success scenario	Simple	5
UC-16: No Availability	Simple interface. 4 Participating Actors. 6 Step success scenario	Average	10
UC-17: Smart Pricing	No interface. 2 Participating actors. 4 Step success scenario	Complex	15

6.3 Sequence Diagrams for UCs

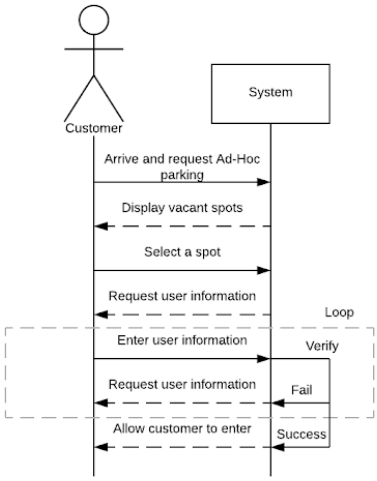
6.3.1 UC-1: Register



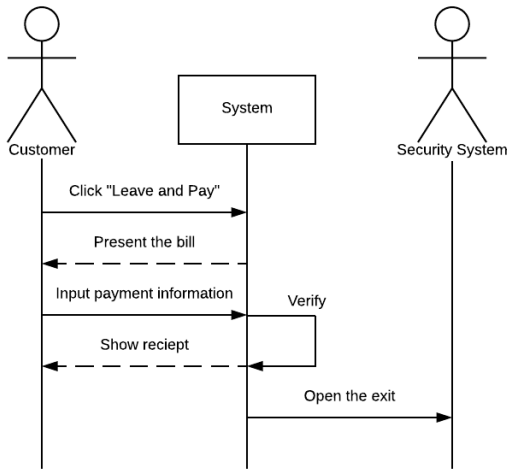
6.3.2 UC-3: Reservation



6.3.3 UC-4: Adhoc



6.3.4 UC-14:Payment



6.4 Technical Complexity Factors (TCFs)

Technical Factor Total: 44.

Technical Factor	Description	Weight	Perceived Complexity	Factor (W*PC)
T1	Distributed system: website and database are separate entities that must communicate with each other as well as with garage modules	2	3	6
T2	Users expect good performance, slow performance could result in increased traffic/wait time and incorrect billing for customers	1	4	4
T3	End-user expects efficiency but nothing exceptional	1	3	3
T4	Calculation of dynamic pricing model is very complex and requires heavy processing	1	4	4
T5	Design is reusable and can be implemented in any garage with the appropriate hardware modules	1	3	3
T6	Installation requires various prerequisites and is not automatic; also requires hardware installation in garage but overall should not be too difficult	0.5	3	1.5
T7	Ease of use is very important; main goal is to be easy for all customers to use	0.5	5	2.5
T8	Website and database systems can be easily moved to new machines, but portability of garage hardware is not important	2	0	0
T9	Easy to change and add modules to due to modular design of components, minimally required	1	1	1

T10	Should concurrently support many users (both in the garage and accessing the website)	1	4	4
T11	Special security features have been implemented and are vital to the operation of the garage	1	3	3
T12	No direct 3rd party access	1	0	0
T13	Virtually no training required; system is almost fully automated	1	0	0
T14	Learning parking demand from history statistics is very difficult	2	3	6
T15	Proposing a dynamic pricing function based on parking demand is complicated	2	3	6

7 Domain Analysis

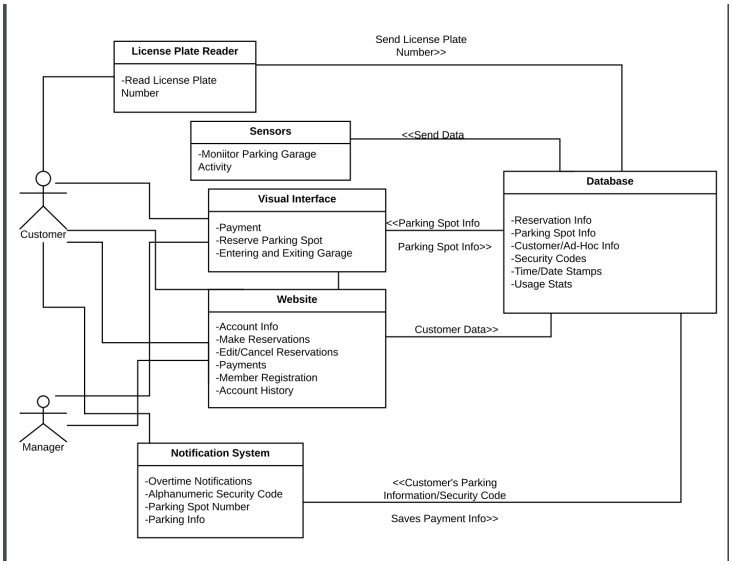


Figure 8: The Whole Domain model.

7.1 Domain Model Derivation

The following domain model was derived from the use cases and requirements with the highest priorities. These use cases include allowing customers to login and/or register for an account (UC-1) and make reservations (UC-3), along with allowing both registered members and ad-hoc customers to view spots, park their car, keep track of their parking duration, and pay for their bill at the end (UC-2,UC-4,UC-5, UC-14, and UC-15).

The key boundary concepts of this system are the license plate reader, the visual interface that the customers will interact with upon entering the garage, the website, and the notification system. The key internal concepts of this system are the databased and sensors.

The license plate reader is a camera system that takes a photo of the license plate when the customer first enters the garage, and compares it to the database entries to determine if the customer has an existing reservation. Based on that, the visual interface, which retrieves the data from the database, either allows the customer to select a parking spot in the garage or simply directs the customer to their reserved parking spot. The information

inputted from the visual interface is saved in the database. Customers also have direct access to the website, where they can create accounts to make reservations and track their account history. Customer data along with any upcoming reservations are saved in the database. Along with this, on the website, customers will be able to see a display of all the available parking spots at a given time via the visual interface and database. The visual interface for the current state of the parking lot is constantly maintained by sensors placed throughout the parking garage. The notification system periodically sends customers updates and notifications. Through the notification system, the customers can track the time that has elapsed and will also receive an alphanumeric security code that will be required for them to enter the garage. The notification system interacts with the database to retrieve customer and parking information. There are also sensors in place to monitor traffic activity in the garage and ensure that the the customers are parking in the correct spot.

7.2 Tracebility Matrix

The following traceability matrix was derived from the use cases and the concepts. For each use case, the different concepts that are involved to implement the use case are shown, and the priority weights are given based on how many concepts are involved in the implementation of the use case and the complexity of the use case itself.

PW	Use Cases	License Plate Reader	Visual Interface	Website	Database	Notification System	Sensors
8	UC1			X	X		
15	UC2		X		X		X
23	UC3			X	X	X	
13	UC4		X		X		
17	UC5	X	X			X	X
3	UC6		X		X		X
13	UC7			X	X		
7	UC8					X	X
10	UC9	X	X			X	X
4	UC10					X	X
12	UC11			X	X		
7	UC12			X	X		
14	UC13		X	X	X	X	
23	UC14			X	X		
4	UC15			X	X	X	X
10	UC16		X	X		X	

Figure 9: The tracebility matrix of domain model.

7.3 Concept Definitions

Responsibility Description	Type	Concept Name
To check if customer has a reservation	K	License Plate Reader
To collect information for ad-hoc customers	D	Visual Interface
To give real time display of all currently available parking spots	N	Visual Interface
To allow ad-hoc customers to pay for parking	D	Visual Interface
To give managers access to current situation in parking spot	N	Visual Interface
To collect user information and make reservations	N	Website
To give registered customers an alternate payment method	D	Website
To register new members	N	Website
To manage smart pricing system	D	Database
To hold customer/parking/license plate/reservation info	N	Database
To keep track of time elapsed	D	Database
To manage usage stats	D	Database
To extract reservation information for incoming customers	K	Database
To allow managers to edit pricing and access customer info	K	Database
To send customers periodic notifications regarding time elapsed	D	Notification System
To send customers parking confirmation and alphanumeric security code	D	Notification System
To manage traffic control	N	Sensors
To ensure that customers are parking in the correct spots	K	Sensors
To monitor occupancy of the garage	N	Sensors

Association definition:

Concept Pair	Association Description	Association Name
License Plate Reader ⟷ Database	License plate reader scans the plate of cars and sends the plate number to the database where it is stored to be accessed later.	Store license
Visual Interface ⟷ Database	Visual interface requests information on the status of parking spots from the database to be displayed to customers.	Get parking info
Website ⟷ Database	Website sends information about reservations, users, and payment to the database to be accessed by other systems.	Update user info
Notification System ⟷ Database	Notification system accesses database for customer information, such as phone number/email and parking spot number	Get notification info
Security System ⟷ Notification System	Security system signals notification system to send authentication codes to customer	Allow security code
Security System ⟷ Database	Security system requests security information stored in the database, such as a customer's unique security code	Get security code
Sensors ↔ Database	The sensors constantly monitor traffic control as well how occupied the garage is at any given time. Based on the occupancy and traffic, the database can set a price for certain spots.	Set Price

7.4 System Operation Contracts

UC-1: Register

- Preconditions:
  - Customer is not already a registered member
  - Customer will enter a valid username, email address, and driver's license number
  - Customer will create a password

- Postconditions:
  - Customer's account information is stored in the database

UC-3: Reservation

- Preconditions:
  - Customer is logged into their account on the website
  - Reservations must be made at least a day before requested reservation time
  - The selected spot is available for reservation at the requested time
- Postconditions:
  - Customer's reservation will be stored in the database

UC-4: Ad-Hoc

- Preconditions:
  - Customer does not have a reservation for the current time
- Postconditions:
  - Customer is able to park in the garage
  - Customer will be registered for an account if they did not previously have one

UC-14: Payment

- Preconditions:
  - Customer is done parking and has moved their car from their spot
  - Customer had either parked by reservation or ad-hoc
- Postconditions:
  - Customer will be charged for their stay based on agreed upon rates
  - Security system will allow customer to leave

8 Interaction Diagrams

8.1 Use Case 1: Register

We decided to assign the responsibility to register to the website (system), as the website is the main interface through which the customer can make parking reservations. The website has the responsibility of allowing customers to login and make reservations, which ensures that the website has focused specialty and does not have too many responsibilities assigned to it. Even though the customer interacts with the system to for logging in and making reservations, we assigned the database the responsibility to verify and store the data that is being received. In this way the database can easily access information about customer when it is needed for parking. We use the Publisher-Subscriber design pattern to improve this use case's design. In this case, the customer is the subscriber while the garage itself is the publisher. Once the subscribers input valid information, the publisher releases information of interest to the subscriber(that their account has been created). On the other hand, if the subscriber inputs invalid information, the publisher shows an error message showing no registered account.

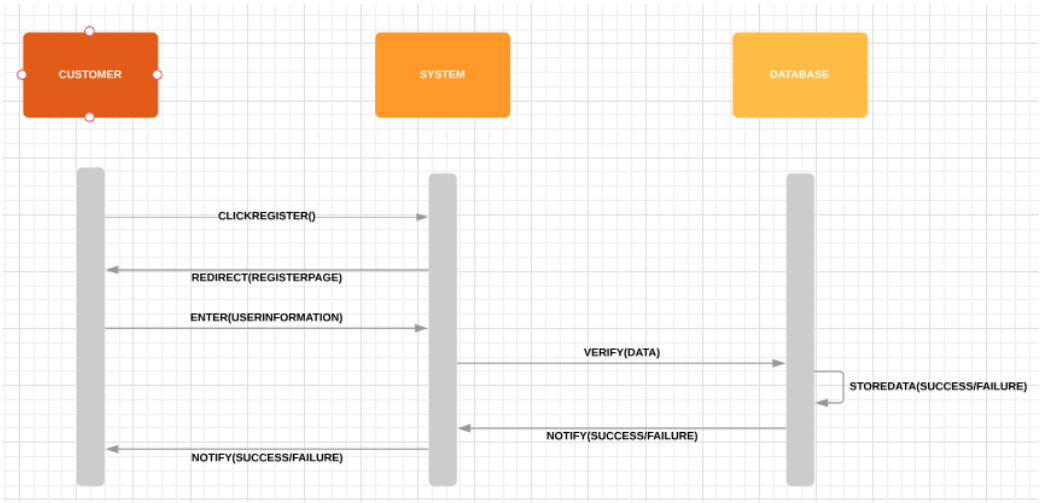


Figure 10: Sequence Diagrams for Use Case 1.

8.2 Use Case 3: Reservation

When a customer attempts to place a reservation, the system attempts to verify their account by querying the database for the correct account info. In this use case, the database's main responsibilities are keeping track of and reporting account information and reservation/spot information. This data is then given back to the system, which is responsible for passing information from the customer to the database and displaying relevant information from the database to the customer in an easy to understand format. The system is also responsible for processing the customer's action on the website, such as redirecting the customer to different pages on the website based on their requests. As stated above, we use the Publisher-Subscriber design pattern to improve this use case's design as well. Once the customer inputs a desired time frame during which they want to reserve a parking spot, they receive the status of the parking spots in the garage that they are able to reserve a spot. In this test case, the publisher gives the subscriber the pertinent information that the subscriber needs to reserve a free parking spot or to notify the subscriber of no valid reservation spots .

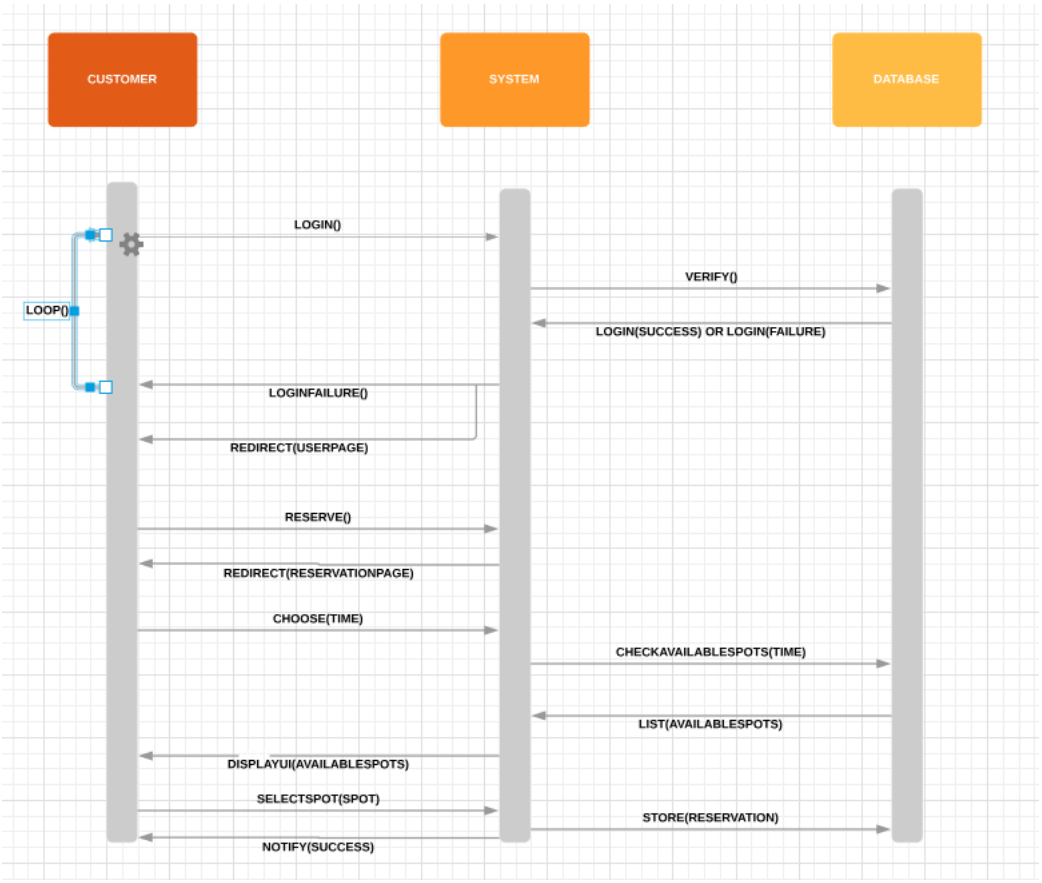


Figure 11: Sequence Diagrams for Use Case 3.

8.3 Use Case 4: Ad-hoc

When an Ad-Hoc user enters the garage, they will interact with a user interface that will be placed at the front of the parking garage. The user interface(denoted as system in the interaction diagram) will be responsible for showing the ad hoc customers all the available parking spots and for requesting and verifying user information so that the users can enter the garage. Since the user interface is the first system that ad-hoc users interact with upon entering the garage, we decided to assign the previously mentioned responsibilities to the the system as both responsibilities should be completed before the user enters the parking garage. Similarly to Use Cases 1 and 3, the subscriber is the customer who selects a parking spot to park

in at the entrance gate terminal. The publisher(garage control) will send the subscriber a list of possible ad-hoc parking spots that the subscriber can then choose from.

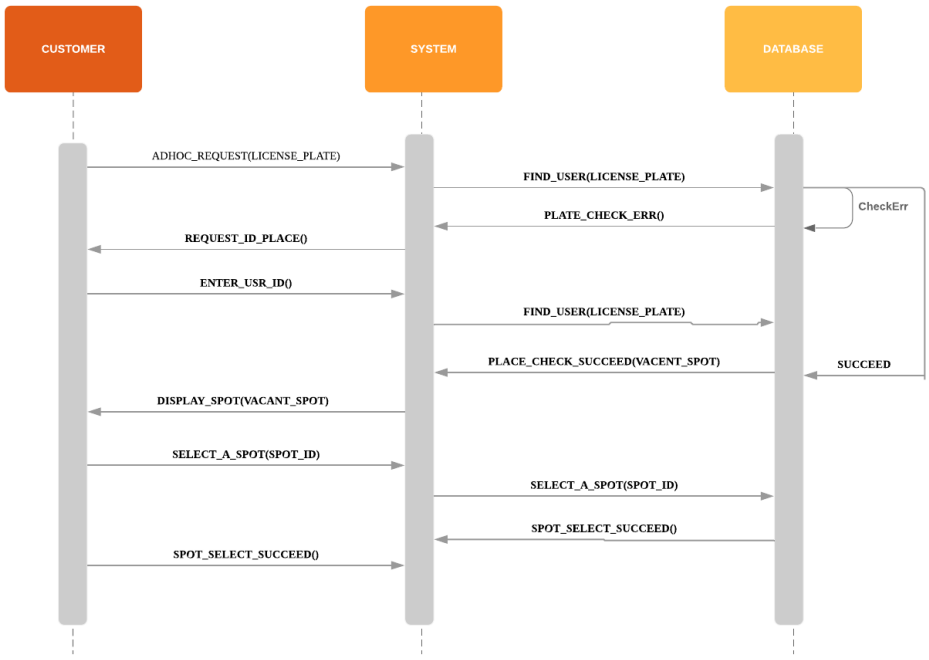


Figure 12: Sequence Diagrams for Use Case 4.

8.4 Use Case 14: Payment

After a user has completed their parking (reservation or ad-hoc) they will move to the exit. They will interact with the system and request to pay and leave. The system will read the license plate and request parking information from the database. Total cost will be computed and then the system will check the database to see if this is a registered user or not. If the user is a registered user then the credit card associated with the account will be charged and the person will be allowed to leave. If the account is not registered then payment information and billing information will be requested. Upon verification of payment method, the user will be allowed to leave. This

Use Case uses the Publisher-Subscriber design pattern as well. When the user reaches the exit gate, in order to leave the garage, the publisher will present the subscriber with the total parking cost. The customer will pay the noted price and will be allowed to leave the garage premises.

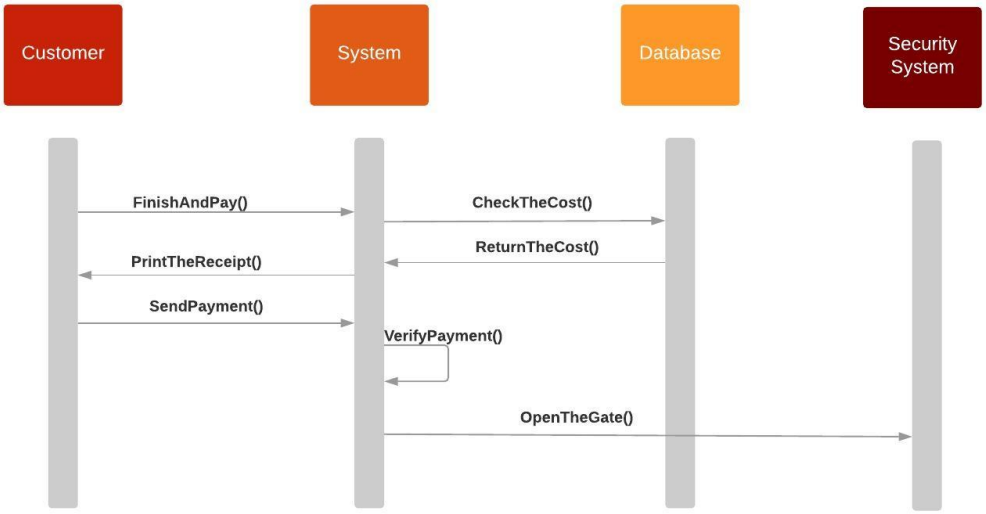


Figure 13: Sequence Diagrams for Use Case 14.

9 Class Diagram and Interface Specification

9.1 Class Diagram

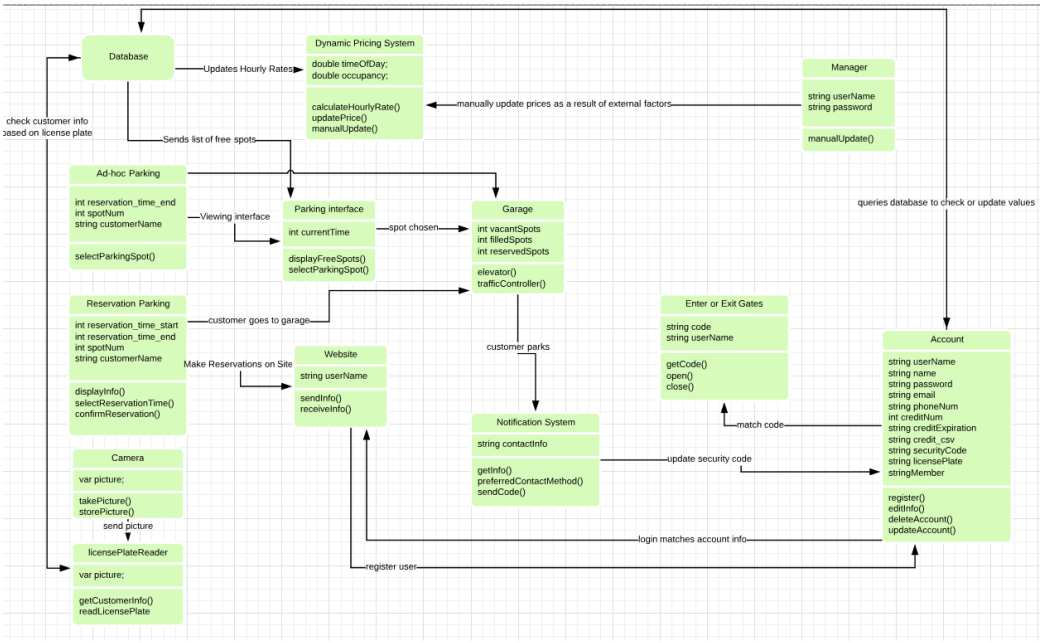


Figure 14: Class Diagram.

9.2 Design Patterns

For the majority of our project, we use the Publisher-Subscriber Design Pattern. As stated in Section 8 (Interaction Diagrams), when the subscriber executes certain actions such as selects a time slot for reservation or tries to leave the garage, the publisher will send the subscriber the pertinent information the they need to complete their actions. The publisher will send the subscriber a GUI with free spots in the garage or a notify the subscriber of the price they need to pay for their parking spot. This design pattern improves our implementation of the project by making most of the transactions based on request vs. event- based communication. This forms a line of direct communication between the publisher and subscriber that streamlines most of the transactions between the two as well as the interactions with the database.

9.3 Object Constraint Language

For our classes, we have some simple preconditions for class and their operations. Ad-hoc Parking and reserved parking have the preconditions of signing into the garage, the license plate of the driver, and the contact method. These are then relayed to the database. The camera class has no preconditions besides needing a vehicle to be in front of it to be able to sense the license plate and send the data to the entrance gate. The notification system needs the customer's contact information so the security code can be sent to the customer. For the account class, all the fields in the registration page must be filled out in a valid format in order to be stored in the database. The manager class needs the special login information to be able to manually update the dynamic pricing system or values in the database.

9.4 Data Types and Operation Signatures

1. Manager
   (a) Attributes
      - int customer_id: identification number of the customer
      - string first_name: customer's first name
      - string last_name: customer's last name
      - string email: customer's email address
      - string phone_number: customer's phone number
      - string password: customer's password
      - date dob: customer's date of birth
      - string creditcardno: customer's credit care number
      - string csv: credit card's csv code
      - string securitycode: security code sent to customer's phone
      - enum Type('Membership', 'Ad-hoc')

   (b) Operation: (all operations attach customer to any created objects)
      - Register(): Creates a new user and inserts the user info into database.
      - addCar(): creates a car and inserts into database
      - addCreditCard(): creates a credit card and inserts into database
      - newReservation(): creates a reservation and inserts into database

2. Camera

   (a) Attributes

      - var picture : picture of the customer's vehicle (to get the license plate)

   (b) Operation: (all operations attach customer to any created objects)

      - getCustomerInfo(): check the customer information from the database
      - readLicensePlate(): Read the license plate with the picture taken from the camera

3. PriceSystem

   (a) Attributes

      - int currentTime: the current time in the current time zone

   (b) Operation: (all operations attach customer to any created objects)

      - calculateHourlyRate(): calculate the hourly based price
      - updatePrice(): update the price to the database
      - manualUpdate(): manually update the price to database

4. Ad-hoc Parking

   (a) Attributes

      - int reservation_time_end: time that the customer leaves the garage with their vehicle
      - int spotNum: parking spot the customer parked in
      - string customerName

   (b) Operation: (all operations attach customer to any created objects)

      - selectParkingSpot(): select the Parking Spot with spot id spotNum

5. Reservation Parking

   (a) Attributes

      - int reservation_time_start: start of reservation time
      - int reservation_time_end: end of reservation time

- int spotNum: spot that is reserved by the customer
- string customerName: name of customer

(b) Operation: (all operations attach customer to any created objects)

- displayInfo(): display the vacant parking spots to customers
- selectReservationTime(): let customers to select the time spots
- confirmReservation(): confirm the customers' revervation

6. Parking Interface

(a) Attributes

- int currentTime: current time in current time zone

(b) Operation: (all operations attach customer to any created objects)

- displayFreeSpots(): display the vacant parking spots to customers
- selectParkingSpot(): let the customers select the parking spot

7. Garage

(a) Attributes

- int vacantSpots: number of vacant spots in the garage
- int filledSpots: number of filled spots in the garage
- int reservedSpots: number of reserved spots in the garage

(b) Operation: (all operations attach customer to any created objects)

- elevator(): go to the floors where the selected parking spot on
- trafficController(): allow only one car entering one floor

8. Website

(a) Attributes

- string userName: customer's username used to log in online or at the terminal

(b) Operation: (all operations attach customer to any created objects)

- sendInfo(): send registering user information to customer module

- receiveInfo(): receive registered user information from customer module

9. NotificationSystem
   (a) Attributes
      - string contactinfo: email or phone number that customer can be contacted at
   (b) Operation: (all operations attach customer to any created objects)
      - getInfo(): get user information from the dataset
      - preferredContactMethod(): check the emailphone information
      - sendCode(): generate the info and send it to user
10. Elevator
   (a) Attributes
      - string code: security code that customer receives once they park
      - string username: username of customer
   (b) Operation: (all operations attach customer to any created objects)
      - getCode():get the verify code from user and check the code
      - open(): Open the gate
      - close(): close the gate

9.5 Traceability Matrix

1. Database
   (a) Responsibilities
      - Store customer and reservation information
      - Check to see if incoming customer has a reservation
      - Update hourly rates for dynamic pricing system
2. Website
   (a) Responsibilities

Classes	Domain Concepts								
	Database	Website	Dynamic Pricing System	License Plate Reader	Security System	Traffic Control	Parking Interface	Manager	Garage Controller
Manager Options	X	X	X					X	
Member Registration	X	X					X		
Member Login	X	X					X		
Entrance Gate						X	X		X
Exit Gate					X	X			X
Camera				X	X	X			
Notification System					X	X			
Parking Spot Sensor	X			X	X	X			X
Elevator				X	X	X			X
Statistics		X	X					X	
Reservation Parking	X	X	X	X	X	X	X	X	X
Ad-hoc Parking	X		X	X	X	X	X	X	X
Parking Spot Analysis			X			X	X	X	
Database Connection	X	X	X	X	X	X	X	X	X

Figure 15: Traceability Matrix

- Collect customer information
- Allows customers to make/update reservations

(b) Classes

- sendInfo(): collects customer info from website and sends it to database
- receiveInfo(): receives responses from the system and possibly display it

3. Dynamic Pricing System

(a) Responsibilities

- Set prices for customers based on various conditions

(b) Classes

- calculateHourlyRate(): calculate rate per hour for garage parking
- updatePrice(): update the current price
- manualUpdate(): in case of system failure, manually update prices

4. License Plate Reader
   (a) Responsibilities
   - Scan license plate to gather customer information on incoming customer

   (b) Classes
   - readLicensePlate(): query database with license plate to check for customer information
   - getCustomerInfo(): gather info about customer from license plate

5. Security System
   (a) Responsibilities
   - Notify the customer when they have successfully parked
   - Send alphanumeric security code to customer which they will use to enter and exit the garage

   (b) Classes
   - getInfo(): receive customer information
   - preferredContactMethod(): store customer's preferred contact method (ie. phone or email)
   - sendCode(): send alphanumeric/confirmation code to customer

6. Traffic Control
   (a) Responsibilities
   - Control the amount of traffic congestion in the garage
   - Uses sensors in the garage and cameras on each floor to gauge vehicular movement on each floor

(b) Classes

- displayFreeSpots(): show all available parking spots on visual interface
- selectParkingSpots(): select parking spot

7. Visual Interface (Parking Interface)

(a) Responsibilities

- Allow walk-in customers to choose parking spot
- Collect walk-in customers' information
- Allow for manual input in case of system failure

(b) Classes

- displayFreeSpots(): show all available parking spots on visual interface
- selectParkingSpots(): select parking spot

8. Manager

(a) Responsibilities

- Manually update values in the database
- Manually update price values for the dynamic pricing algorithm
- View statistics and reservations of the garage

(b) Classes

- displayFreeSpots(): show all available parking spots on visual interface
- selectParkingSpots(): select parking spot

10 System Architecture and System Design

10.1 Architectural Style

In terms of structure of the garage logic modules, the Auto Park system follows a component-based design style. Each task that must be completed within the system is performed by a module dedicated to that task (the notification system handles the sending of messages to customers, the traffic

control system handles the amount of cars driving on each floor, etc). Since there are many unique functions which must operate in parallel with one another and process the same information, each component of the system must be able to communicate with the relevant other subsystems. Every component is responsible for independently executing its primary functions while also coordinating with the other components to ensure smooth operation.

As for memory and data sharing, the system uses a database-centric architecture. All information relating to user accounts, parking space status, reservations, and payments are saved in databases which are accessible by the other modules in the system. This database design allows the various components of Auto Park to access, view, and edit the same data, allowing for easier communication between components in many cases.

The direct communication between components in the garage is also heavily based around an event-driven architecture. Each primary function in the system must happen at a specific step during a normal use case. For example, the security system waits for a customer to park their car correctly before generating a code, and the notification system waits for this code to be generated before sending it to the customer. Each step in the system's process is triggered by the completion or progress of a previous step, which is the main philosophy behind an event-driven messaging style.

10.2 Identifying Subsystems

Figure 16 shows all subsystems designed in our system.

10.3 Mapping Subsystems to Hardware

Our website runs using typical server client technology. The server handles all request sent out from clients, which could be cellphone, laptop or anything that can run a browser. Apart from this, our data processing runs on the spark platform. It's configurable for users to run over single machine, several machines and even hundreds of machines if necessary. The main structure of spark is shown in Figure 17.

We launch the data processing job through the driver program, which creates the sparkcontext according to the configurations. It will convert the job into multiple tasks and schedule them over different executors through cluster manager. Executors starts together when sparkcontext is created. When finishing the task, executors will send results back to cluster manager.

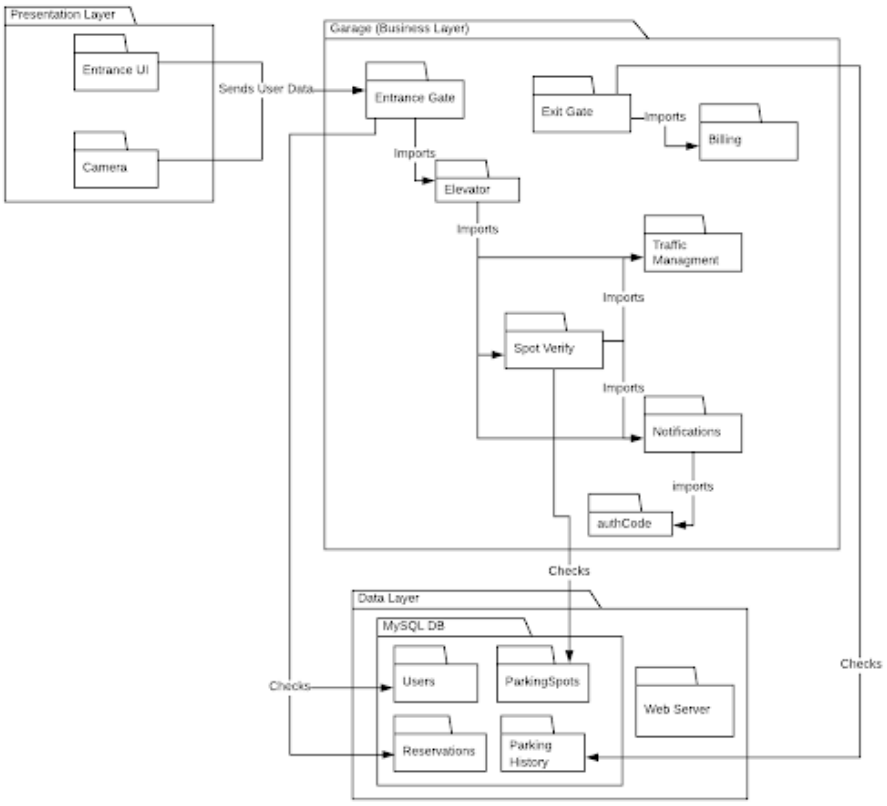


Figure 16: Subsystems in our System.

In our project, we run the data processing using 32 executors with driver memory 20GB and executor memory 80GB.

10.4 Persistent Data Storage

A MySQL database was used to store the data acquired by the system. The database stores information about the customer's contact/payment information as well as information (ie. date, time, reserved parking spot) about any reservations they may have at the parking garage.In this way, the database is also used to determine if an incoming customer has a reservation or is an ad hoc customer. If a customer has parked in the parking garage, the customer's parking spot location is also stored in the database.

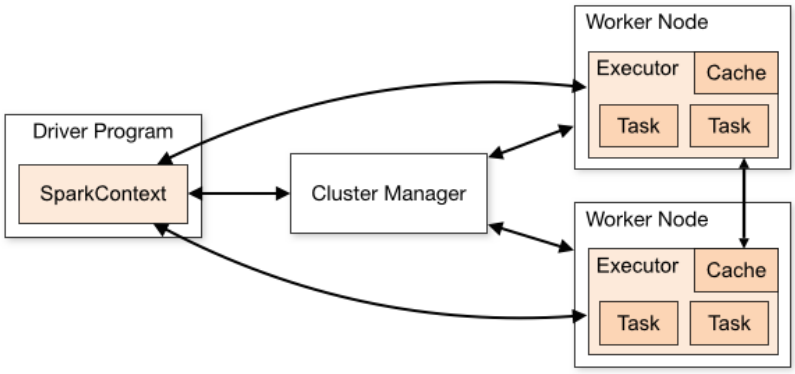


Figure 17: An Overview of the Spark Structure [1].

10.5 Network Protocol

For communication between client and server and for the structure of the web applications, our system uses Node.JS. Using node allows the server to load scripts inline with the web pages and redirect the user to different pages via routes based on requests they send to the server. This way the web page displays as a normal HTML page would, but the server is always "listening" for requests from the user. Using node also allows for many different packages to be utilized in the web applications, such as code which can securely handle and encrypt passwords or code which can be used to display the visual interface of parking spots.

10.6 Global Flow Control

10.6.1 Execution Orderness

The parking garage system is both event driven and procedure driven. The system is procedure driven because when the customer arrives at the parking garage, a sensor will read the car's license plate number and check to see if there is already a pre-existing reservation under that customer's name. Additionally, there are sensors on each floor to ensure that no more than one car is moving on each floor at a time and that each car is parked in its designated spot. The system is event driven because if a customer does not have a reservation, they will be required to input their contact information and choose a parking spot before entering the garage.

10.6.2 Time Dependency

The system uses timers to record the duration of a reservation/parking. The timers will be real time as they will be used to ensure that customers are not staying past their reservations. For reserved customers, the system will also give them a one hour grace period during which the customer can come to the garage to park at anytime with their reservation. After this grace period however, the parking spot will become unreserved and be open to other reservations.

10.6.3 Concurrency

Each car that is currently at the parking garage will have its own thread.

10.7 Hardware Requirements

1. Website Hosting Server
    - 33 MB disk space
    - 100 Kb/s connection speed
2. Database Hosting
    - 10 GB disk space
    - 80 GB memory for spark processing
    - 100 Kb/s connection speed
3. Garage Hardware
    - Camera (for license plate scanner)
    - Sensor for each parking spot
    - Keypad for security at each pedestrian entrance/exit
    - Tablet for entrance gate display

11 Algorithm and Data Structure

In the smart pricing system, we collected the data from the government. Those data cover the usage of the parking lots in Seattle from 2012 to 2017. We first analyze the parking lots usage, tying to use logic regression methods to estimate the usage over time (hour, day, month). This could help us define the time based demand function, which represents how the demand varies with the time (hourly, daily, monthly). There are other methods to model the time-based demand function, such as the neural network. However, based on our collected data, we found that the demand function is linear and the polynomial could fit the curve. Then, we try to use linear least squares regression and cubic spline interpolation to calculate the coefficient of the polynomial for the time-based demand function.

Besides the time-based demand function, we also proposed several price-based demand functions. The time-based demand function represents the relation between the parking lot usage and the time. Heuristically, it shows when the parking lot used most often and when the parking lot is used rarely. To design a dynamic pricing system, the goal is using the price to adjust the usage of the parking lot and make the most benefits on the fixed parking lots. When there is huge demand for the parking lots, the price will increase and demand should decrease. When there is small demand for the parking lot, the price should decrease to the minimum. So, based on this fact, we propose the price-based demand function. However, for different cases, the demand function could be impacted by the price differently. So, we proposed three demand models which is exponential, reciprocal and linear model for different scenario. Finally, we would calculate the real demand function based on the time and price. We calculate the total benefits by integral the demand function over time and price. So the final benefits function is the function of price over time. And, based on which, we could calculate the best price by solving the optimization problem of maximizing the benefits function.

11.1 Demand Function

Let $f(t)$ represent the time based function learned from real world data, where t indicates current time. Denote the price function at current time as $p(t)$. Although $f(t)$ can indicate the overall parking demand, we need to figure out the change of demand $d(t) = h(p(t), f(t))$ when given the dynamic price $p(t)$. This actually depends a lot on the driver's strategy. For simplicity,

we are aimed at a simple relation such that lower price results in high parking demand while higher price gets low parking demand. Here we list two types of relations:

- Linear relation. $d(t) = \log_{10} f(t) - k * p(t)$, where k is a constant coefficient. The logarithm of $f(t)$ is simply because that $f(t)$ can be very large values according to our fitted results.
- Expoential relation. $d(t) = f(t) * \exp\{-\lambda p(t)\}$, where λ is a constant coefficient. As $p(t)$ increases, the $d(t)$ decreases and vice versa.

11.2 Price Optimization

The profit gain is defined as a function $g(T)$ that means the profit gained from the beginning to some time T, which could be described as an integral process. The final objective function is as follows:

$$\max_{p(t)}\{g(T)\} = \max_{p(t)}\{\int_0^T p(t)d(t)dt\}, \tag{1}$$

where we propose to solve by simply taking the $\frac{\partial g(T)}{\partial p(t)} = 0$ as in [2].

11.3 Data Structure

The system is not using any data structure. Instead, all of the information regarding the user accounts on the website and information about the current state of the parking garage is being stored in a mysql database.

12 User Interface Design and Implementation

12.1 UC-1: Register

Navigation: 1 total click from the main page (click the register button)

Data Entry: total is indeterminant as keystroke per user will be different, 7 clicks

- On Register page enter the follow fields and click to next field

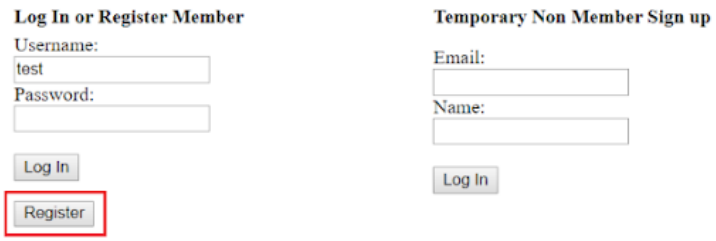


Figure 18: Register system.

  - Name
  - Username
  - Email
  - Password
  - Confirm Password
  - Phone
- Select the submit button after all fields are entered

Figure 19: Register Page.

There have not been any significant changes we have made to this simple registration process. It has already been implemented on our website.

12.2 UC-3: Reservation

Navigation: total 3 clicks, indeterminant keystrokes as follows

- Enter Login information as follows
  - Enter Username
  - Enter Password
- Click Login button (server sends user to account main page) shown below figure
- On account page click Make a Reservation Page(user sent to make a reservation page) shown below

Figure 20: Register Page.

Data Entry: total is 5 keystrokes, 2 - 7 clicks

- Select date of reservation by using calendar tool (number of clicks depends on date)
- Enter time of reservation in 24 hour time (5 key strokes → xx:xx)
- Select parking spot choice from the parking garage spot map tool (preliminary design shown below as the GUI has not been fully implemented yet)

There have not been any significant changes to our plan of utilizing this reservation GUI, but it has not been implemented yet. As of now, we have a field where customers can enter a date and select an available spot from a list of free spots (indeterminate keystrokes and 3 clicks to enter a time block

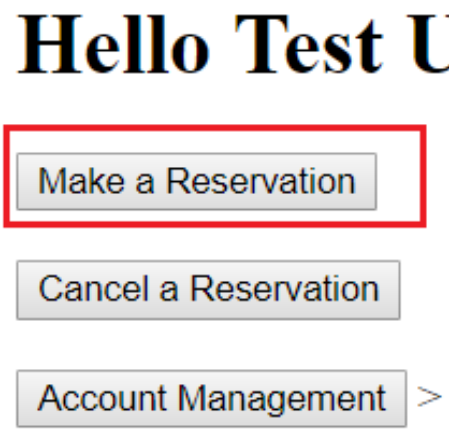


Figure 21: Welcome Page.

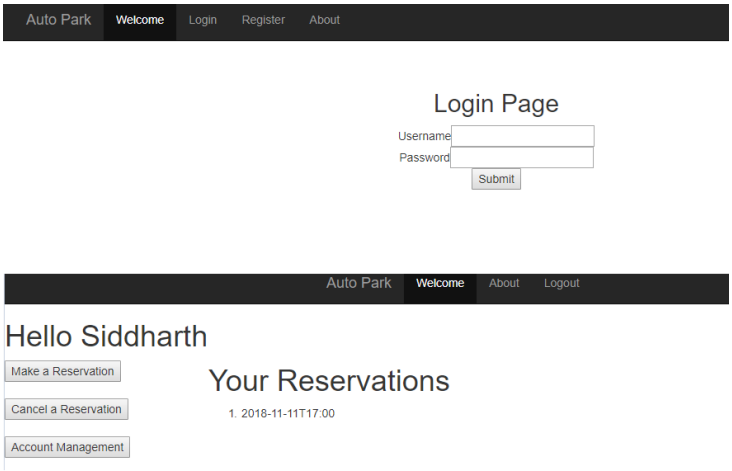


Figure 22: Login Page.

as well as confirm a spot to reserve). However, we plan on lessening the user effort by having the user select a date (2 clicks: one for the date and one for the month) and an available spot (2 clicks: one to select a spot and one to confirm the reservation).

12.3 UC-3: Login

Navigation: total 2 clicks, indeterminant keystrokes as follows

- Enter Login information on homepage (shown above) as follows
  - Enter Username
  - Enter Password
- Click Login button (server sends user to account main page)
- On account page click Make a Reservation Page(user sent to make a reservation page)

Data Entry: total is 5 keystrokes, 2 - 7 clicks

- Select date of reservation by using calendar tool (number of clicks depends on date)
- Enter time of reservation in 24 hour time (5 key strokes $\rightarrow$ xx:xx)
- Select parking spot choice from the parking garage spot map tool

This login process part of the website is already up and running. There have not been any significant changes besides ones intended to make the site more aesthetically pleasing.

13 Design of Test

13.1 Web Form Test Cases

1. Login: A registered user can input their username and password to log into their account
2. Registration: A new user can input their information and register for an account. All the data must be collect via a web form. Testing can be done via querying of database for the new user
3. Reservation: A logged in user can input time, day, and parking spot into a web form to reserve a spot. Users cannot reserve a spot that is already reserved for that same day. Testing can be done via querying of database for new reservations with correct user, date, time, and spot information

4. Cancel Reservation: A logged in user can cancel existing reservations. Testing can be done by checking database and validating that that the correct reservation has been cancelled

5. Update Information: A logged in user can update their user info via a webform. Users have to make their changes and click save to update their user info. Testing can be done via query of database and validating that the user data is up to date and the changes made via webform are reflected in the database

13.2 Garage Test Cases

13.2.1 Entrance gate

1. Main: main execution loop, this method should be able to continually accept customers coming in. Also it should send new and registered users to the correct handling function (handleNew or handleExisting). Testing can be done by hard coding a license string and testing to see if the function sends the user with the associated license string to the correct handling function

2. checkResTime: checks a list of reservations and determines if any are within the acceptable checkIn times for a reservation. Testing can be done via giving a list of reservations in the form of tuples (reservationID (int), userID (int), DateTime (datetime), CheckIn (boolean), ParkingSpot (int)). Then, by inspecting the result and making sure that either no reservation is returned in the case of no reservation check in allowed at the current time or a reservation is returned if the user has a reservation made for the current time

3. handleNew: handling function for new users. This function gets new user data, inserts into db and allows the new user to select a spot to do ad hoc parking. This function can be tested by sending in an unregistered license string and starting the function. After input of data the tester can query the database to see if the new user data has been inputted. Also the test should check to see if the function sends the correct spot based on user choice to the elevator handlePerson function

4. handleExisting: handling function for registered users. This function should allow users to choose adhoc or check in to a registration. The test should have an existing user come in and try both adhoc and reservation check in and see if the correct spot is sent to the elevator handling function

5. scanPlate: This function reads the license.jpg file in the project folder and determines the license plate. This function is easy to test as all it does is return the license string with the highest confidence for the machine learning algorithm. The test should have the function called on many pictures and see if the license string return is correct

13.2.2 AuthCode

1. generateCode: This function just generates a random 8 character string. It can be called and the output just has to be an 8 character string

13.2.3 Billing

1. Email: this function determines the amount of time parked and bills the user likewise. The bill is sent as an email. The function takes in user email and hours parked so the test should pass these values in. The test can be considered successful if the email is sent successfully with the correct charge

13.2.4 Elevator

1. handlePerson: this function takes in (spot, resID, plate) and extracts the floor from the parking spot and "brings" the person to the floor if the floor is ready. To test this we have to first send a floor that is ready to be parked and see if the function calls the bringCar method to the floor. Then, to test the waiting part, the function should be called again while the first car is still parking and see if the wait functionality is operating. If the first park is successful and the second is queued then the function is operating correctly

2. Bringcar: this function spawns a thread to perform spot verifcation. The only test required is to make sure this function spawns a thread and then returns to the main

13.2.5 Notifications

1. sendAuth: This function send a text to the specified user phone with the 8 character code. This function can be easily tested by supplying a Twilio registered number and seeing if the function call results in a text from the system

2. sendWrongSpot: This test is the same as the one above however the message should just be different ("You are in the wrong spot")

13.2.6 TrafficManagement

1. isFloorReady: this function just returns a boolean based on the global list floorsReady. It can be easily tested by ensuring that the floor (index) supplied is the correct value based on the floorsReady boolean list

2. readyFloor: this function takes in a floor as a parameter and sets the index in floorsReady for the specified floor to True. This can be easily tested by checking the state of the floorsReady list after the function is called. If the specified floor index is now true the function is successful

3. Unreadyfloor: this function takes in a floor as a parameter and sets the index in floorsReady for the specified floor to False. This can be easily tested by checking the state of the floorsReady list after the function is called. If the specified floor index is now false the function is successful

13.2.7 ExitGate

1. scanPlate: same test as the function in entrancegate

2. Get_datetime_hours: this function queries the database and determines how long a user has been parking for. It returns a double value representing how many hours a user has parked. Testing this function is a bit more complicated as it requires interactions with the database. The function looks for the first occurrence in the ParkingHistory table of the user's license where the ending time is not specified yet. So to test this the program code to test the function must generate some synthetic start time for parking in the database and see if the function returns the correct elapsed time.

3. Main: this function is the main execution loop and controls when the user leaves the garage. It automatically bills registered users and asks unregistered/new users for their email for digital receipt. To test this function, new and registered user data has to be synthesized. To test registered users allow the function to scan a registered license plate and if the function sends the correct bill then the test is successful. To test unregistered users, give an unregistered license plate string and enter a valid email. If the function sends the correct bill then the test is successful.

13.2.8 Spot Verification

1. checkParking: this function takes in (spot,resID,floor,plate) and checks if the correct spot has been occupied. This function behaves differently based on if the spot checking is adhoc or reservation. For adhoc the function just determines where the person parked on the floor they went to since adhoc users are allowed to park wherever they want to park. For reservations it makes sure the user parks in the specified spot and texts the user if the spot is not correct. There are 3 situations to test: adhoc parking, reservation successful parking and reservations unsuccessful parking. To test adhoc the occupy function in the simulation file can be used to simulate a sensor so if the function correctly watches a floor and determines that the adhoc user parked then the test is successful. The reservation success works the same way. For reservation unsuccessful the occupy function can be used to occupy the wrong spot and if a wrong spot text is sent then the test is successful

2. getUnoccupiedList: this function takes in the floor and returns a list of all unoccupied spots. This function is easily tested as the function returns data straight from the database so the data can be validated by comparing it to the database

3. unReserve: this function sets the reserved flag for a specified spot to 0. This function can be tested by checking the database after the function call

4. findSpotOccupied: this function takes two occupancy lists and determines what spot is occupied. It can be tested by sending occupancy lists and seeing if the result is the missing element

5. getPhonefromPlate: this function returns a phone number string based on the plate parameter. This function can be tested by a quick database check

14 History of Work, Current Status, and Future Work

14.1 History of Work and Current Status

As of now, we have created a website which allows users to register, log in to their accounts and make a reservation for a parking spot at any given date and time. Additionally, there is a license plate reader system in place that will read the license plate number of a car entering the garage, as well as a logic in place which allows users to switch spots if necessary. We have also implemented a system that simulates entering and exiting the garage along with a way to allow users to manually enter their license plate number in the interface at the front of the garage in case of system failure. We also have a notification system in place so that customers can be sent confirmation codes, security codes, and payment information based on their preference of SMS or email notification. Additionally, an algorithm has also been developed to implement the dynamic pricing system, which we can then replace with the flat pricing system we currently have in place.

14.2 Future Work

Future work for this project includes having to implement a GUI from which customers can view a realtime display of the parking lot and select their preferred spot. While the smart pricing system has been implemented, it still needs to be integrated with the rest of the modules and database.

15 Project Management

15.1 Merging the Contributions from Individual Team Members

In addition, we need to think about the uniformity of the diagrams in our report in order to appear more professional. We also need to uniformize the report formatting in order to maintain consistency between reports.

15.2 Project Coordination and Progress Report

Currently, we have implemented a website with basic functionality (register, login, and reservation). We have implemented most of the garage logic except for the user interface screen that customers see when they enter the garage. We have a license plate reader system that is currently in place that can also recognize ad-hoc users. Both ad-hoc users and registered users can park in the garage and are also able to switch spots if necessary. There is also a system in place that simulates entering and exiting the garage, as well as a way to allow users to manually input their license plate number in the interface at the front of the garage in case the license plate reader does not work properly. We also have a notification system in place so that customers can be sent confirmation codes, security codes, and payment information based on their preference of SMS or email notification.

We currently have a flat-rate payment system in place; however, we are working on implementing the dynamic pricing system we have completed a lot of research on already. The algorithm will be based on demand and time of day. In addition, we are working on implementing a manager account for the garage so that owners or managers can manually input information and alter some settings in the garage such as price for special circumstances. We are also working on creating the visual interface that will allow users to see all of the available spots in the parking garage. Even though we have a system in place that allows users to enter and exit the parking garage and switch spots, we are currently working on integrating and refining it.

15.3 Breakdown of Responsibilities

All members are contributed equally to the report.

Corey Chen:

- Database Functionalities
- CSS and HTML formatting for the website
- Login and Register modules
- Integration Testing
- Website Navigation
- Website/Database Communication

Chunhua Deng:

- Email notification module
- Exit garage module
- Cubic spline interpolation for fitting curve based on the data.
- Propose the price-based demand function.
- Total benefits Optimization.
- Final dynamic pricing system optimization
- Integration of the report

Jonathan Garner:

- SMS Notification Module
- Authentication Module
- Integration Testing
- Database Interactions
- Spot Status Module
- Parking Map Visual Interface

Siyu Liao:

- Data Retrieval Module

- Data Cleaning Module
- Data Processing Module
- Data visualization
- Calculate the price based demand function with multiple model.
- Profit optimization based on the demand function.

Siddharth Musale

- Database Creation and Management
- Entrance Gate
- Exit Gate
- Notification and Email module integrations
- Traffic Management
- Elevator
- Spot Verify
- Reservation page
- Open ALPR api integration
- Simulation
- All unit testing

Ridhima Sakhuja:

- Login and Registration (Website) Modules
- CSS and HTML formatting for website
- User info/password input validation
- Integration Testing
- Website Navigation/Database Communication

Xianglong Feng:

- Data cleaning
- Logic regression for fitting curve based on the data.
- Propose and design the price-based demand function.
- Proposed and design the multi-module based method for dynamic pricing algorithm
- Proposed and design the dynamic updating method for price-based demand function
- Propose and design the ease out quart function for price-based demand function in peak time
- Propose the design the ease out quart function for price-based demand function in adjustable time
- Design the solution to solve the optimized price based on time
- Total benefits Optimization.
- Final dynamic pricing system optimization
- Data Visualization for final results.

16 Dynamic Pricing for AutoParking

The dynamic pricing algorithm is the soul of this smart parking system. The idea of using dynamic pricing algorithm is to 1) best leverage the parking resource and make the best benefits; 2) make the parking resource convenient for all user.

Like the flight tickets in holidays, the parking lot in the rush hour is quite limited comparing with the need. The parking price for the rush hour should increase to make the best benefits. And also, just like the flight tickets in the other time when there are adequate number of tickets for so few traveller, the manager of the parking garage should decrease the parking price to attract users to increase the benefits and so they could better maintain the system.

However, currently due to the lack of real parking data and experiment of dynamic pricing system, most of the parking lots in street and garage are using fixed parking price. There are some parking system provides different parking price based on the location (i.e., the parking price in the central of the downtown is higher than that in the outsides of the downtown.). But in the same parking slot, the price is the same for the whole day in a year.

The other advantage of the dynamic pricing system is making the limited parking resources real convenient for all users. This part may be difficult to understand. Take the flight tickets in the holiday for example, usually it's hard for someone to buy one tickets in such time. However, there may be cases that someone really needs to take the flight for a surgery, international conference and so on but can not get one. This is not real convenient because of the one who real needs can't get one. The dynamic pricing system will leverage the price to find the people who really needs, and make sure that anyone really need a parking slot could find one.

For the parking system, the case may be more complicated. In the peak time, if the price increases, the people that want to use the parking garage will decrease. If the price is very high, then, people will find alternative way such as taking the subway, bus, taxi or even ride a bike. For some other cases, such as business work, seeing doctor for patients who are inconvenient, people still want to use the parking garage even the price is higher than the taxi. For such cases, they could not find a parking lot in the rush hour if the system uses a fixed pricing system. A good system should meet the need and find the one who really need and make sure they could get the service.

16.1 Dynamic Pricing Related Work

The idea of dynamic pricing has been widely used in markets nowadays. In the case of energy cost [3], dynamic pricing can reduce peak demand of a city, and decrease resource cost, for example extra investment in handling the peak hour, thereby improving social welfare. With such observation, dynamic pricing strategy in the parking application has been investigated in many studies. San Francisco Park (SFPark) [4] uses dynamic pricing for traffic congestion control, where they adjust price proportional to the parking resource utilization. iParker [5] aims at increasing parking utilization, revenue and lowering drivers' cost (searching and walking time). Assuming drivers know how the price and parking occupancy change along time and they would choose accordingly, [6] finds that the optimal charging results in

the balance between the cruising time and parking space convenience. Although many existing works present their methods to achieve the optimum solution, many of them give lots of theoretical assumptions and are lack of tests over real world data. In this project, we aim at capturing parking demand from real world data and use the temporal and spatial properties to find the pricing strategy maximizing revenue.

16.2 Requirement of Dynamic Pricing

The dynamic pricing system could adjust the number of customers and make the best benefits by changing the parking price based on the demand of the parking spaces. Based on the real parking data, collected from the Seattle government's website, the demand is heavily depend on the time of a day. As is shown in Figure [**?**], the parking lots reach the peak from 10:00 to 15:00, and the parking lots are rarely used in the night.

The second factor that could impact the demand for parking lot is the parking price. A lower parking price will attract more people to drive here and park here while a higher parking price will force people find alternative method to commute.

So, based on the two factors above, we can find that the demand for parking lot is a function with both price and time. The demand function varies based on the location along the time and price. For example, the parking lot around the railway station shows a different demand function based on time comparing the parking lot in the center of the city. In the center of the city, the parking lot will reach its peak in the noon, while the parking lot around the railway station is heavily depending on the time table of the train.

As a result, given a parking garage, we should firstly collect enough parking data and the parking variation on different prices. Then, we should use logic regression to calculate the model for the price and time based demand function, with which the system could modify the current price to best use the parking resources and make the best benefits.

16.3 Parking Records of Seattle

There are public on-street parking records released by the City of Seattle Department of Transportation (SDOT), which starts from January 2012 [7]. We take the data from January 2012 to September 2017, which is around

5.3GB with 62,327,970 records. Several records are shown in Figure 23 as examples and corresponding data format is explained in Table 9.

	A	B	C	D	E	F	G	H	I	J
1	TransactionId	TransactionDateTime	TransactionDate	timeStart	timeExpired	Duration_mins	Amount	PaymentMean	MeterCode	ElementKey
2	13968676	2012-01-01T22:07:59Z	1/1/2012	22:07	23:22	75	2.5	COINS	10015002	25706
3	13968818	2012-01-01T23:30:59Z	1/1/2012	23:30	1:30	120	4	CREDIT CARD	10023002	25710
4	13968824	2012-01-01T22:45:59Z	1/1/2012	22:45	0:45	120	4	CREDIT CARD	10096002	9357
5	13968660	2012-01-01T22:51:59Z	1/1/2012	22:51	0:51	120	4	CREDIT CARD	10210002	25718
6	13968821	2012-01-01T23:28:59Z	1/1/2012	23:28	0:38	70	2.25	CREDIT CARD	10223002	2789
7	13968820	2012-01-01T23:50:59Z	1/1/2012	23:50	1:50	120	3	CREDIT CARD	12093002	39393

Figure 23: Example of On-street Parking Data in Seattle

Table 9: Data Format for On-street Parking Data in Seattle

Name	Type	Description
TransactionId	integer	Unique identifier number for a record
TransactionDateTime	timestamp	Date and time of the record
TransactionDate	timestamp	Record date
timeStart	string	Parking starting time
timeExpired	string	Parking ending time
Duration_mins	integer	Length of parking in minutes
Amount	double	Payment amount in dollars
PaymentMean	string	Payment in credit card, coin, phone, etc.
MeterCode	integer	Pay station identifier
ElementKey	integer	Street segment identifier

16.4 Parking Demand Analysis

We capture the parking demand of the Seattle city by counting number of parking transactions that start in a pre-defined time range, which can be hourly, daily, weekly and monthly. For example, in the case of hourly parking demand, if a transaction started on 21:07, it is considered as one parking request within 21:00 - 22:00. The total number of parking requests of the time range is defined as the parking demand. Given the large data volume, we get all the statistics using Spark [8] on Ubuntu 16.04 with 20GB memory.

Figure 24 shows the hourly parking demand of the Seattle City. It can be seen that the rush hour is around 7:00 - 8:00 when the demand peak appears.

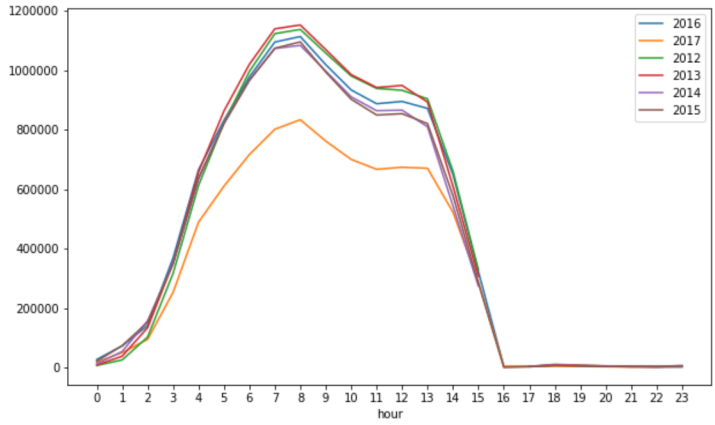


Figure 24: Hourly Parking Demand of the Seattle City

However, after 15:00, parking demand significantly drops down. This can be attributed to that people usually leave the city and drive to home at that time, so there are much less parking demand.

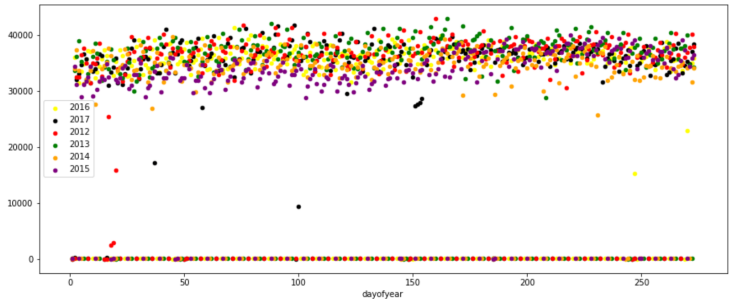


Figure 25: Daily Parking Demand of the Seattle City

Figure 25 shows the parking demand of every day throughout a year. It is found that every year remains the same parking demand for each day. Moreover, there seems to be a line formed by dots in Figure 25, which can be explained by Figure 26. There are almost none parking records tracked on Sunday, which means that every Sunday our calculated parking demand would be close to zero, as shown in Figure 25.

We also draw the monthly parking demand as in Figure 27. It is found that every year the monthly parking demand remains the same. However, there seems less parking demand in January, September and December. Since

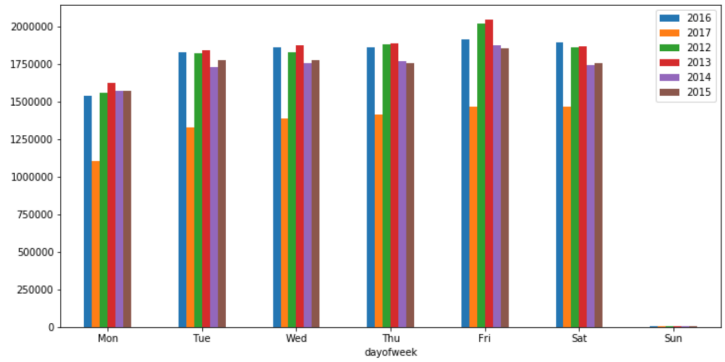


Figure 26: Weekly Parking Demand of the Seattle City

Seattle is quite close to Canada and in the north, it may be the reason that in winter people drive less than other seasons.

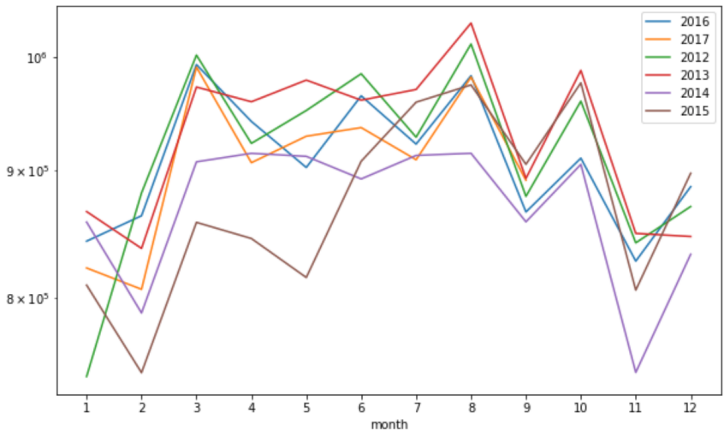


Figure 27: Monthly Parking Demand of the Seattle City

16.5 Parking Demand Curve Fitting

16.5.1 Model selection

There are several models for the collected parking data, such as the Gaussian models, polynomial models, Poisson models and the neural network based models.

The Gaussian and Poisson models heavily depended on the two or three parameters to represent the sequence based progress. As we could observed from our data, we found that the data is more like a linear system. Considering that the neural network works well for non-linear system, we try to use polynomial models to estimate the time based demand function.

Given the parking data, we have several methods to estimate the parameters for the polynomial based model for the time-based demand function, such as the linear least squares regression and cubic spline interpolation. Here, in this project, we use the linear least squares regression to calculate the coefficients of the polynomial based model for the time-based demand function.

16.5.2 Linear least squares regression

In linear least squares regression, we got the real data as time (t) and the demand (y). So we got several data point $(t_1, y_1), (t_2, y_2)...(t_n, y_n)$. We are seeking for the polynomial

$$y = a * t^6 + b * t^5 + c * t^4 + d * t^3 + e * t^2 + f * t^1 + g, \tag{2}$$

which fit the data. In the other words, we want to minimize the error.

Let the real data be Y and the error is $E = Y - y$; The squared error is$E^2 = (Y - y)^2$. The total error for all the data set is:

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (Y_i - y_i)^2 \tag{3}$$

The goal is to minimize the error and so we will calculate the partial derivative for a,b,c,d,e,f and g. Calculate the value by set the partial deriva-

tive equal to 0.

$$\frac{\partial Q}{\partial a} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^6\right)\right) \tag{4}$$

$$\frac{\partial Q}{\partial b} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^5\right)\right) \tag{5}$$

$$\frac{\partial Q}{\partial c} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^4\right)\right) \tag{6}$$

$$\frac{\partial Q}{\partial d} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^3\right)\right) \tag{7}$$

$$\frac{\partial Q}{\partial e} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^2\right)\right) \tag{8}$$

$$\frac{\partial Q}{\partial f} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*\left(-t^1\right)\right) \tag{9}$$

$$\frac{\partial Q}{\partial g} = 2\sum_{i=1}^{n}\left(Y_i - \left(a*t^6 + b*t^5 + c*t^4 + d*t^3 + e*t^2 + f*t^1 + g\right)*(1)\right) \tag{10}$$

16.6 Fitting Curve Result

The fitting curve for weekly and hourly parking data is shown in Figure 6 and Figure 7. As can be seen from the two pictures, the polynomial based model can fit the parking data. The weekly parking data could be fitted completely and the hourly parking data shows some small miss match which will only introduce small impact.

We also calculate the fitting curve for the monthly data. So in total we could obtain the fitting curve for monthly, weekly and hourly parking data.

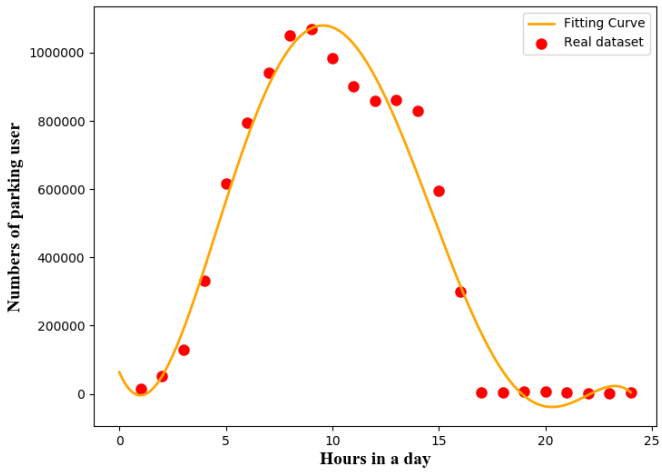


Figure 28: Curve Fitting for Hourly Parking Demand

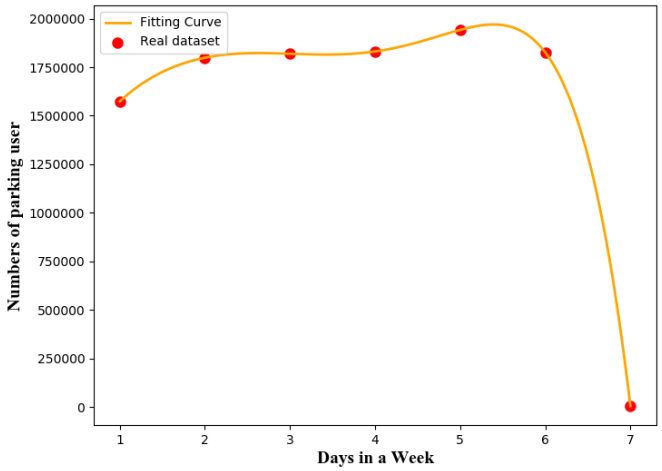


Figure 29: Curve Fitting for Weekly Parking Demand

For the monthly and weekly parking data, we will calculate the coefficient. We will spend the most effort in solving the optimize problem based on hourly parking data since the parking price will be calculated based on hour. Assume the coefficient for weekly and monthly is $E_w(w)$ and $E_m(m)$, the

final time-based demand function is shown below.

$$D_t(t) = E_w(w) * E_m(m) * D_h(t) \tag{11}$$

16.7 Reservation Based Model

In [6], researchers assume that one driver only books one spot each time and group reservation is not considered in this work. Drivers can request reservation anytime during service. The reservation will be confirmed as long as there are available spots, which follows the assumption as in [6] that drivers care more about parking availability than spending time to figure out how to save money. Furthermore, since drivers' decision strategy is unknown to us, this work won't consider such factors and this handling strategy can also be found in similar works like [6]. Given assumptions above, we would like to adopt a mathematical model for our parking garage as presented in [6] which is defined as follows.

Let C denote the number of parking spots and each spot can serve N periods. X^t is a N-dimensional vector containing number of available spots for each period at booking time t, where $0 <= X_i^t <= C$ and $i = 1, \ldots, N$. Number of vacant spots can be utilized to help determine the price. For example, we could adjust to lower price when there are more available spots while to higher price when there are less. Thus, It is natural to define the N-dimensional pricing vector P^t that relies on X^t.

On the other hand, drivers' demand can be easily affected by the price. Let's define a function $D^t_{[u,v]}(E[P^t_{[u,v]}])$ to represent the demand from u-th period to v-th period at booking time t, which mainly relies on the corresponding expected price. For simplicity, $E[P^t_{[u,v]}] = \sum_{i=u}^{v} P_i^t/(v-u+1)$ as presented in the [6].

Assume that reservation can be modeled by a poisson process. The intensity can be defined as $\lambda^t_{[u,v]}(X^t, P^t) = A^t_{[u,v]} D(X^t, P^t)$, where $A^t_{[u,v]}$ is either 0 or 1 to indicate if there exists at least one available parking spot from u-th period to v-th period at booking time t. Thus, the total reservation rate is given by

$$\Lambda(X^t, P^t) = \sum_{u=1}^{N} \sum_{v=u}^{min\{N, u+n\}} \lambda^t_{[u,v]}(X^t, P^t), \tag{12}$$

where the n means maximum periods that a driver can choose. If there is not, it could be simply set as N. As a result, our objective function is defined

as following.

$$V(X^t,t) = \max_{P^t}\{Q_0 V(X^t, t-\Delta t) + Q_1[\sum_{i=u}^{v} P_i^t + V(X^t - e_{[u,v]}, t-\Delta t)]\}, \quad (13)$$

where V is the maximum revenue from 0 to booking time t, Δt is sufficient small during which at most one booking reservation arrives, $Q_1 = \Lambda(X^t, P^t)\Delta t$ means the probability of successfully booking, $Q_0 = 1 - Q_1$ then means the probability of not booking the spots, and $e_{[u,v]}$ is a zero vectors with u-th entries to v-th entries as ones. By taking Δt to the limit 0, we have:

$$\lim_{\Delta t\to 0}\frac{V(X^t,t)}{\Delta t} = \max_{P^t}\{\sum_{u=1}^{N}\sum_{v=u}^{v=\min\{N,u+v\}} \lambda_{[u,v]}^t(X^t,P^t)(\sum_{i=u}^{v} P_i^t - V(X^t,t) + V(X^t - e_{[u,v]}, t))\}, \quad (14)$$

where we could solve the P^t by taking the derivative as 0 and then use it to express $V(X^t, t)$.

For example, the single period model in [6] assumes an exponential relation between demand and the price, which is expressed as $\lambda(P^t) = a\exp\{-P^t\}$. By setting the gradients as in 14 to zeros, we could solve the expression of price function as

$$P(x,t) = 1 + V(x,t) - V(x-1,t). \quad (15)$$

Moreover, if we plug-in the price function, we achieve an equation as in [6]:

$$V'(x,t) \times \exp V(x,t) = \frac{a}{e}\exp V(x-t,1), \quad (16)$$

which has the solution as following given the boundary conditions $\forall x, V(x^0, 0) = 0$ and $\forall t, V(0,t) = 0$:

$$V(x,t) = \ln[\sum_{i=0}^{x}(\frac{a}{e}t)^i\frac{1}{i!}]. \quad (17)$$

We simulate the results by setting T ranging from 2 to 34, number of service periods as 24, value of coefficient a as 8, which is the same as in the paper [6]. We compare with the fixed pricing model as shown in the Fig 30. In this experiment, dynamic pricing is slightly better than fixed pricing that gains revenue linearly with book time. However, we also found that with high settings of fixed price, dynamic pricing in this reservation based model will be less effective as fixed price.

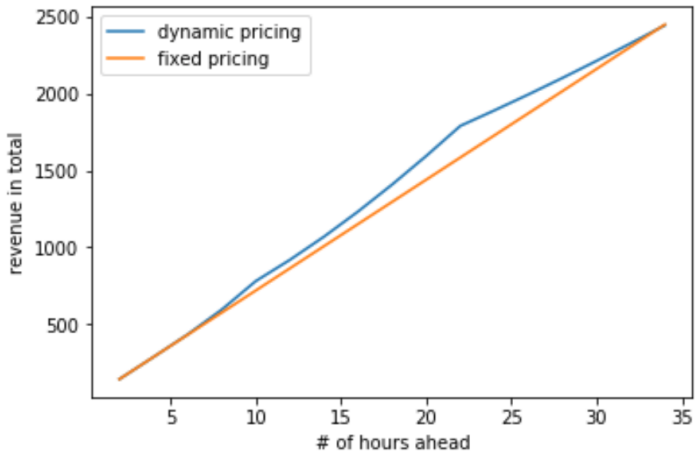


Figure 30: Reservation based Model Revenue

16.8 Our proposed model: Multi-models based dynamic pricing

Comparing the previous works, the main problem is that they want to use one demand model for a general use case. This demand function will includes the price and time as the variables and calculate the price by solving the optimal solutions. There are also several works that introduces several demand functions to fit different use cases. However, for any one demand function, the demand function will not be changed as long as it is chosen as the demand function, which means the demand function will run forever to calculate the price.

In real world scenario, there should be several demand functions. For example, if one demand function works very well for many weeks but there would be a football match next week, the demand function will definitely not work for the next week.

By observing the user data and the real world use case, we proposed the multi-models based dynamic pricing algorithm. In our algorithm, we assume there are many demand function for many use cases and the system is build with several models. In different day, different time period we could use different demand function. The pricing system will be collecting the user data all the time and changing the demand function in the real time.

In this paper, we use the hourly parking demo to show how our algorithm

works. For the model of the weekly and monthly, we will still calculate the fitting model and calculate the coefficient to tune the hourly based pricing system.

16.8.1 5 modules based pricing system for parking in a day

As shown in Figure 6, we can see that the parking lots reach its peak from 7:00 to 13:00 while it is rarely used in the night from 14:00 to 3:00 of the next day. So we divide the parking into three part. The peak time, the adjustable time and the Free time. In the peak time, the parking lot is very crowed and people will have difficulty finding a parking lot. In the adjustable time, we have adequate number of parking lots and people do not have difficulty finding a parking lot. In the free time, there are huge number of the parking lots available.

For the peak time, what we should do is increasing the parking price to reduce the users, so that the people who real need to park will find a parking lot and the traffic burden will decrease. In the adjustable time, we should decrease the parking price to attract people. In the free time, it is the time people leave and nothing could be done to change the situation.

So, in the end, we could divide the whole day into five parts which are free time (0:00-3:00), adjustable time (3:00-7:00), peak time (7:00-13:00), adjustable time (13:00-16:00) and the free time (16:00-24:00). In different part, we will use different price based demand function to tune the final demand function.

16.9 Price based Demand function for each parts

There is no real user data for price-based demand function. The price based function is more complicated and could only be obtained by carrying out experiments on real customers. However, since no dynamic pricing algorithm has been implemented in real word, there is no data about the price impact, and so we could not obtain the price based demand function based on real data.

So in this project we could only proposed several price-based demand function for readers to choose. Actually in a real system, we think the use case should be the same which is that the system has several price-based demand function and the system will dynamic update the demand function it is using based on the current user data.

In general the demand should have a negative relation to the price. For different use case, it could be different, such as the linear, exponential and so on. Here we use the ease out quart function to simulate the price base function.

16.9.1 The price-based demand function for Peak time

In this part, we will increase the price to decrease the number of user. And we will use the ease out quart function to model the relationship between the price and the parking lot user. The ease out quart function is described as following:

$$D_p(p) = -C * \left(\left(\frac{p}{Max\{P\}} \right)^K - 1 \right) + D \tag{18}$$

For quart function K=4. Here we normalize the price to reduce the impact of the price scale and make it more general. This function could be tuned by C and D. So, once we collect data in real world, we could update those parameters to fit the real price based demand function. Also, in this function, K is also adjustable in order to fit the real data.

As can be seen in figure 8, the number of the parking lot user is decrease as we increase the price. At the first stage, where the increasing coefficients is not too high (less than 0.5) the number of the user does not decrease too much. However, if the price coefficient increases too much, such as 0.8, the number of the user could drop to a half. In the price based demand function, we could also set the C to estimate the price that will make all users refuse to use the parking lot.

16.9.2 The final demand function for Peak time

Given the time based demand function:

$$D_t(t) = a * t^6 + b * t^5 + c * t^4 + d * t^3 + e * t^2 + f * t^1 + g, \tag{19}$$

We first select the the peak time period by setting a threshold (TS) as the peak usage. The idea for this part is to use the price-based demand function to calculate a new price that could decrease the parking usage to an acceptable value (around TS) and maximum the benefits.

$$TS - \Delta \leq D_p(p) * D_t(t) \leq TS + \Delta \tag{20}$$

$$Max\left\{\int_t^T D_p(p) * D_t(t)\, d_t\right\} \tag{21}$$

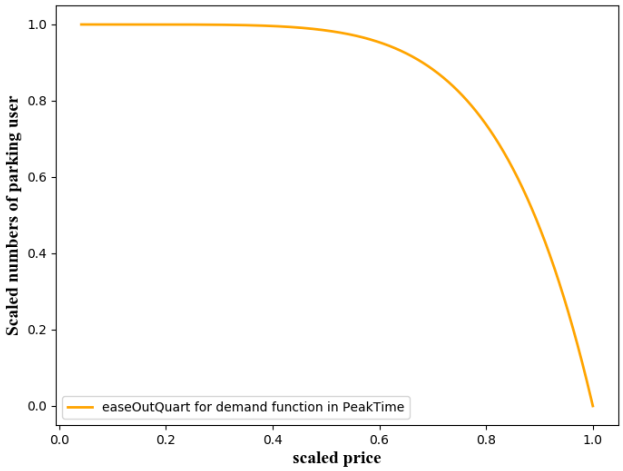


Figure 31: Price-based demand for Peak time

As can be seen in Figure 8, by increasing the price, we could adjust the number of parking lot user to an adjustable value. At the same time, we could make sure that the final benefits will increase comparing with the fixed price. For this part, we will prove our method in the conclusion section.

16.9.3 The price-based demand function for Adjustable time

In this part, the parking resource is not fully used. There is enough space for more users. As a result, we could decrease the parking price to attract more users to park in the parking garage. At the same time, we should also make sure that we can have better benefits.

First to model the price-based demand function for the adjustable time period, we still leverage the ease out quart function. The new function is shown bellow:

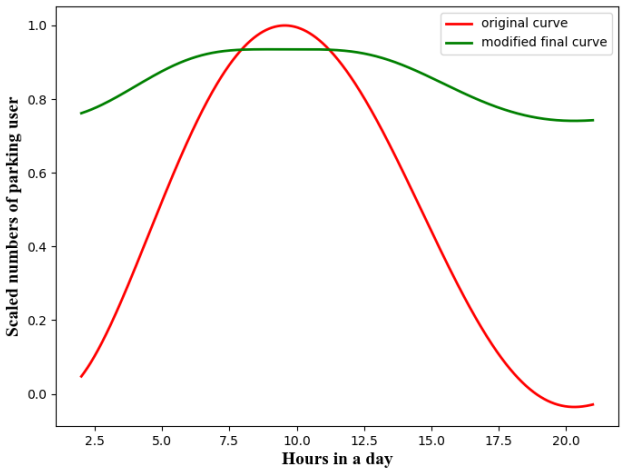


Figure 32: Modified Parking Demand in Peak Time

$$D_p(p) = C * \left(\left(1 - \frac{p}{Max\{P\}} \right)^K - 1 \right) + D \tag{22}$$

The price-based demand function for the adjustable time period curve is shown in figure. It is very obvious that if we decrease the price little (i.e., fewer than 0.5), the increase of the user is also very few. If we decrease the price with a huge amount, there will be an obvious increase in the demand function.

Also by collecting the real data from the physical world, we could update the parameters such as the K, C and D. If the demand is less sensitive witht price, we could increase the K and decrease C.

16.9.4 The final demand function for Adjustable time

Reducing the parking price will introduce the risk of decreasing the final benefits comparing with the fixed price. So, in this part we should make sure that we could still increase benefits by increasing the parking lot users, while at the same time we should make sure the number of the parking user will not reach the peak threshold (TS). So the final price should be limited by the following two equation.

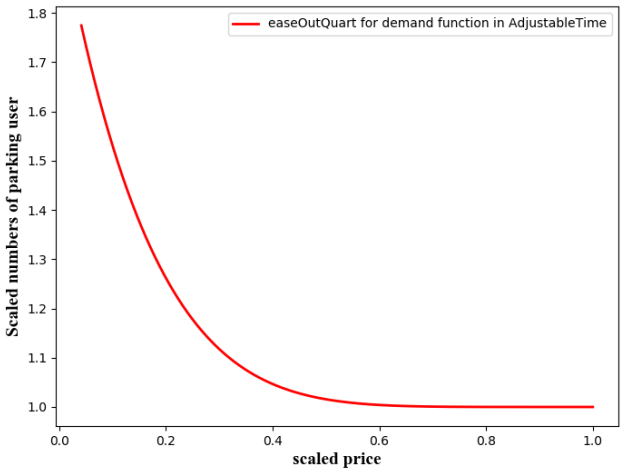


Figure 33: Price-based demand for Adjustable time

$$TS \leq D_p\left(p\right) * D_t\left(t\right) \tag{23}$$

$$Max\left\{\int_t^T D_p\left(p\right) * D_t\left(t\right) d_t\right\} \tag{24}$$

In this system, there will be two time periods that belong to the adjustable time. So we implement the price-based demand function in the two parts to modify the original time-based demand function. The two modified function curves are shown in figure 11, and figure 12.

As can be seen in the two pictures, for the earlier one, the original curve shows little usage at the beginning and the increment percentage is huge, and at the end of the curve, since the usage is almost reaches its peak time, the increment will not increase. For the latter one, the usage is around the peak at the beginning and so the increment is not that obvious while for the later part the increment is much more since the usage is relative smaller comparing with that of the beginning.

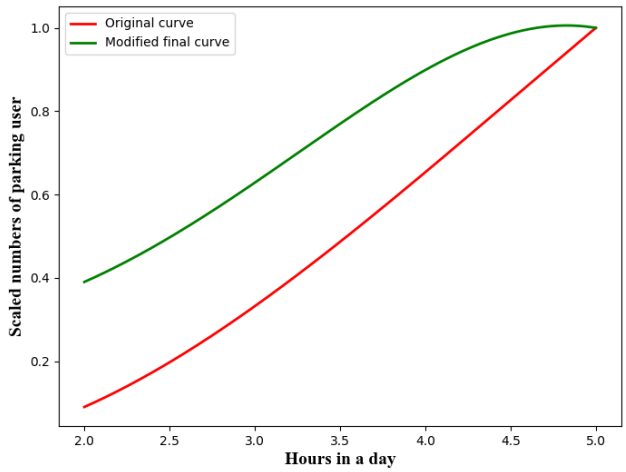


Figure 34: Modified Parking Demand in Adjustable Time (earlier)

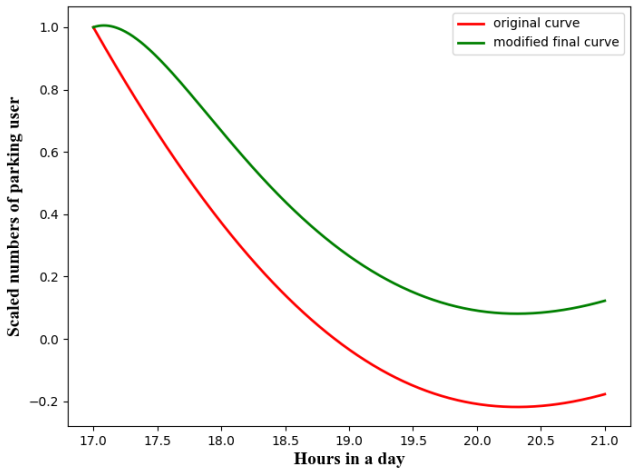


Figure 35: Modified Parking Demand in Adjustable Time (Later)

16.10 The final demand function for a day

So, when we calculate the final demand function, we will divide it into 5 parts as we described before. And we will calculate the new demand function after we applied the price-based demand function. Then, in the last step, we will connect the five parts together as our final demand function.

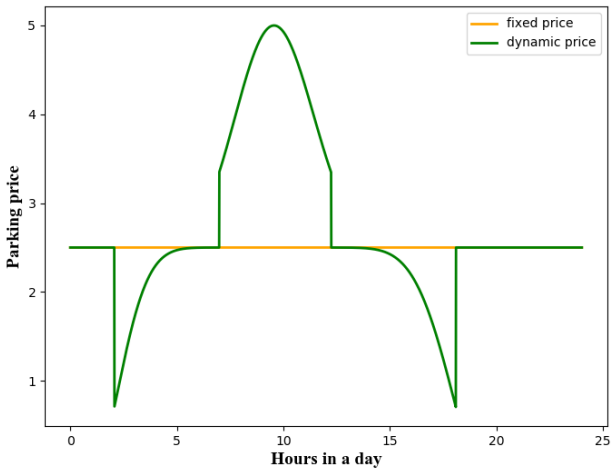


Figure 36: Modified price:[U+FF1A]X axis is the 24 hours in a day, y axis is the new price

The figure 36 shows the new price we calculated by solving the previous optimize problem. It can be seen that, in the free time, the night, we could use the fixed price or we do not charge any fees. In the adjustable time, the morning and the late afternoon, we decrease the price. And at the beginning, the price is very low and it increases when the parking user increases. For the peak time in the middle of the day, we increase the price.

As is shown in figure 14, the orange curve is the curve for the original demand function by time. The green curve is the final curve modified by he dynamic pricing algorithm. It is obvious that at the adjustable time periods, the demand is increased comparing to the original one. At the peak time, the demand is decreased by the dynamic pricing comparing to the original one.

Based on the curve shown in Figure 37, our dynamic pricing algorithm could adjust the amount of parking user, so that it will increase the number

of parking use at the spare time while reduce the number of parking user at the peak time by dynamically changing the price.

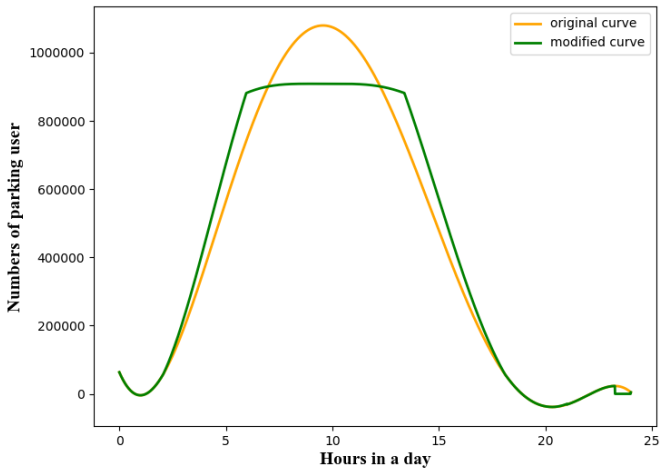


Figure 37: Modified Parking Demand based in Time: [U+FF1A] X axis is the 24 hours in a day, y axis is the demand forb parking garage

16.11 The final benefits for a day

Another issue we concerned is the final benefits, because decreased parking user and the decreased parking price will decrease the final benefits. In our project, we can make sure we will increase the final benefits by solving the optimal problem we discussed in section 7.

After we implement the price-based demand function for the five parts, we will calculate the final benefits by integrating the parking data for a whole day.

As can be seen in figure 14, the orange curve is the original benefits with a fixed price while the green curve is the final benefits modified by the dynamic pricing algorithm.

It is very obvious that our dynamic pricing algorithm will increase the final benefits comparing with the fixed pricing. The increment at the adjustable time is limited but it increase a lot at the peak time, which is what we expected at the beginning.

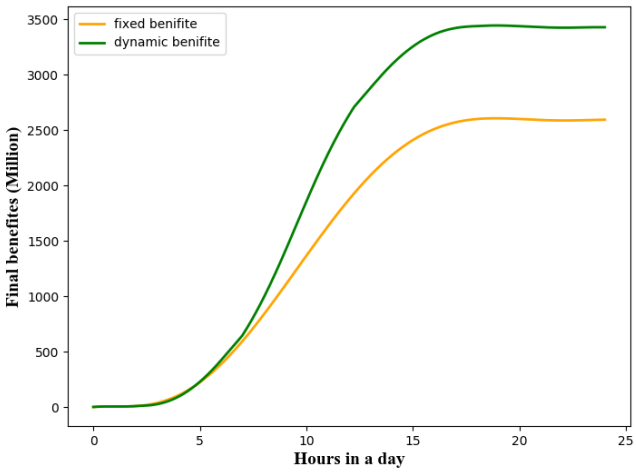


Figure 38: Modified and Original benefits[U+FF1A]X axis is the 24 hours in a day, y axis is the benefits obtained from the parking garage

16.12 Dynamic Pricing Conclusion

By showing all the results, it is obvious that our dynamic pricing system could adjust the parking user by changing the parking price, making more people to park at the spare time and fewer people to park at the peak time. And, more importantly, even we need to reduce the price or decrease the number of parking user at some time, our results show that the parking system still increase the final benefits.

To work with the parking system, the algorithm will calculate the estimated parking price and show the price to the customer when there is customer coming to park. The customer will pay with the current price. Then after the pre-set time period, the pricing algorithm will collect the parking user information from the data set. With these data, the pricing algorithm will check the price-based demand function and update the parameters in the demand function. So, the new customer will have different parking price based on the coming time and current number of user.

References

[1] "Spark overview." https://spark.apache.org/docs/latest/cluster-overview.html. Accessed: Nov. 4, 2018.

[2] Q. Tian, L. Yang, C. Wang, and H.-J. Huang, "Dynamic pricing for reservation-based parking system: A revenue management method," *Transport Policy*, vol. 71, pp. 36–44, 2018.

[3] A. Faruqui, "The ethics of dynamic pricing," in *Smart Grid*, pp. 61–83, Elsevier, 2012.

[4] "Sfpark." http:// sfpark.org/. Accessed: Nov. 1, 2018.

[5] A. O. Kotb, Y.-C. Shen, X. Zhu, and Y. Huang, "iparker-a new smart car-parking system based on dynamic resource allocation and pricing.," *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2637–2647, 2016.

[6] Z. S. Qian and R. Rajagopal, "Optimal dynamic parking pricing for morning commute considering expected cruising time," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 468–490, 2014.

[7] "Sdot parking records." http://wwwqa.seattle.gov/Documents/Departments/SDOT/ParkingPr Accessed: Nov. 1, 2018.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets.," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.