Blockchain and Docker Assisted Secure Automated Parking Garage System

Group 4:

Luan Tran	- lmt185
Khanh Nguyen	- ktn31
Shalini Choudhury	- sc1822
Tan Ngo	- ttn64
Duc Nguyen	- dhn25
Zhouyang Xiao	- zx150
Nainil Patel	- npp71

Project blog : https://sites.google.com/scarletmail.rutgers.edu/ruparking/main?authuser=1

SCAN ME

Table of Contents

Contribution breakdown:

Task	Shalini	Duc	Luan	Tan	Khanh	Nainil	Zhuoyang
Customer Statement	70%					15%	15%
Glossary of Terms	30%	70%					
System Requirements					40%		60%
Functional Requirement Specification	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Effort Estimation(Use case points)		80%		20%			
Domain Model Analysis					15%		40%
Interaction Diagram	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%	14.3%
Class Diagram & Interface Specification			40%	30%	10%	20%	
System Architecture & System Design	20%	10%		10%	20%	25%	15%
Algorithm & Data Structure		16.7%		16.7%	16.7%	50%	
User Interface and Implementation			100%				
Design of Tests		50%				25%	25%
History of work, Current status, Future works	50%						50%
Project Management	100%						

1. Customer Statement

Parking lots, garages and on-street parking spaces are often sources of anxiety for drivers, and currently there are not many ways to address these problems. Parking garages either use an attendant or a toll gate before entering a lot, making the process of entry and leaving the parking lot extremely slow and cumbersome, assuming that the parking lot even has space available; if it doesn't, then the driver has to back out, in potentially high traffic, causing undue stress. If the driver happens to get in to the lot, he has to take the extra time to find a parking spot and on finding one it might not be suitable for the type of vehicle he wants to garage. Another issue with parking is that the driver has no idea as to how much time he has remaining on his meter and usually overshoots or undershoots his target time, resulting in either a ticket or a waste of money spent on extra time. Finally, security is a major concern for all the customers since according to the Federal Bureau of Justice Statistics, more than 11 percent [1] of U.S. property crimes occur in parking garages and lots. Hence the parking garages are not secure and also the personal information of the customers ,i.e., the credential and payment details may be compromised.

To address the aforementioned problems this work proposes blockchain and docker assisted secure automated parking garage system.

With the aim to provide our customers with a streamlined process of easy and flexible parking. This project would implement enhanced automated garaging system to enable customers to view and securely reserve suitable spots for parking their vehicles. Customers through the user interface hosted on our website will be able to reserve a parking spot from the time of check-in and then extend the timing if required from the web interface by login into their account. The same slot will be updated in the parking spot availability list as soon as the customer reservation time expires and he does not renew. This work creates an intuitive way for users to quickly locate available parking spots in a busy community, while also incentivizing users to only park during their designated time and not use the buffer period before being charged extra.

For an efficient automated garaging system multiple novel ideas has been included which brings in great assistance to both the customer and the parking garage manager. The new customers are provisioned with one time registration with their credentials and hence, the returning customers will have the benefit of just signing in, providing vehicle type and reserving slots according to their convenience and suitable for the car size (choose from the available vacant slots). Also, for walk-in customers the real-time dimension detection of the customer's vehicle, alots suitable slot type to choose from. In this work we will be detecting black listed cars from the amber alerts sent out by law-enforcement agencies and blacklisted car website [2] together with broadcasters, and transportation agencies to activate an urgent bulletin for crime related cases. Our detection algorithm would match all incoming cars with the stored information of the black listed vehicles and immediately send push update to the garage manager through the management user interface such that further action can be initiated. Through the push update technology, the information will be sent to the management even without querying the server for any updates. This feature will provide promptness in reacting to crime related situations. Alongside our license plate recognition system is equipped with the accuracy level of the license plate number detection, which means the image frames from the video are processed multiple times until the algorithm reaches the confidence level of 85% and above to ensure correctness of the information. This accuracy in detection is very important to verify whether the plates have been stolen, the registration status, and any notifications of the vehicle being reported stolen.

Besides using license plate scanning, it is also possible to use the ETC (Electronic Toll Collection) system. Through the special short-range communication between the on-board electronic tag installed on the windshield of the vehicle and the microwave antenna on the ETC entrance of the toll gate, the computer network technology is used to perform background settlement processing with the bank. In this way, the no-stop parking charge would be achieved. More importantly, the customers could also recharge their ETC cards from the bank transfer or by going to the given top-up sites. So that there is no need for the customers to link their bank card or credit card on another website, which can definitely reduce the hazard for the personal information leaked or property stolen.

Another novel idea that has been included in this project is blockchain-assisted payment system. Data privacy protection concerns are driving new regulations around the world. As they act to protect data privacy online and improve personal data protection, our mechanism of data protection would go beyond just complying with all the new requirements and will build trust with consumers and users and stand out from other competitors in the business. Since encryption is central to block chain, a smart blockchain based application will improve data integrity and will have no single point of failure since the data is distributed and decentralized. This brings in a feature where a consumer has the right to instruct a business not to sell personal information to a third party, meaning that a business that tries to sell a blockchain network will have a harder time removing individual blocks from each chain.

The docker based containerization of information is another feature which adds novelty to this work. This method of decentralized information exchange will act as a backbone to support the infrastructure of automated parking garage. In order to maintain automation in parking and eliminating any manual assistance, a system has to be built where data/information access and exchange has to be done without complicated and cumbersome means and with robust support. Some parking companies like Icon Quik Park own 300 garages around New York City, require infrastructure which would enable scalability by providing adequate data management and exchange and also a platform for hosting applications. To comply with this requirement for adequate management of available parking spots and combined pricing across multiple garaging location, containerized information exchange between multiple garaging locations is executed. This method of data exchange is real-time and for better assisting the customers, information is instantaneously available regarding in which garaging location more spots or preferred spots are available as well as pricing in different location and also crime related incident detection. If a customer decides to use different garage locations under the same management throughout the day for their convenience, then combined pricing will be charged once when the customer finally checks out for the day. This provides great flexibility to the customer since there will be only one time charge and no recurrent payment every time the customer checks in to a garage location managed by the same owner. The combined pricing will be charged depending upon the parking rate in a given location at a particular time of the day, hence this is dependent on the dynamic pricing of a garage location. This feature is also supported by docker assisted container migration for information accessibility.

As can be seen from the above discussion that to coordinate availability of parking, pricing and exchange of crime alerts and black listed car information, between different garaging location it is extensively important to host these data/state in a lightweight virtual software entity which can be exchanged between different garaging locations further apart such that all the garaging related information are available at all the same owner administered parking facility. All the parking management applications and its dependencies will also be supported in the container. Containers, also are generally less resource-demanding and faster to instantiate at the destination location that it is offloaded to.

Fully automated parking service theoretically eliminate the need for parking attendants. In addition to this, it is only necessary to register at the first time and then the vehicles' information of the customers will be automatically recorded, which makes the entry and exit of parking more rapid. At the same time, it has also a more reasonable toll system and more convenient search system of parking spots, and can deal with sudden cases like the amber alerts. So, as long as the customer's privacy is guaranteed, this will be a comprehensive upgrade compared to the traditional parking.

2. Glossary of Terms

Blacklisted plates: The plates which are announced by the law enforcement for being involved in any crime.

Buffer time: The extra time to allow the customer to check out from the spot allotted to them.

Confirmed Reservations: Reserved single use parking spots that are paid for ahead of time. Done by visiting the website.

Containerization: Containerization involves bundling an application and data together with all of its related configuration files, libraries and dependencies required for it to run in an efficient way across different computing environments.

Customer: Person who enters the garage with either a parking reservation or looking to make a walk-in reservation.

Database: Hosted on the website, used to store customer data and parking information.

Decentralized: In a decentralised system lower level components operate on local information to accomplish global goals.

ETC (Electronic Toll Collection): It is the system aims to avoid the delay on toll roads, HOV lanes, toll bridges and toll tunnels by collecting tolls without cash and without requiring cars to stop.

Encryption: In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it.

Extension: Customers are allowed to increase the amount of time their car is in the garage, as long as there is available time slots once their time is over.

License plate scanning: It is going to be a scanner at a checkpoint that will record the license plate number of the vehicle and store it into the systems database.

Manager's account: It is the account which the admin of the parking garage has access to. The manager account keeps track of the dynamic pricing, customer details, slots availability. The manager account have access to information related to other parking locations owned by the same company.

No-Show: The act of missing a reservation. The manager will still collect the payment due to paying upon reservation.

Online Reservations: An online reservation system is a web interface you can use for reservation management.

Overstay: An unavailable spot that may impede future reservations due to the customer not returning within the allotted reserved time.

Registration: The customer sign-up with their credentials and these information are available in the company database. When registering before reserving, the customer is asked to input their name, genders, phone number/email, credit card number, address, driver's license number and date of birth. By registering, it allows the customer to reserve parking spots.

Servers: Server is a device that provides functionality for other computer programs or Devices.

Understay: A newly vacant spot due to a customer leaving before the conclusion of their reserved time.

User Interface: The user interface (UI) is the point of human-computer interaction and communication in a device.

Walk-ins: Parking customers who walked into the garage without a reservation.

3. System Requirements

Functional Requirements

Req	**PW**	**Description**
REQ-01	6	The system create accounts for customers.
REG-02	6	The system allow the customer to log in
REQ-03	6	The system allows customers to view and reserve their suitable parking spots.
REQ-04	7	The system allow walk-in customers to make on-spot reservation.
REQ-05	6	The system allows the reservation to be canceled before the agreed time.
REQ-06	8	The system allows the customer to change the reservation time based on their demands
REQ-07	8	The system allows the customer to edit their information
REQ-08	5	The system allows the customer to see the past transactions.
REQ-09	6	The system shows the remaining reservation time to customers.
REQ-10	7	The system allows parking time extension on the website.
REQ-11	5	The system updates the parking slot availability list as soon as the last customer leaves the parking spot.
REQ-12	5	The system update the parking slot availability list as soon as one customer reserves the parking spot.
REQ-13	9	The system provides suitable parking spots based on the size of the customers’ vehicles.

REQ-14	5	The system provides customers with a road map to go to the reserved parking spot.
REQ-15	7	The system scan license plates.
REQ-16	8	The system shall recognize registered customers via plate number.
REQ-17	7	The system shall detect the frequent of customers via plate number and address the type of members based on the frequently use of the customer.
REQ-18	6	The system updates the list of blacklisted car from the website periodically
REQ-19	7	The system detects blacklisted cars from the amber alerts for all incoming cars.
REQ-20	9	The system exchanges the data between multiple garaging locations regarding the customers and blacklisted car
REQ-21	5	The system informs the manager about the blacklist-car warning.
REQ-22	4	The system detects the license plate number with accuracy/confidence level.
REQ-23	7	The system allows combined pricing across multiple garaging locations
REQ-24	6	The system provides dynamic pricing of garaging locations.
REQ-25	7	The system provides a reasonable charging standard on the basis of the peak time of the day.
REQ-26	5	The system allows the customer 5 mins buffer time before being charged for the next time slot.
REQ-27	8	The system allows online payment through user interface.
REQ-28	7	The system allows pre and post payments. (by bank cards or cash)

REQ-29	6	The system shall allow the changes of payment methods.
REQ-30	5	The system shall mail or message customers in case of an emergency.
REQ-31	9	The system protects the personal information of customers.
REQ-32	7	The system will ask if their are another reservation under the same user at the same time in order to make sure and reduce the cost.
REQ-33	6	The user data or information will be input to the database and stored in the main database.
REQ-34	4	The system will automatically charge for the cancellation fees.
REQ-35	8	The system will connect all the garages owned by the same admin and exchange local information at other garaging location
REQ-36	5	The system will notify the manager the available slot in order to update the parking lot spaces.
REQ-37	4	The system will back-up and encrypt customer information for privacy and evade hackers.
REG-38	6	The system will send code to customers confirming the reservation.

Non-Functional Requirements

Req	PW	Description
REQ-39	4	The online reservation requires internet connectivity
REQ-40	7	The customers can check the transaction history.
REQ-41	9	The customers can edit or cancel reservation 2 hours in advance. After 2 hours, the customer is charged a cancellation fee.
REQ-42	6	When registration completes, the customers need to confirm their email
REQ-43	4	Recovery time will not exceed over 5 minutes
REQ-44	8	When the customer complete the payment, the money transaction will be transferred to the owner and based on the contract to receive the fee when applying our system
REQ-45	9	The system will be equipped with exchanging information between different garaging locations
REQ-46	7	The system will be tested in order to check for the potential bugs and then debugged.
REQ-47	9	The system will be simulated in order to check the license plate and check if the customer is registered and the data information will be transmitted to the database with the notification system.
REQ-48	6	The system will be periodically updated without affecting the stored information.
REQ-49	7	Any damage to customers property in the garaging area will be compensated by the garage owner.
REQ-50	8	The system is modeled in a way that it is implementable for any

		automated garage type and the architecture can be altered as per the garage admin requirements.

User-interface requirement

Req	PW	Description
REQ-51	5	In order to create an account, customers need to have a mobile device or state-of-the-art technologies that could use any radio access technologies and establish connectivity for accessing our website and create an account. When registering the account, customers need to input their information such as last name, first name, email, mobile number, date of birth, licence plate number.
REQ-52	8	To log in to the customer's account, the customer needs to put the adequate email and password.
REG-53	5	If the customer forgets the password, the UI will show the reset password features in the login page
REG-54	5	If the customer hasn't registered the account on our page, the UI will show the unregistered features in the login page and lead to the signup page
REQ-55	9	To update and edit the customer's information, the customers need to log into their account and update or edit their information
REQ-56	9	To make an online reservation through web, customers need to log in to their account, choosing the hours which is suitable for them and choosing the slots based on the dimension of the car; lastly,choose their payment method and make a submission. Then the system will receive the data and

		return confirmation code to the customer’s email.
REQ-57	7	To update and edit the customer’s reservation, the customer need to log into their account and update or edit or cancel their reservation (for online customer)
REQ-58	7	To change or cancel reservation, the customers needs to access their account, choose reservation and input the confirmation ID. The user then selects change reservation or cancelation by choosing yes or no option.

4. Functional Requirement Specification

Stakeholders

Stakeholders are the people interested in the success of the organization. They are classified as primary and secondary stakeholders. In our case, the primary stakeholders are the garage owner and managers, who will be directly utilizing the system to increase efficiency and profit of the organization. Another set of stakeholders will be the customers, who will also use our system to have a better and systematic garaging experience.

Actors and Goals

Actors	**Goals**	**Use Cases**
Parking User Interface	To display the available parking slots on the website and allow online payment. On payment confirmation reserve slots depending on customers slot selection in the garage location of customers convenience. Also allow reservation cancellation within a time window.	UC3, UC4, UC5, UC7, UC8, UC18, UC19, UC22, UC23
Parking Interface	To display the empty slots for the walk-in customers and allow them to park when entering the garage	UC3, UC4, UC5, UC6, UC8, UC18, UC19, UC23
System	Update customer's information, reservation and payment method	UC24
System	Update the parking information, the time elapses, time left and available slots if the customer wants to extend the slot reservation time.	UC11, UC12, UC13, UC14, U23
Security System	To protect customer information	UC21
Security	To ensure that customers reserving the slots on the	UC20

	website and the customers arriving for availing parking service are the same.	
Camera	Detect blacklisted car	UC17
Garage Owner	To fix slot price for regular and peak hour.	UC8
Garage Owner	To keep track of the available/unavailable parking-slot, and time before reservation for each slot elapses.	UC11, UC12, UC15,
Garage Owner	To receive push updates on black listed car detection	UC17
Customer	To register and log in on the website	UC1, UC2
Customer	To modify or update their information or payment method on the website	UC24,UC25
Customer	Choosing paying online or on the spot (walk-ins)	UC25
Customer	To check for the available slots and reserve with the help of online payment through the UI.	UC3, UC4, UC5, UC8, UC9, UC10, UC18
Customer	To check for the available slots and and assign the customer a spot with on spot payment.	UC3, UC4, UC5, UC6, UC18, UC19
Customer	Arriving at the garage	UC15
Customer	Exit the garage and pay for the parking hours.	UC9, UC10, UC16
Customer	Receive confirmation code and reservation details through email	UC11, UC12, UC13, UC23
Customer	Cancel the reservation on the website	UC7
Customer	Paying for the parking hours	UC9, UC10

Use Cases

Casual Description

UC1: Sign Up - Customer create an account on our website

UC2: Log In - Customer use their registered credential to access the website

UC3: Slot Availability - Check if slots are available for parking

UC4: Slot Preference - Select from the available parking slots as per choice

UC5: Adaptive Parking - Allotment of slots according to the size of the vehicle

UC6: Walk-In - Customers without advanced reservations

UC7: Reservation Cancelation Or Editing - Cancel reservation before showing up

UC8: Dynamic Pricing - Charging customer depending on the time of the day when the demand if high/low

UC9: Payment - Charge your customers at the end of parking tenure

UC10: Combined Payment - Let customers use multiple parking locations and be charged once finally when he leaves.

UC11: Time elapse - To keep an account of time elapsed since parking.

UC12: Time update - Send customer update about the time left before reservation expiry

UC13: Extra Parking Time - Bonus time alloted to the customer before they are charged for the next time slot

UC14: Reservation Extension - Extend current reservation time from website

UC15: Arrival - Customer entering the garage

UC16: Departure - Customer exit the garage

UC17: Blacklisted Car Alert - Inform garage manager about blacklisted car detection

UC18: Suitable Garage Location - Choose the preferable parking location

UC19: Comparative Price Parking - Park at the location which offers least price

UC20: Parking Security - Send QR-code to the customer when making online reservations or offline, when the customer gets to the desire-slot, to get into the slots, the customer needs to give the code to the system and then the system will allow the customer to get in.

UC21: Information Privacy - Encrypt customer information for secure transaction

UC22: Email Updates - Send pay amount, security code, time elapsed and start of buffer time.

UC23: Reservation- To reserve a slot in 2 hours

Traceability Matrix

REQUIREMENT	**CASE**	**Register Online**	**Log-In Online**	**Make Reservation Online**	**Walk-in**	**Enter**	**Depart**
Create a New Account		X			X		
View or Reserve Parking Spots			X		X		
Change or Cancel Reservations			X				
Update Spot Availability				X	X		
Provide Real-Time Pricing				X	X		
Allow Payment				X	X		
Scan License Plates							X
Recognize Customers							X
Detect Blacklisted Cars							X
Give Internal Directions							X

View or Extend Remaining Time			X				
Send Email		X		X			X
Edit Profile Data			X				
View Profile History			X				
Share Data between Garages				X	X		X

Fully Dressed Description and Sequence Diagram

Use Case UC-17: Blacklisted Car Alert
Related Requirements: REQ-03, REQ-14, REQ-17, REQ-18, REQ-19, REQ-20, REQ-21
Initiating Actor: Garage Owner
Actor's Goal: To being notify if there is a blacklisted car in his/her garage.
Participating Actors: Garage Owner ,Customer, System
Preconditions: The garage owner is able to access blacklisted car data from the law enforcement
Postconditions: The customer is allowed to park either the plate is in the blacklist or not.
Flow of Events for Main Success Scenario: ->1. The garage owner updates the blacklist plate. ->2. Customer accesses the garage throughout the main gate and the camera scans the plate <-3. The system compares the plate with the blacklisted car data a. If the plate is in the blacklist, notify the garage owner. <-4. Allow the customer to access.

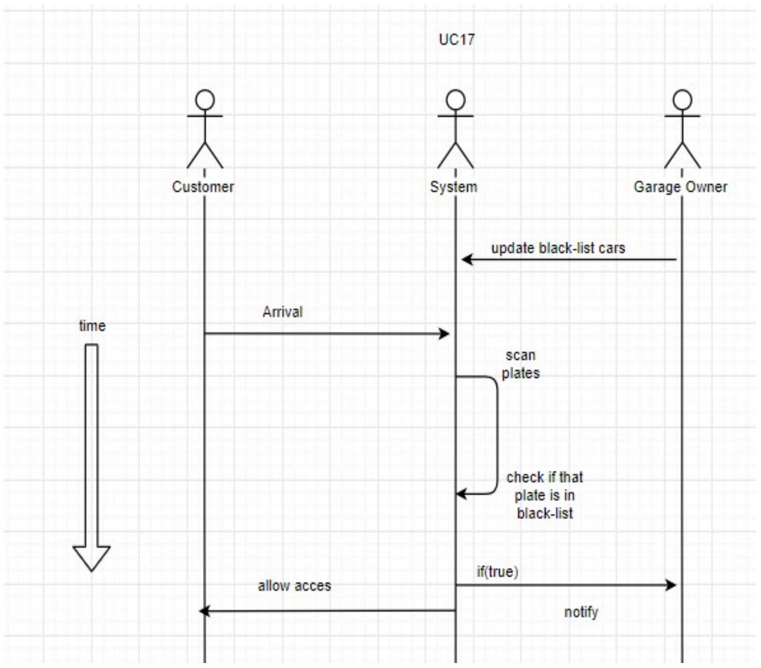


Figure 4-1 Sequence diagram of UC-17

Use Case UC-10: Combined Payment
Related Requirements: REQ-02, REQ-08, REQ-19, REQ-22, REQ-38, REQ-39
Initiating Actor: Customer
Actor's Goal: To have the ability to continue the parking time in other locations in the system
Participating Actors: Customer, System
Preconditions: The customer is required to have an account in the system.
Postconditions: The customers are required to pay more if their parking time is over.
Flow of Events for Main Success Scenario: ->1. The customer makes payment on our website. <-2. The system sends verification email. ->3. The system enters a loop and the exit condition is when time elapses is equal to the paid parking duration . a. If the customer check-in, time elapses starts counting. b. If the customer check-out, time elapses stops counting.

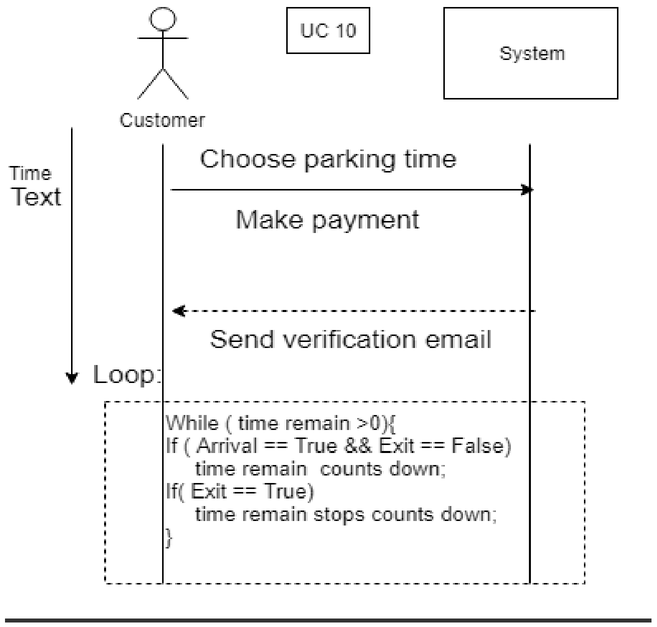


Figure 4-2 Sequence diagram of UC-10

Use Case UC-8: Dynamic Pricing
Related Requirements: REQ-5, REQ-8, REQ-12, REQ-19, REQ-22
Initiating Actor: Customer
Actor's Goal: To charge customers based on the time making reservation on a day.
Participating Actors: Customer, System
Preconditions: The customer is required to have an account in the system and make advanced reservation.
Postconditions: The customers are required to pay more if their parking time is over.
Flow of Events for Main Success Scenario: ->1.Customers log in to the website and make reservations. <-2. System calculates the price based on the time making reservation. ->3. Customers make payment. --4. Customers arrive at the parking lot.

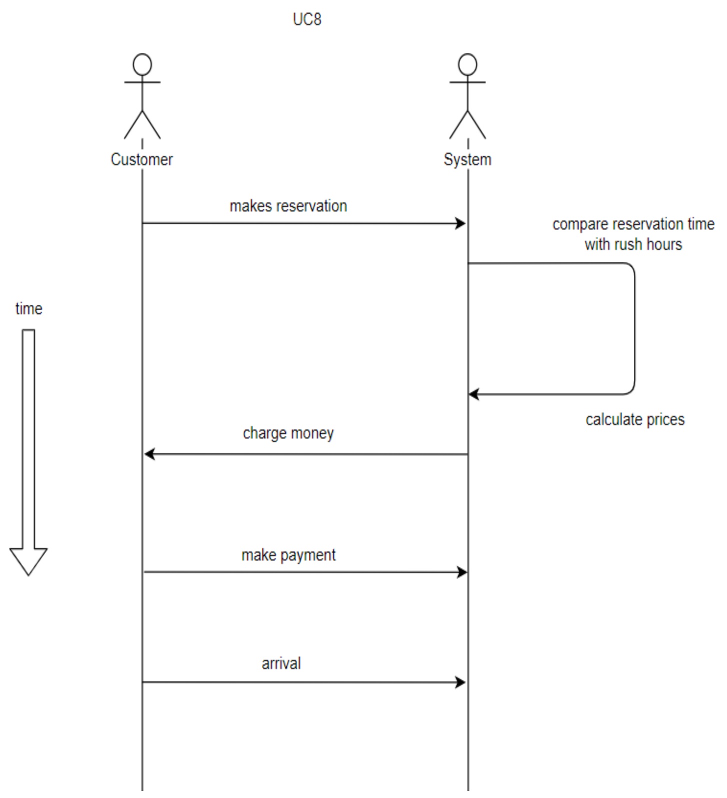


Figure 4-3 Sequence diagram of UC-8

Object Constraint Language:

Classes	Invariant	Precondition	Postcondition
Account	Context account inv:self.useraccess	pre:self.useraccess=true	Post:self.userAccess= self.webaccess
Manager	Context manager inv:self.salary	Context manager :: (s:integer) Pre:self.salary=0	Context manager::(S:integer) Post:self.salary= h*w
Garage	Context garage inv:spots	Context garage::(g:integer) Pre: self.spots=vacantSpots	self.spots =self.spots- self.spotnum

Camera	Context camera inv:
	String: self.licenseplate
	Context camera inv:
	String:self.blacklistlicen
	se
	Context camera inv:
	String:self.shapeofcar
	Context camera inv:
	String:Price
	Context camera inv:
	Integer:Spot
		Context camera:
		(p: string)
		self.licenseplate=0
		Context camera:
		(z: string)
		self.blacklistlicense=onli
		neblakclistcar
		Context camera:
		(m: string)
		self.shapeofcar=0
			If self.licenseplate
			=self.blacklistcar,
			return false
			else if
			shapeofcar(car):
			return Price,Spot

Reservation	Context reservation inv:self.reserve Context reservation inv:self.spotnum	Context reservation::(r:integer) Pre:self.spotnum=true(sp ot is available)	If self.spotnum=true then self.spotnum=self.spo tnum-1(spot is reserved and there is one less spot) else return false
Price	Context price inv : self.hour Context price inv : self.month	Context price :: (h:integer) Pre.selfhour=0 Context price:: (m:integer) Pre.selfmonth=0	Post:total = self.hour + self.month
Notification System	Context notification inv: String: self.email Context notification inv: String self.cost	Context notification: (l: False) Pre.hasreservation= True Context notification:	Post: If self.reservation= True and self.timereservation!=

	Context notification inv: String: self.timestart Context notification inv: String self.timeend Context notification inv: String self.qrcode	(n.False) Pre.selftimereservation=0	expire, sendemail(self.email) return self.cost, self.qr code, self.timestart,self.cost self.timeend

5. Effort Estimation using Use Case Points:

Actor:

UAW: 3(15) + 3(10) + 1(5) = 80.

The complexity of the unadjusted actor weights is determined by the amount of work, and the weights are based on the complexity.

Actor	Description of relevant characteristics	Complexity	Weight
Parking Interface	The website will display the parking interface to provide information about parking for customers.	**Complex**	**3**
Plate Recognition	The scanner interacts through a camera device and software API that allows license plates to be read.	**Average**	**2**
Database	Database stores all customers information, garages and blacklist plate. The database also links to multiple subsystems.	**Complex**	**3**
Customer	Customer can interact with the system via website to view available spot and make reservation.	**Average**	**2**
Shape Recognition	System use camera to detect the shape of the vehicle to charge the customer	**Average**	**2**
Garage Owner	Garage owner can interact with the system via website to adjust parking prices and manage blacklist plate .	**Simple**	**1**

Use case:

UUCP: 6(15) + 9(10) + 7(5) = 215.

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, the higher the complexity. The weights are based off the complexity.

Use case	**Description of relevant characteristics**	**Complexity**	**Weight**
UC1: Sign Up	Simple user interface. 2 Participating actors.	Average	10
UC2: Log In	Simple user interface. 2 Participating actors.	Average	10
UC3: Slot Availability	Graphical interface. 2 Participating actors.	Average	10
UC4: Slot Preference	Graphical interface. 3 Participating actors.	Complex	15
UC5: Adaptive Parking	Simple user interface. 3 Participating actors.	Complex	15
UC6: Walk-In	Simple user interface. 3 Participating actors.	Complex	15
UC7: Reservation Cancelation Or Editting	Simple user interface. 3 Participating actors.	Simple	5
UC8: Dynamic Pricing	Simple user interface. 2 Participating actors.	Simple	5
UC9: Payment	Simple user interface. 2 Participating actors.	Average	10
UC10: Combined Payment	Simple user interface. 3 Participating actors.	Complex	15

UC11: Time elapse	No user interface. 2 Participating actors.	Average	10
UC12: Time update	Simple user interface. 2 Participating actors.	Simple	5
UC13: Extra Parking Time	Simple user interface. 2 Participating actors.	Simple	5
UC14: Reservation Extension	Graphical interface. 2 Participating actors.	Simple	5
UC15: Arrival	No user interface. 2 Participating actors.	Average	10
UC16: Departure	No user interface. 2 Participating actors.	Average	10
UC17: Blacklisted Car Alert	No user interface. 3 Participating actors.	Average	10
UC18: Suitable Garage Location	Simple user interface. 2 Participating actors.	Simple	5
UC19: Comparative Price Parking	Simple user interface. 2 Participating actors.	Simple	5
UC20: Parking Security	Simple user interface. 2 Participating actors.	Average	10
UC21: Information Privacy	No user interface. 2 Participating actors.	Complex	15
UC22: Email Updates	Simple user interface. 2 Participating actors.	Simple	5
UC23: Reservation	Simple user interface. 3 Participating actors.	Complex	15

6. Domain Model Analysis

Domain Model

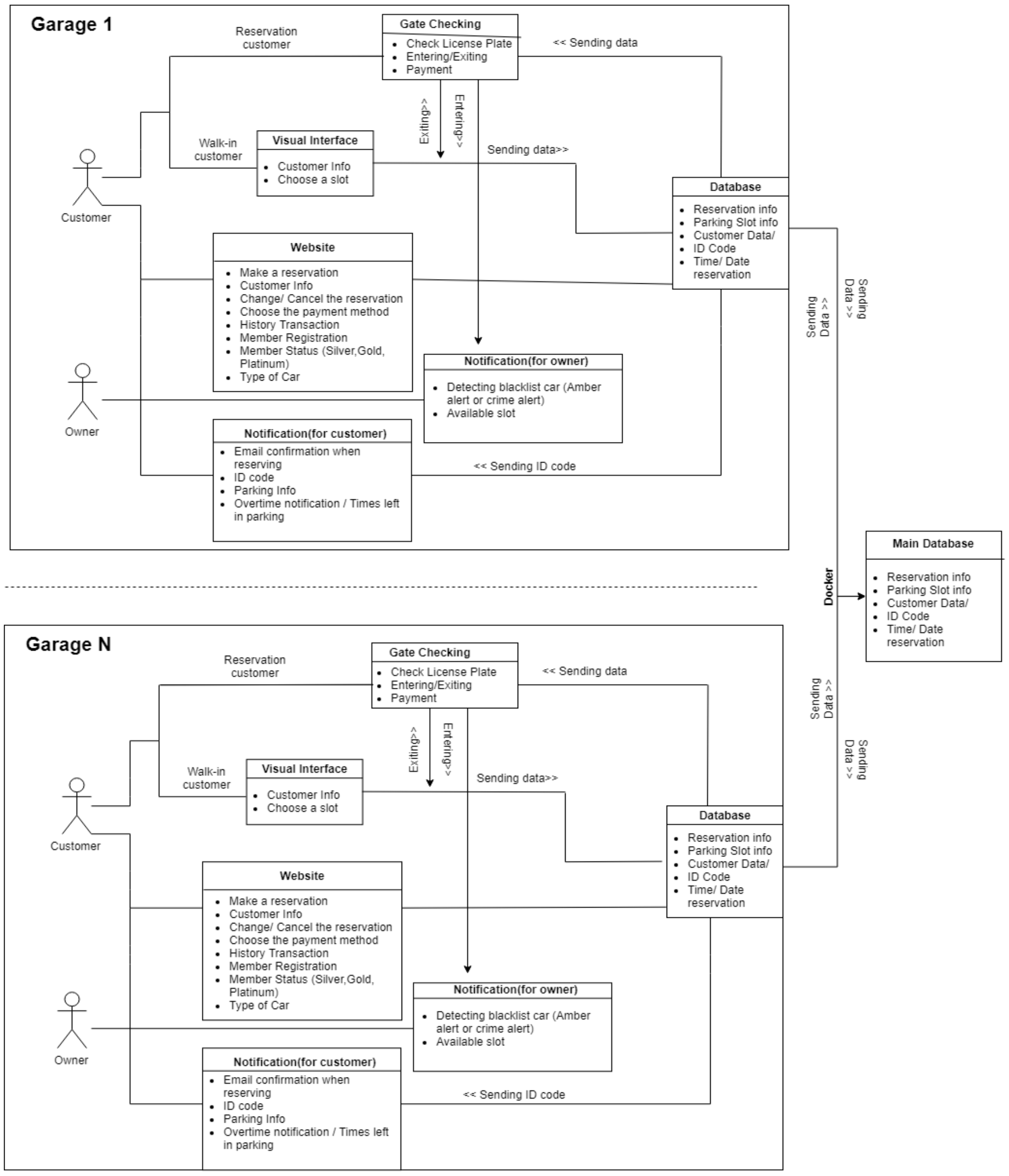


Figure 6-1 Domain model

Domain Model Derivation

The following domain model is derived from the use cases and functional requirements with the highest priorities. The system with each use case allows the customer to register on the website (UC-1) and log in when complete registration (UC-2), then reserving for the desire slot (UC-4); on the other hand, regarding the walk-in customer which have the same concept as the web-register customer but reserving offline (UC-4) . Additionally, after reserving, the customer will park their car with the designed ID code (UC-20), keep track of the remaining time of the parking and pay for their bill or extend their duration time if the parking slot is available for that time. (UC-8, UC-9, UC-11, UC-12, UC-14, UC-18, UC-22,U-23)

There are four concepts that play a main role in the system: the camera system which having the license plate reader function which could define whether that is the customer which had reserved or the crime license plate and having the size detected function ; the visual or user interface that allows the customer to reserve and entering/exiting when reserving is complete and helping the customer to get into the desired slot; the notification system which sends the warning when detecting the crime license plate(for the manager) or sending the duration of time,the ID code,the reservation (for the customer); the pricing based on the user demands or the combined price when parking at multiple places in continuous time.

The four main procedures of the system

- **The camera system:** When the customer enters the garage, at first sight the camera with the license plate reader will read the license plate and compare with the license plate in the database to check if the customer had reserved on the internet or not in terms of web-register customer and apply both when checking for the exit.
- **The visual and user interface:** When registering on the internet or reserving with the visual interface, the customer's information will be stored in the database and retrieving the data when implementing the license plate reader function. Moreover, the customer can register, log in and see the available slot at a given time to make,edit or cancel the reservations ; these actions will be saved in the system or in other words, the database.
- **The notification system:** When the registration is complete, the system will send an ID code, the slot and the hour parking for the customer to park in the right place and when entering the slot, the customer will need to show the ID code to acquire the lot.

Moreover, during the parking time, the customer will receive the email notify the remaining time to make sure that the customer can take the car in time. And for the manager, sending a warning to notify if the blacklist car is entering the parking lot in order to increase the safety for the customer and for the owner. Moreover,to notify the owner if a slot is available when the customer leaves the parking lot.

- **The pricing system:** When completing the registration, a price will be shown in the user interface and the price will depend on if the hour is rush hour or not, the adequate slot, the shape of the slot, the customer can choose to pay there if they use the system once. However, when the customer wants to park in continuous time, the customer can reserve through user interface or visual interface and complete the payment which will be the total price of each parking hour when leaving.

Concept Definitions (D-doing; K-knowing; N-neither)

Responsibility Description	Type	Concept Name
To check if the incoming customer has a reservation.	K	Gate Checking
To check if the incoming vehicle has a black-list plate.	K	Gate Checking
To estimate the size of the incoming vehicles.	D	Size Detecting Camera
To obtain the customer information and make sure the online reservations.	N	Visual/User Interface
To check if the exiting vehicles have finished the payment.	K	Gate Checking
To change or cancel the reservation before the desired time.	D	Website
To choose the payment methods by customers	K	Website
To manage customers and employee information and status.	K	Website
To record the history transaction.	N	Website
To send an email confirmation when reserving.	D	Notification (for customer)

To send an ID code for each parking.	D	Notification (for customer)
To notify customers if the parking is overtime.	D	Notification (for customer)
To show time remaining and real-time price to customers.	N	Notification (for customer)
To notify owners if there is any black-list car detected.	D	Notification (for owner)
To notify the number of available slots.	D	Notification
To obtain reservation information.	K	Database
To obtain parking slots information.	K	Database
To obtain ID code for incoming or reserving customers.	K	Database

Association Definitions

Concept Pair	**Association Description**	**Association Name**
Gate Checking↔Database	When vehicles entering, the gate checking checks the reservation information, black-list related information and the estimated size of the incoming vehicles, then sends the data to the database. When vehicles exiting, the database sends the vehicles' remaining time, the parking charge and the payment method to the gate checking.	Sending data
Gate Checking↔Visual Interface	The gate checking would check if the customers is entering or exiting the parking gate, if entering, the customers need to choose a slot from the visual interface.	Entering & Exiting

Visual Interface↔Database	The visual interface send the information of the customers combined the chosen slots to the database.	Sending data
Website↔Database	The customers could make a reservation or change/cancel the reservation on the website. They can also choose their payment method on the website as well. When the reservation status is changed, the information will be sent to the database and the data there will be updated. Also, the database sends the history transaction data to the website.	Reservation/Payment status changes
Gate Checking↔Notification (for owner)	When doing the gate checking, if a black-list car is checked, then the owner will get a notification from the gate checking. The gate checking will also notify the owner if there is no parking slot left.	Alarm notification
Notification (for customer) ↔Database	After reserving, the customers will receive an email confirmation and an ID code from the database. In the meanwhile, if the customer is parking overtime, the database will generate an overtime notification to remind the customer.	Sending ID code & overtime alert
Database↔Main Database	All the data in the every-garage database has an interaction with the main database.	Sending data

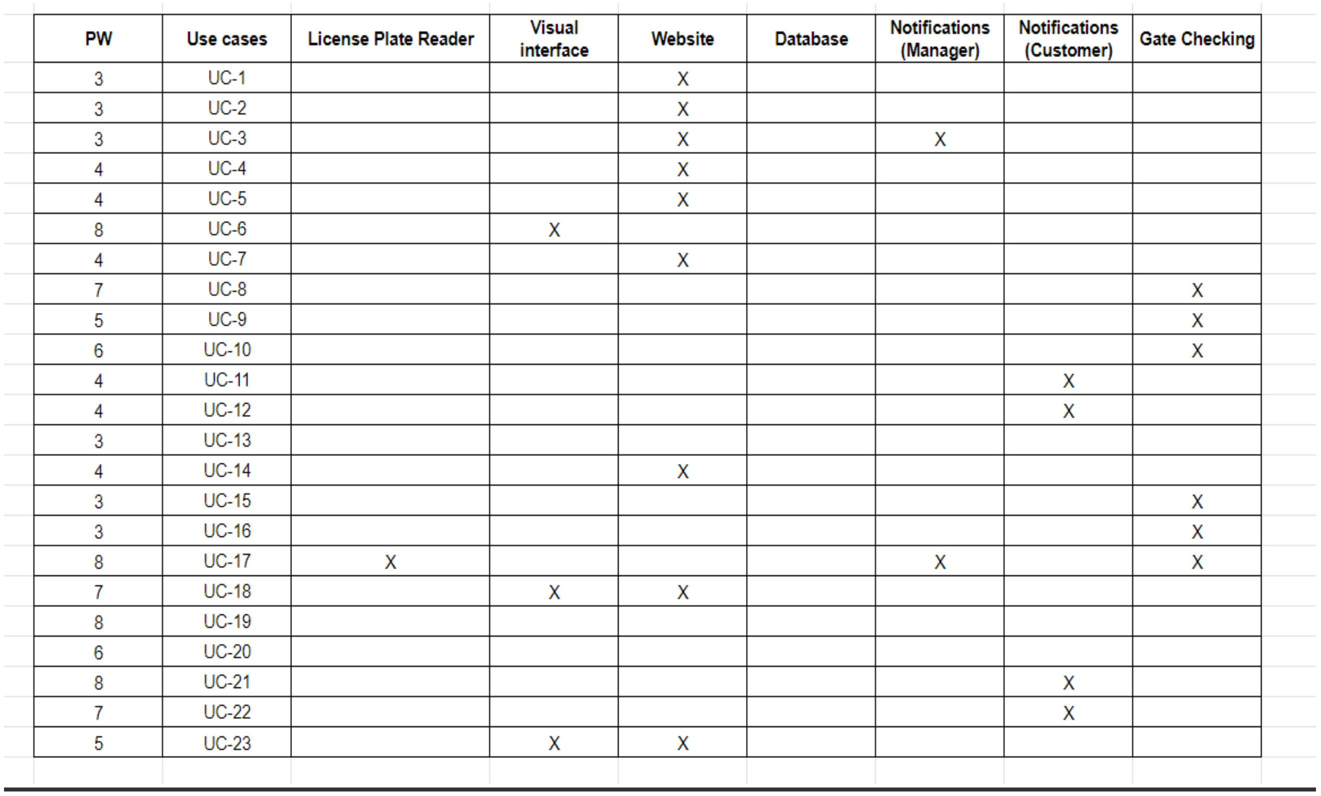
Attribute Definitions

Concept	Attribute	Attribute Description
Gate Checking	Check License Plate	Check if the incoming vehicle is reserved and black-listed.
	Entering/Exiting	Check if the vehicle is entering or exiting the gate.
	Payment	Check the parking charge to be paid and also the payment method.
Visual Interface	Customer Info	Gather the customers' information.
	Choose a slot	Choose an available slot for the customer on the basis of the size of the vehicle.
Website	Make a reservation	Make a reservation on the website.
	Customer Info	Gather the customers' information.
	Change/Cancel	Change/Cancel the reservation.
	Payment Method	Choose/Change the payment method.
	History Transaction	The history of transaction could be checked on the website.
	Member Registration	The customer could register a member card on the website for the parking discount,etc.
	Member Status	The members can check their member status on the website and check the members' privilege.

	Type of car	The information of the type of the car can be uploaded on the website for a better reservation.
Notification (for owners)	Detecting black-lisk car	The owner will get a notification if there is a black-list car detected. (from amber alert or crime alert,etc.)
	Available slot	The owner will receive a notification if there are a few parking slots left. The owner can also change their charging standard based on the number of available slots.
Notification (for customers)	Email confirmation	The customers will get a confirmation email after the reservation to make sure the security.
	ID code	Every vehicle has a unique ID code for corresponding its own parking information.
	Parking Info	The parking information like the place of the selected slot, the parking charge, etc.
	Overtime/Time left	The customers will receive a notification if the parking time is over the desired time. They may also be notified if there is, for example, five minutes left for their parking.
(Main) Database	Reservation Info	Store the information that if the customers have already a reservation and the desired parking time and parking garage.
	Parking slot Info	Store the information that if a specific slot is used or not. Also store the information of

		how many slots are being used at that time.
	Customer Data	Store the customers’ information.
	ID Code	Every vehicle has a unique ID code for corresponding its own parking information. Store and generate the ID code.
	Time/Date reservation	Store the exact time and date of the reservation for customers.

Traceability Matrix



PW	Use cases	License Plate Reader	Visual interface	Website	Database	Notifications (Manager)	Notifications (Customer)	Gate Checking
3	UC-1			X				
3	UC-2			X				
3	UC-3			X		X		
4	UC-4			X				
4	UC-5			X				
8	UC-6		X					
4	UC-7			X				
7	UC-8							X
5	UC-9							X
6	UC-10							X
4	UC-11						X	
4	UC-12						X	
3	UC-13							
4	UC-14			X				
3	UC-15							X
3	UC-16							X
8	UC-17	X				X		X
7	UC-18		X	X				
8	UC-19							
6	UC-20							
8	UC-21						X	
7	UC-22						X	
5	UC-23		X	X				

Mathematical Modeling

An average transfer of data between any two actors has a man-in-the-middle risk, where a third bad actor can tap the message midway and see what is inside. To make such attempts at theft meaningless, we will use asymmetric encryption to secure various forms of transactions between us and our customers; this will protect key customer information by making any records unreadable and virtually uncrackable.

We will accomplish this by using the following mathematical technique:

1. We will give the customer a *public key,* which is the left side of the equivalency:
   $m^{E}\ mod\ N^{\blacksquare} \equiv c^{\blacksquare}$
2. The customer will replace "m" with their intended message using a padding scheme and send us back the encrypted message "c".
3. We will decrypt the message using our *private key* and use the content of the message. The message could be the license plate number of the customer's car, for example.
4. Following the example in step 3, we would then record this transaction in our entry/exit ledger (in its encrypted form).
5. In the event of the theft of this ledger, the customer's privacy would be safe and their travel history will not be exposed. This is due to the fact that the bad actor does not have the *private key* required to make actual meaning of all the data in the ledger.
6. Any bad actor who decides to brute-force decrypt the data would be stuck solving the puzzle for decades or longer.

The determination of the perfect private key is where the largest amount of mathematical modeling is required. We have to make sure that even if the public key is known, the private key cannot be cracked using it. We also have to make sure that the private key cannot be easily cracked. In order to achieve these objectives, we would have to pick the right values for "N", "E" and a third letter "D", which will be used to undo the effects of "E".

"E" can be any small number greater than 1 of our choosing, with the only condition being that it cannot share any factors with $\phi(N)$, which is something we'll explore soon.

To find "N", we will pick any two really large, preferably 100 digits or more, prime numbers and multiply them together:

$$N = P_1 * P_2$$

To understand why this is the best idea, we will first have to define something called the breakability of a number "X". The breakability of a number is the amount of numbers ≤X, but greater than 1, that do not share any common factor with X.

The best "X" to pick therefore is a prime number. This is because a prime number cannot be broken up more than twice, once with a factor of 1 and another with a factor of itself. This property makes it so that there will be no ambiguities in future calculations.

Here is the formal definition, when X is a prime number:

$$\phi(X) \ = X - 1$$

The following property is also valid for the multiplication of two prime numbers :

$$\phi(A * B) = \phi(A) * \phi(B)$$

We can then port this information with "N" this way:

$$\phi(A * B) = \phi(A) * \phi(B)$$

$$\phi(P_1 * P_2) = \phi(P_1) * \phi(P_2)$$

$$\phi(N) \ = (P_1 - 1) * (P_2 - 1)$$

This part is extremely difficult to do in the other direction and is the reason, why we would want to keep it $\phi(N)$secret.

The number "D" is another number we would want to keep secret. Why? Because as we will see, this number is really the one that will unlock the encrypted message easily and in one calculation. This is our *private key code*.

To begin to find "D" we must define Euler's Theorem. This theorem utilizes $\phi(N)$, the breakability of N, that we found from a previous part in its definition.

Here is how the Theorem is defined, using our variables:

$$m^{\phi(N)} \equiv 1 \ mod \ N$$

Using this and some properties of modulus mathematics, we can logically deduce the following:

$m^{k*\phi(N)} \equiv 1 \ mod \ N$; because $1^{k^{\blacksquare}}$always equals 1, not matter the k.

$m * m^{k*\phi(N)} \equiv m \ mod \ N$; because any number times 1 equals itself.

$m^{k*\phi(N)+1} \equiv m \ mod \ N$; using basic rules of exponents

We can use this definition on the original public key: $m^E \ mod \ N^{\blacksquare} \equiv c^{\blacksquare}$, and find the following, assuming that we're looking for a "D" that depends on $\phi(N)$:

$$m^{D*E} = m \; mod \; N$$

Then, $D = (k * \phi(N) + 1)/E$

All this means is that our secret $\phi(N)$factorization has been utilized in a way that incorporates the non-secret parts E and N. We can now use D to transform the encrypted “c” message to the decrypted message “m”.

In conclusion, by incorporating all the steps, we will have an extremely strong encryption system. This will put the customer’s mind at ease, knowing that even if ledger data is leaked, their information is safe.

7. Interaction Diagram

Online Registration

Use Case UC-1: Sign Up
Related Requirements: REQ-01, REQ-06,REQ-15, REQ-17, REQ-38, REQ-51
Initiating Actor: User
Actor's Goal: To enter their information to create a new account.
Participating Actors: Website and Database.
Preconditions: User has a car, driver's license, email address and a payment method and has a connection to the Internet.
Postconditions: Account is created successfully.
Flowchart of Registering: 1. If the user decides to Register: a) They will click "Sign Up" b) They will be Redirected to the Sign-Up page. c) They will enter their name, license number, plate number, payment method, and email address. d) The database will store their information. e) They will redirected briefly to the confirmation page. f) They will be redirected to the login page. They can now Log-In.

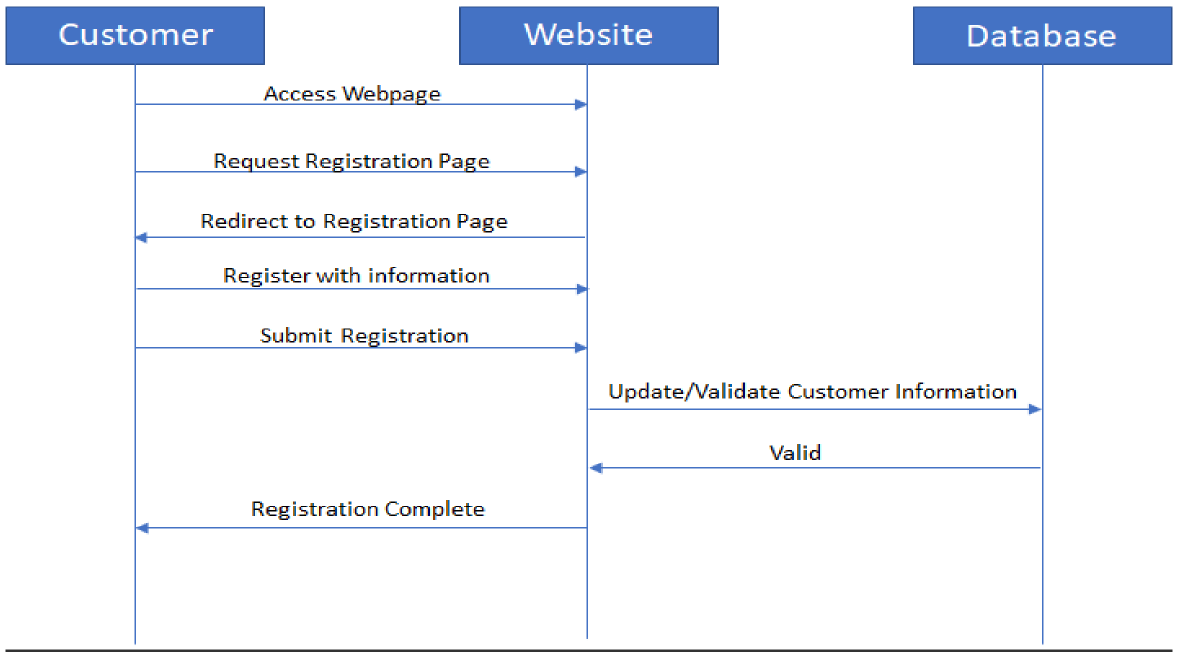

Customer
Website
Database
Access Webpage
Request Registration Page
Redirect to Registration Page
Register with information
Submit Registration
Update/Validate Customer Information
Valid
Registration Complete

Figure 7-1 Interaction diagram of online registration

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 customers 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.

<u>Login</u>

Use Case UC-2: Log In
Related Requirements: REQ-02, REQ-06,REQ-07, REQ-09
Initiating Actor: User
Actor's Goal: To gain access to their account on website
Participating Actors: Website, Database

Preconditions: User already has an account on our database

Postconditions: User login successfully

Flowchart of login:

1. If user decide to login

   a) They try to input their username and password
   b) Click login
   c) Website will send the data to verify
   d) The database send back the result
   e) If the user id and password match in the system, website redirect user to homepage with their account login

Else repeat step a)

2. If user click forget password

   a) Redirect to forget webpage to input email
   b) Send request to database
   c) Database send new password to user’s email
   d) Then user starts login process

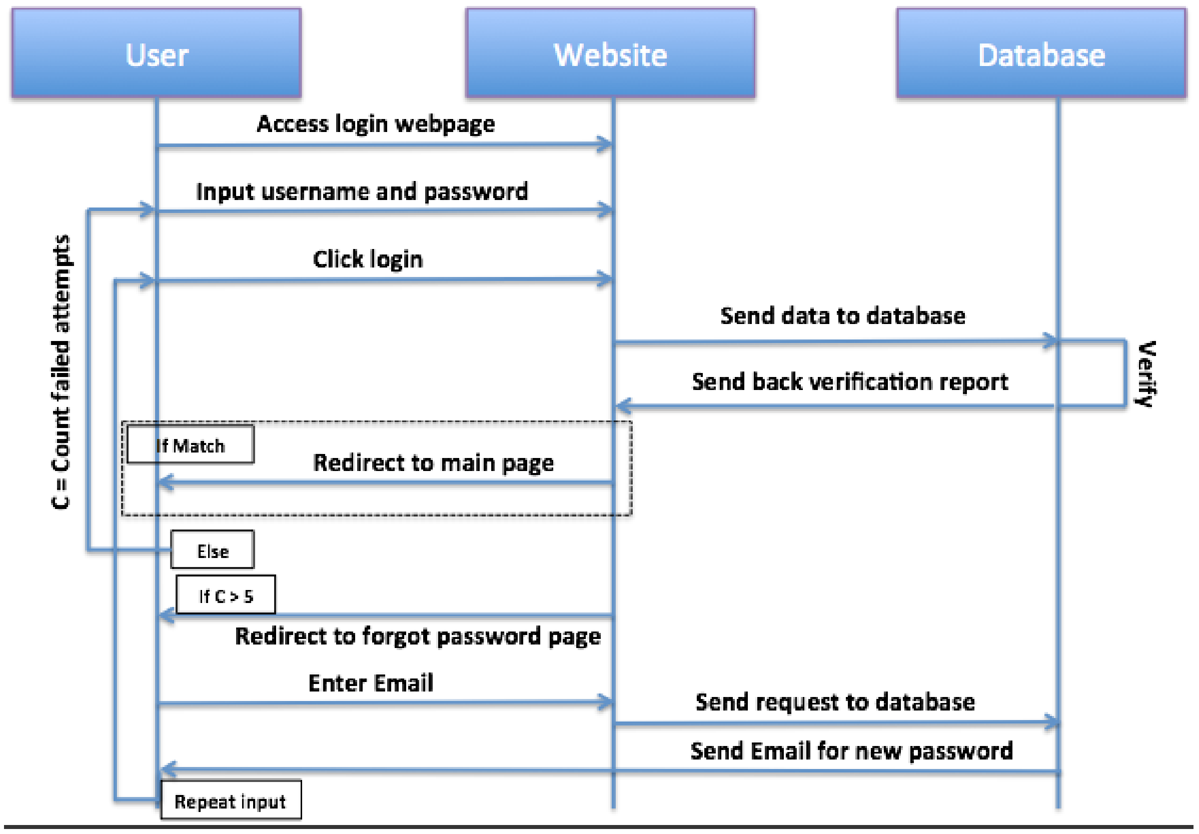


Figure 7-2 Interaction diagram of log-in

The users access the RU parking website inputs the credentials: user id and password. They then click the login tab and wait to be authenticated by the database. After this verification if the response if positive the user is allowed to login and redirected to the home page, however if the authentication verification is negative then the user repeats the input of userid and password for maximum 5 times. When the trials exceeds 5 attempts the user is redirected to forget password page and prompted to enter the registered email where the password change link is sent.

Arrival

Use Case: 1. **UC6:** Walk-In - Customers without advanced reservations 2. **UC15:** Arrival - Customer entering the garage
Related Requirements: REQ-01, REQ-02, REQ-03,REQ-11,REQ-12, REQ-13, REQ-

14,REQ-15,REQ-18, REQ-21, REQ-23,REQ-40, REQ-45

Initiating Actor: Garage

Actor's Goal: Allow customers to enter the garage.

Participating Actors: User, Detect System(Camera), Visual Interface, Database

Preconditions: User has made a reservation on the website or just a walk-in customer.

Postconditions: User enters the garage and finds a desired parking slot.

Flowchart of Arrival:

If user decides to enter the parking lot, the camera will check the license plate

1. If the license plate does not belong to the black list.
    1) If this license plate has already made an online reservation
        a) The user should make a confirmation from the visual interface.
        b) The car can enter the garage.
    2) If this license plate has not made an online reservation
        a) The user will make a new reservation with his/her information in the visual interface.
        b) The visual interface will send an ID code to the user.
        c) The new reservation data will be sent to the database.
        d) The car can enter the garage.
2. If the license plate belongs to the black list.
    1) The detect system will inform the manager with a warning alert.
    2) The car with this license plate can not enter the garage.

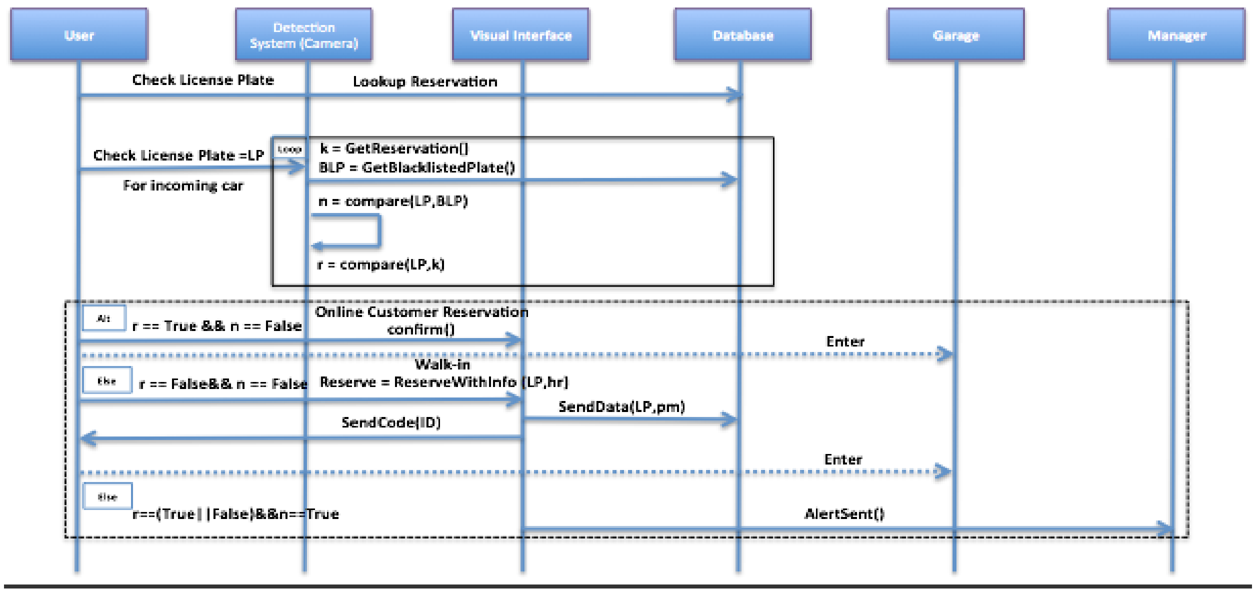


Figure 7-3 Interaction diagram of arrival

The user comes in and the License plate is detected by the camera and the corresponding reservation is looked up. A loop is run continuously where in a check is done whether the license plate detected already has a reservation and if it is blacklisted license plate or not. For a given license plate if a reservation is found and the car is not blacklisted then the customer reservation is confirmed. In case when no reservation is found and the license plate is not blacklisted then the customer falls under the category of walk-in clients. His reservation is done on spot by quick signup method. While when license plate matches the blacklisted cars list then an alert is sent to the manager irrespective of whether a corresponding reservation is found or not. Similarly to the above use cases, the subscriber is the customer and The publisher(garage control) will lookup the corresponding reservation in DB and authenticate the subscriber to park the car in the already assigned slot.

Reservation

Use Case:

1. **UC3:** Slot Availability - Check if slots are available for parking
2. **UC4:** Slot Preference - Select from the available parking slots as per choice
3. **UC5:** Adaptive Parking - Allotment of slots according to the size of the vehicle
4. **UC14:** Reservation Extension - Extend current reservation time from website
5. **UC23:** Reservation- To reserve a slot in 2 hours

Related Requirements: REQ-02, REQ-05, REQ-09, REQ-11, REQ-12, REQ-13, REQ-26, REQ-28, REQ-40, REQ-54
Initiating Actor: User
Actor’s Goal: User will successfully reserve for the parking slot
Participating Actors: Website, Database
Preconditions: User already has an account on our database and login successfully
Postconditions: Reservation is saved in database and the reserved slot will be marked as not available. The system will send confirmation email and charge money.
Flowchart of reservation: 1. If user decides to make a new reservation: a) They try to log in and access to the website b) They will enter the type of vehicle (size of vehicle) c) Database will match the type of vehicle and return the map related to it to the website d) The website will display the map with all slots including available and unavailable e) They will check available time f) Website will display time after grabbing data from database g) User will choose the time h) Website will send the data to database for reservation i) Then database will be updated j) Database return confirmation and prices to website for displaying k) If user want to change time => repeat step 1e l) Else user will press Confirm and the reservation will be saved in database m) Then the email confirmation will be sent to the user 2. Else => User just wants to extend the time: a) They try to log in and access to the website

b) They will check available time
c) Website will display time after grabbing data from database
d) User will choose the time
e) Website will send the data to database for reservation
f) Then database will be updated
g) Database return confirmation and prices to website for displaying
h) If user want to change time => repeat step 2a
i) Else user will press Confirm and the reservation will be saved in database
j) Then the email confirmation will be sent to the user

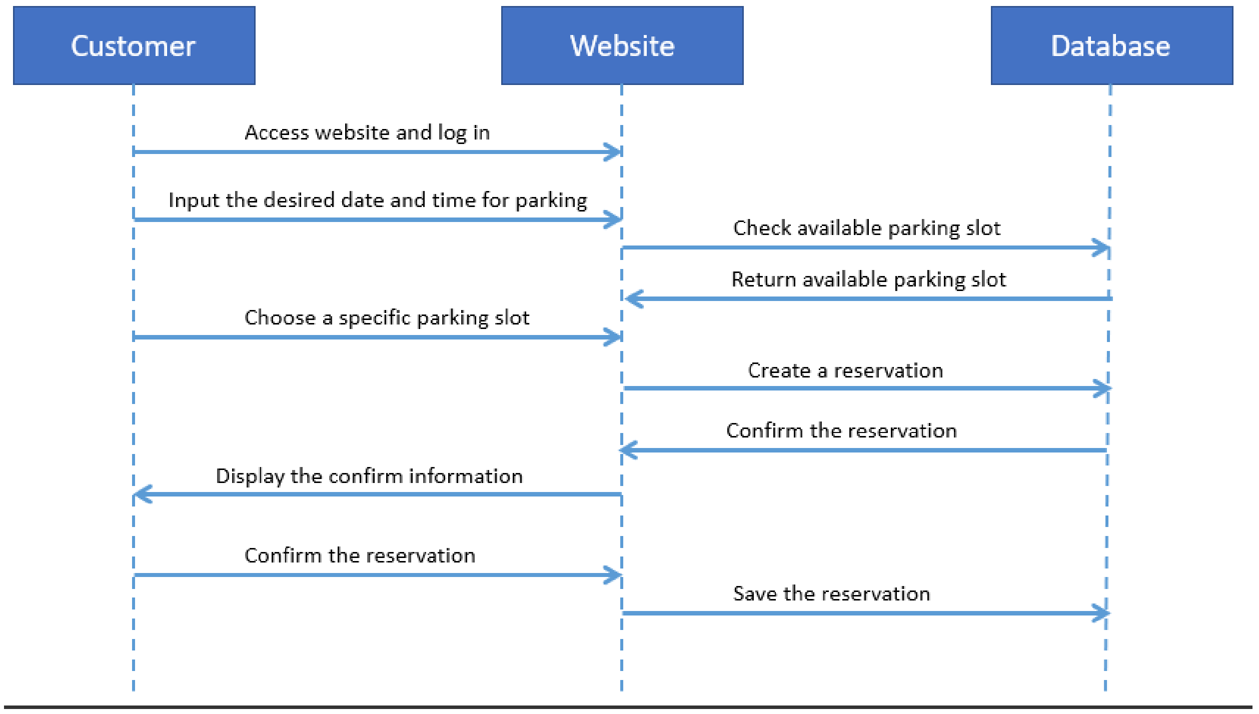

Customer
Website
Database
Access website and log in
Input the desired date and time for parking
Check available parking slot
Return available parking slot
Choose a specific parking slot
Create a reservation
Confirm the reservation
Display the confirm information
Confirm the reservation
Save the reservation

Figure 7-4 Interaction diagram of reservation

When the customer plan to make a (online) reservation, he/she needs to first go to the website and fill in the desired date and time of the parking. Then the database will get the request from the website and query all the available parking slots which meet the demand of the customers. When it finished, the customers will get the information about the available parking from the website and make a choice for the specific parking slot. The database will get a request of creating the confirmation and sent a corresponding confirmation instruction back. After that, the customers will receive a confirmation (e-mail) from the website. If the customers make the confirmation, this reserving information will be automatically saved in the database. Comparing with the last version (report 2), some parts of this interaction diagram are deleted for the reason that they are redundant at the current situation or they are already achieved in the formal step (like the registration step). For this use case, the Publisher-Subscriber design pattern was applied to improve the use case's design. The subscriber is the customers while the publisher is the website. So that in this use case, the publisher gives the subscriber the information of the available parking slots or notify the customers there is no free slots for the desired parking time of the customers.

Departure

Use Case UC-16: Departure
Related Requirements: REQ-03,REQ8, REQ-14, REQ-17, REQ-18, REQ-19, REQ-21, REQ-22,
Initiating Actor: Garage
Actor's Goal: Allow customers to exit the garage
Participating Actors: Customers, Camera, Database
Preconditions: Customer's vehicle is in the parking lot
Postconditions: Credit card information to make payment

Flow of Events for Main Success Scenario:

1) if customers choose single parking on website, they do not need to pay when exit because they have been charged when they made reservation on website.

a)Garage opens gate

b)Customers exit

c)Camera scan exiting vehicle's plate and stores it in database, as long as leaving time.

2)If customers choose combined payment on website

a)Garage opens gate

b)Customers exit

c) Camera scan exiting vehicle's plate and stores it in database, as long as leaving time.

d) System will base on arriving and leaving time of that vehicle to calculate price.

e)Customers do payments

After leaving garage, customers will get confirmation email.

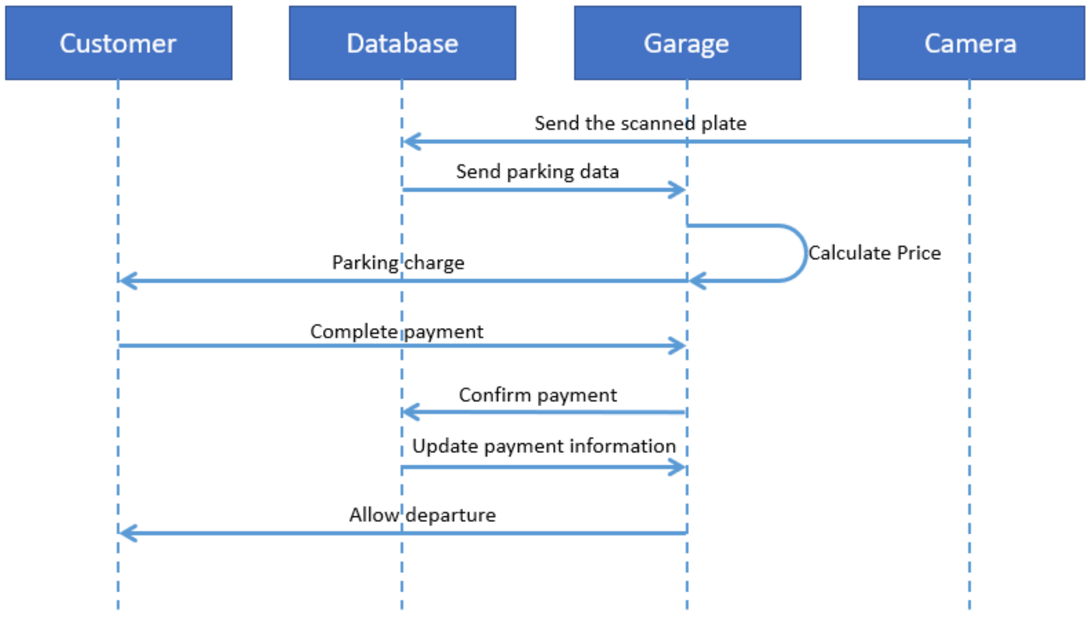
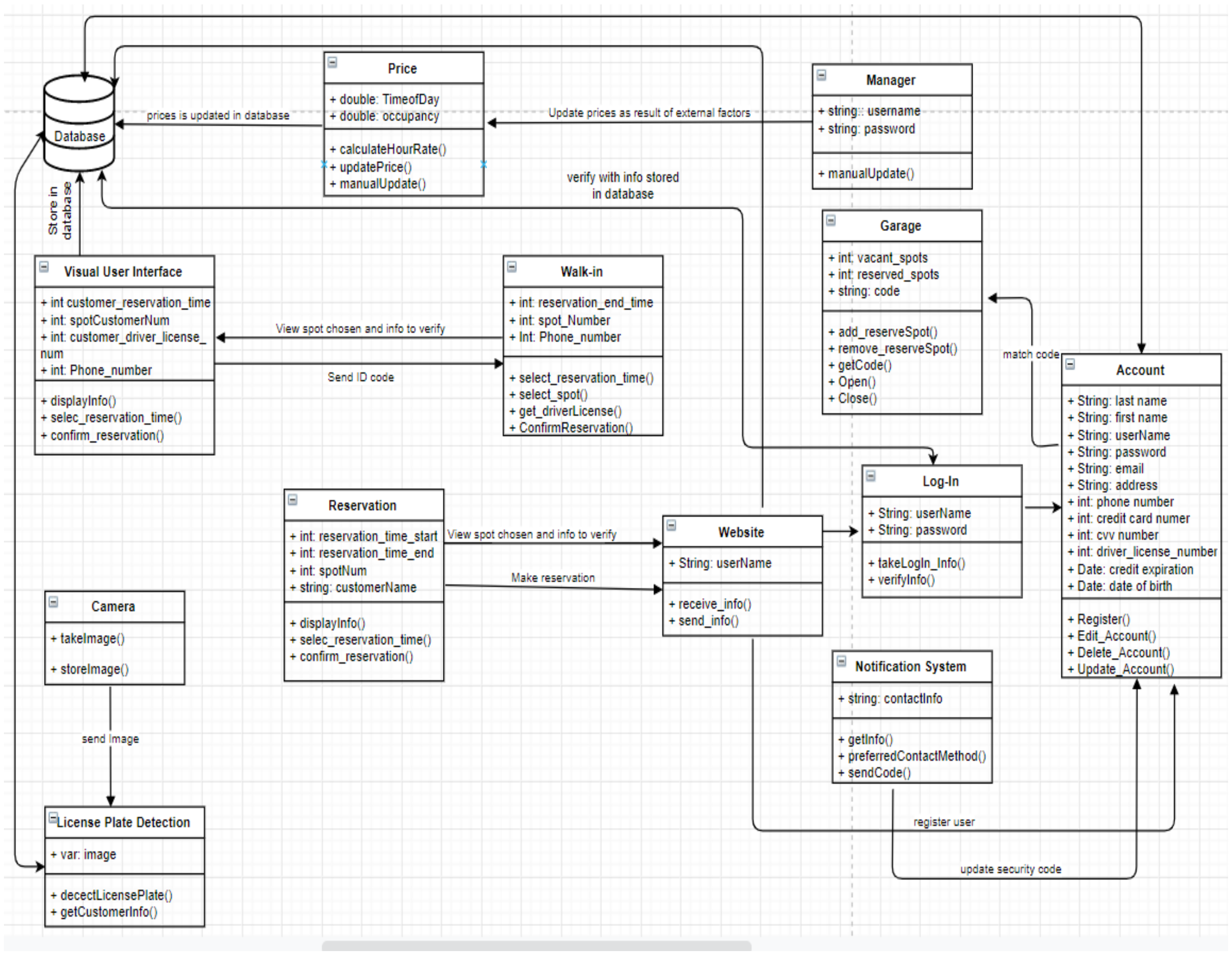


Figure 7-5 Interaction diagram of departure

When the customers want to exit the parking garage, the camera of the gate will scan the license plate of the vehicles and query this vehicle's parking information in the database. While the database found the corresponding information (including using the single or combined parking garage), it will be sent to the garage to make the price calculated. Then the garage will charge the payment for the customers. When the customers complete the payment, the database will receive a confirmation from the garage and then update the payment status. After that, the customers will be allowed to exit. Comparing with the old version (report 2), this interaction diagram made some simplifications. The single-garage case and combined-garage case were made in one diagram here for the reason that this step will be finished in the "send parking data" step. For this use case, the Publisher-Subscriber design pattern was applied to improve the use case's design. The subscriber is the customers while the publisher is the garage. So that in this use case, the publisher will present the subscriber with the total parking charge and the subscribers will pay the noted price before exit the garages.

8. Class Diagram and Interface Specification:

Class Diagram:



Data Types and Operation Signatures:

1. **Walk-in:** the purpose of this class is for walk-in customer register immediately
   - Attributes:
     - int reservation_time: indicate reservation time walk-in customers want to park in the garage

    - int spotNumber: indicate the spot number walk-in customers park in the garage
    - String LastName: last name of walk-in customers
    - String FirstName: first name of walk-in customers
    - int driver_license_num: driver's license number of walk-in customers
- Operations:
    - select_reservation_time(): walk-in will choose reservation time
    - select_spot(): walk-in will choose the spot for parking
    - get_driverLicense(): walk-in display customer information to the monitor
    - confirmReservation(): walk-in will confirm reservation information
    - Receive_code(): receive the code to enter the lot of the garage

2. **Visual User Interface:**
- Attributes:
    - Int customer_reservation_time: walk-in customer reservation time
    - Int spot_customer: walk-in customer choosing lot
    - Int Phone_number: the walk-in customer phone number
- Operations:
    - Display_info():display the info of the customer after reserving to confirm
    - Customer_reservation():display the complete screen after reserving

3. **Account:**
- Attributes:
    - String lastname: last name of the customer
    - String firstname: first name of the customer
    - String username: the username of the customer
    - String password: the password of the customer
    - String email: email of the customer
    - String address: address of the customer
    - Int phone_number: the phone number of the customer
    - Int credit_card_number: the credit card number of the customer
    - Int cvv: cvv number of the credit card
    - Int drive_license_number: license plate of the customer

    - Date credit_expiration: the date expiration of the credit card
    - Date Date_of_Birth: date of birth of the customer
- Operations:
    - Register(): User can register to save their information to the database
    - Edit_account(): User can edit the information of them to the account
    - Delete_account(): User can delete the account
    - Update_account(): Update the code to enter the lot of the garage

4. **Website:** this class is used for customer to sign in, reservation, register
- Attributes:
    - String Username: after logging in ,the customer will show the username.
- Operations:
    - Receive_Info(): receive information of users from reservation
    - Send_Info(): send information to database to store.

5. **Log-In:** this class is used for logging in users
- Attributes:
    - String userName: user name of customers when they register
    - String password: password of customers when they register
- Operations:
    - takeLogin_info(): get username and password of customers.
    - verifyInfo(): verify username and password of customers with database.

6. **Reservation:** this class is used for both online-customer and walk-in customer to make a reservation
- Attributes:
    - int reservation_time_start: the start time to make a reservation
    - int reservation_time_end: the time to end a reservation
    - int spotNum: choose the spot number and reserve
    - string customer_name:
    - String code: getting code after reservation
    - Fast track check-in for walkin customers

- Operations:
    - Display_info(): display the info of the customer after reserving to confirm
    - Select_reservation_time(): The reservation is selected by the customer
    - Confirm_reservation(): display the complete screen after reserving
    - Receive_code: send the ID code to the customer in order to get in the right reservation lot

7. **Camera:** this class will be used for camera in order to take and store image of plates

- Operations:
    - takeImage(): take plate images from car of customers
    - storeImage(): store plate images of customers

8. **License Plate Detection:** this class will be used for detecting plate numbers

- Attributes:
    - var image: this is an image of plate numbers taken from cars of customers
- Operations:
    - detectLicensePlate(): detect License Plate from cars of customers
    - getCustomerInfo(): after detect the Plate it will verify with database and get customer information belongs to that plate

9. **Garage:** this class will show what spots have been reserved and what spots have not reserved yet.

- Attributes:
    - int vacant_spots: numbers of spot have not been reserved
    - int reserved_spots: numbers of spot have been reserved
    - String code: the ID code to enter the lot
- Operations:
    - add_reservedSpot(): add spots that have been reserved by customers
    - remove_reservedSpot(): remove spots when customers leave or cancel reservation.
    - getCode(): the lot of the garage will need the customer Code to let them enter the lot.

- Opening: open when customer enters
- Close: close when customer exits

10. **Notification System:** this class will notify customers by sending code to them.
   - Attributes:
      - String contactInfo: this variable is email of customers
   - Operations:
      - getInfo(): get information of customers from database
      - preferredContactMethod(): use email for sending information and code for parking slots
      - sendCode(): send code to email of users to notify them.

11. **Manager:** this class is for the manager to update the price
   - Attributes:
      - String username: the username of the manager
      - String password: the password of the manager
   - Operations:
      - Manual_update: the manager can update the price based on the average market price

12. **Price:**
   - Attributes:
      - Double TimeofDay: time of the day to determine whether it is rush hour, morning,night,...
      - Double Occupancy: the remaining spot in the manager garage
   - Operations:
      - calculatehourate(): by multiplying the hour with the price per hour to calculate
      - updatePrice(): update the price after the manager is sending the updated price
      - manualUpdate():updating the parking hour and price after each hour passes

Traceability matrix

All the Domain Model in this part is taken from section 4.1 in our report 1. The functionality is based on the diagram above.

REQUIREMENT	CASE	Register Online	Log-In Online	Make Reservation Online	Walk-in	Enter	Depart
Create a New Account		X			X		
View or Reserve Parking Spots			X		X		
Change or Cancel Reservations			X				
Update Spot Availability				X	X		
Provide Real-Time Pricing				X	X		
Allow Payment				X	X		
Scan License Plates							X
Recognize Customers							X
Detect Blacklisted Cars							X
Give Internal Directions							X
View or Extend Remaining Time			X				

Send Email		X		X			X
Edit Profile Data			X				
View Profile History			X				
Share Data between Garages				X	X		X

1. **Database:**
   - Responsibility: this is treated as a single database for each garage(because one of our goals for this project is to help users parking at multiple locations). It stores:
     - Reservation information
     - Empty and reserved parking lot
     - Customer's Data
     - Time and date of the reservation
   - Classes:
     - store()
     - write()
     - update()
2. **Main database:**
   - Responsibility: this main database works as a combination of all single databases above. Also, to keep track of customers who park at multiple locations.
3. **Website:**
   - Responsibility:
     - sign_up
     - sign_in
     - make reservation
     - cancel reservation
     - change reservation
     - view history transaction/parking time
     - declare type of car of the customer

- Classes:
    - send_infor()
    - retrieve_infor()

4. Gate checking:

- Responsibility:
    - check License plate, compare with registered plate
    - Allow entering/exiting prior to completed payment
- Classes:
    - detectLicencePlate()
    - open()
    - close()

5. Notification for customer

- Responsibility:
    - Send confirmation email about: reservation/payment/entering status/ exiting status
    - Send alerts about overstay status
    - Send ID code for confirmation
- Classes:
    - sendInformation()
    - sendCode()

6. Notification for owner:

- Responsibility: to help the garage owner to stay awake when blacklist cars are recognized.
- Classes:
    - retrieveDataFromMainDB()
    - sendInfoToOwner()

Design Patterns:

Design patterns are put into use to help us adapt to any unexpected software changes. The presence of design patterns keeps us flexible and helps us change things systematically in order to not break the entire project due to any minor change.

A decorator design pattern is used in our project. Our entire website design uses the decorator design pattern. This is proven by the fact that even if we get rid of any graphics, fancy buttons, colors, backgrounds, fancy text boxes, scroll features, or any other fancy “eye candy”, the underlying HTML code still ensures the entire website functions no matter what. Even if any of the graphical elements fails or faces some sort of change, the decorator design pattern ensures that all the main features, such as logging in, making a reservation, entering info, all still work. Due to this design pattern, continuity of service remains unaffected.

Another major design pattern that we have used is the proxy design pattern. Proxies are especially useful in our project due to the fact that a lot of things have to be simulated. Proxies ensure that any code written using a simulated device, for example, doesn’t affect the rest of the code when the actual device is reintroduced to the project. Specifically, when reading license plates to let customers in or to check for blacklisted cars, we use a simulated camera, something that just scans the pixels of any .jpg or .png file. Keeping the proxy design pattern in mind, we have ensured that the rest of the code will not be affected when we delete the “fake” camera and introduce a real one. All the functions that are attached to this program will all read the same and will all receive the same kind of data with the same usage.

A final major design pattern that we have used is the command design pattern. This command pattern is especially useful when we have multiple uses for a single feature. Speaking in examples, instead of using our readPlate() function to activate the camera, we use the camera to trigger the readPlate() function instead. This is a better way to do it because our camera can trigger a series of events using an if-else function rather than having to keep on running the readPlate() function in an infinite loop and also we do not have to go in and repeat cumbersome code in every place the camera is needed. This way of triggering saves us a huge amount of headaches if we were to ever change what camera we use to read plates. With a command design pattern, we can simply just change the underlying code of the camera and in the end, still end up with the same “true” value coming out of interface. The rest of the code will understand that command and all will continue as it should; all this without having to mess with any part of the project other than the affected interface.

Object Constraint Language:

Classes	Invariant	Precondition	Postcondition
Account	Context account inv:self.useraccess	pre:self.useraccess=true	Post:self.userAccess= self.webaccess
Manager	Context manager inv:self.salary	Context manager :: (s:integer) Pre:self.salary=0	Context manager::(S:integer) Post:self.salary= h*w
Garage	Context garage inv:spots	Context garage::(g:integer) Pre: self.spots=vacantSpots	self.spots=self.spots- self.spotnum
Camera	Context camera inv: String: self.licenseplate Context camera inv: String:self.blacklistlicen se Context camera inv: String:self.shapeofcar Context camera inv: String:Price Context camera inv: Integer:Spot	Context camera: (p: string) self.licenseplate=0 Context camera: (z: string) self.blacklistlicense=onli neblakclistcar Context camera: (m: string) self.shapeofcar=0	If self.licenseplate =self.blacklistcar, return false else if shapeofcar(car): return Price,Spot
Reservation	Context reservation inv:self.reserve Context reservation inv:self.spotnum	Context reservation::(r:integer) Pre:self.spotnum=true(sp ot is available)	If self.spotnum=true then self.spotnum=self.spo tnum-1(spot is reserved and there is

			one less spot)
			else
			return false

Price	Context price inv : self.hour Context price inv : self.month	Context price :: (h:integer) Pre.selfhour=0 Context price:: (m:integer) Pre.selfmonth=0	Post:total = self.hour + self.month
Notification System	Context notification inv: String: self.email Context notification inv: String self.cost Context notification inv: String: self.timestart Context notification inv: String self.timeend Context notification inv: String self.qrcode	Context notification: (l: False) Pre.hasreservation= True Context notification: (n.False) Pre.selftimereservation=0	Post: If self.reservation= True and self.timereservation!= expire, sendemail(self.email) return self.cost, self.qr code, self.timestart,self.cost self.timeend

9. System Architecture and System Design:

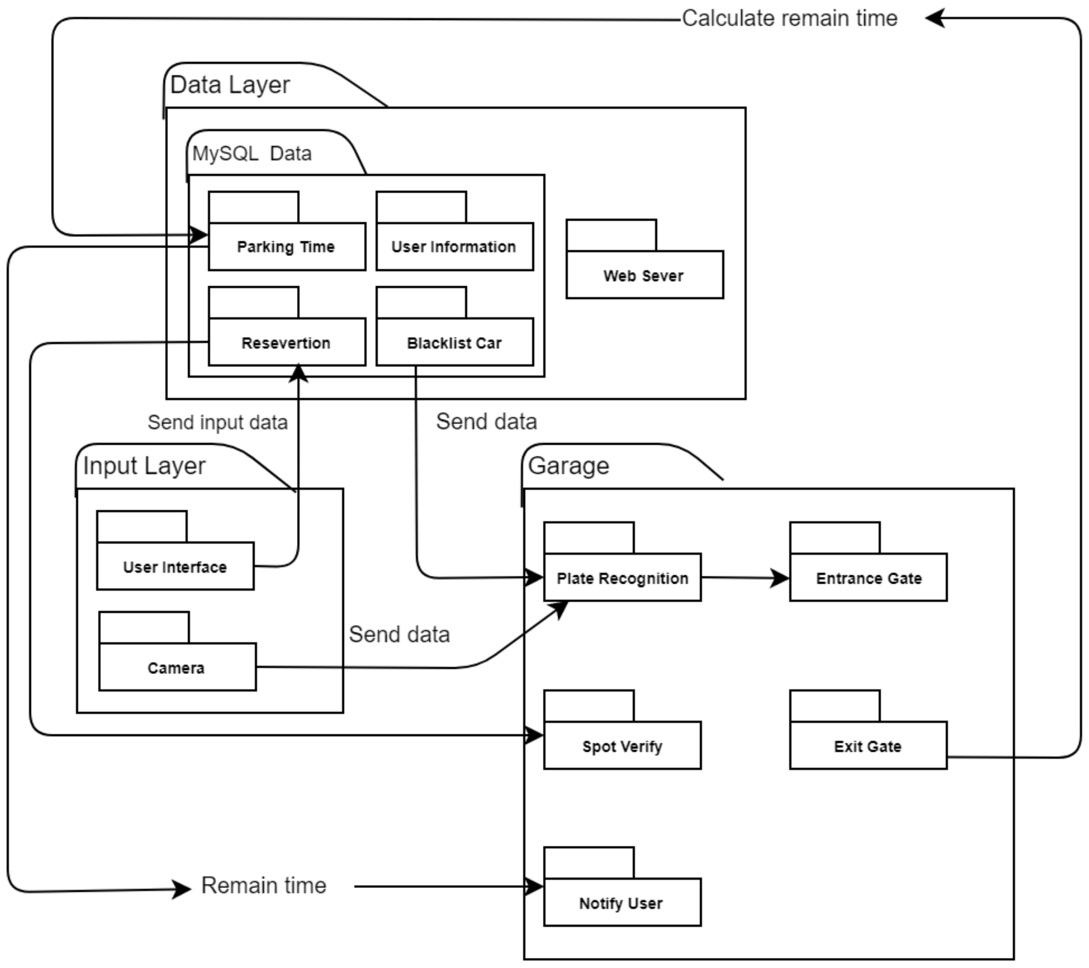
Architectural Styles

As to the structure of the garage logic models, the Automated Parking Garage system follows the Component-Based Development(CBD) design style. In the CBD design style, one component can be configured independently, but it is also restricted by others. In order words, all the processes of the system can be divided and put into each of the component, so that the designer can work on each component respectively. However, the designed component must be made on the basis of the communication with other components. That communication is called as the interface. For instance, in the Automated Parking Garage system, the scanning system handles recognizing the personal information combined with vehicle's license plates. So that the designer can design the scanning system with the processes like: requirements specification, component analysis, requirement modification, system design with reuse, development and integration, and system validation. Since the information read from license plates is related to the customers personal information and their status of reservation, the scanning system is definitely made based on the component of users' accounts and parking reservations.

Besides CBD design style, the project also applies a Database-centric design style. Since all the data such as user accounts with personal information, reservation status, real-time parking price, available parking slot, remaining parking time and even the payments information are stored within the database. The database links different components of the Automated Parking Garage system, and helps to achieve their functions.

What is more, this system is also heavily based on the event-driven design style. This design style means that one step in the system processes is triggered by the completion of the previous step. For example, the scanning system can recognize the information of incoming customers only if they have already created an web-account. Or, the registered customers can only enter the garage once they have made an online reservation.

Identifying Subsystem



The subsystems of our System Architectures comprises of the:

- For the Garage Model
    - Entrance Gate
    - Exit Gate
    - Spot Verify
    - Notify User

- Plate Recognition

- For the Input Layer
  - User Interface
  - Camera
- For the Data Layer
  - My SQL Data
  - Web Server

The User Interface is a place where user can input their personal information via register, login and make payment as well as reservation. All of those information will be stored in Data Layer.

The Blacklist Car subsystem in Data Layer will send the list of stolen or wanted cars to the Plate Recognition subsystem and combine with the real-time live-stream from the Camera, it can make the comparison with the user's plate and it will send notification to the garage owner if the plate is in the blacklist.

When the car enters the garage, the time elapsed starts counting and the garage will send the time elapsed to the database to calculate the remaining time of the user when the car leaves the garage.

Mapping Subsystems To Hardware

The docker function can apply to all the garage that cooperates with our services and connecting the database with each other. Our web services and system will be able to work on multiple devices. Specifically, our system can be able to communicate with Visual User Interface. And the Visual User Interface can handle and transmit the data which is registered by the customer and the data which is provided by Camera,License Plate Detector to the Information Database. The main objective was to make sure that the data collected from the garage is valid when compared to the data in the database regarding online reservation customer and can be stored regarding walk-in customer after implementing the blacklist license plate recognition system.

Persistent Data Storage

Most data collected by the system will be stored in a mySQL database and some data from there will be stored on the blockchain. In both cases, the storage drives used to save this data will be persistent. Persistent storage simply means that our saved data will not be lost nor will it be irretrievable if our hosting machine ever shuts off or fails.

Using a mySQL database, we can store information such as the name or credit card number of the customer, as well as real-time information such as open parking spots. This data is persistent yet mutable because it can be changed. For example, customers can change their credit card numbers or the availability of a parking spot can change. Here, once some data is overwritten, it is gone. That is fine for some applications, however, other applications require data stored to be immutable.

Such a requirement can be satisfied using a blockchain. We can store key data such as transactions, user IDs, and payment method charges in a permanent, immutable manner. This is possible because the blockchain has a feature built-in that makes it so that no data stored on it can be overwritten, ever. Moreover, the data on the blockchain is distributed among lots of different computers called “nodes”. This adds another layer of persistence because even if one machine fails, the other machines (nodes) on the chain will still have the data saved on their respective storage drives. This way, the data is always available and can never be changed.

Network Protocol:

Since we have multiple servers requirements in the project we also need a protocol for communication between the servers. Each of these servers with be hosting containers which will be migrated between the garage locations for distributed information exchange. The migration procedure includes the integration of the target network with the legacy routing, to provide a gradual path for enabling data exchanges. In the initial stage, the network connected data centers through legacy nodes using External/Internal Border Gateway Protocol (BGP) and ISIS routing. Note that, from the control plane point of view, only one TCP/BGP state is maintained per session.

In the starting network, the Provider Edge (PE) router runs BGP with external BGP speaking peers. Besides external peerings, the PE router also maintains internal peering sessions.

Typically, all BGP sessions as well as policies are configured manually using vendor specific CLI.

It is the responsibility of the PE router to negotiate BGP session parameters with both internal and external BGP speakers and maintain BGP and TCP state machine for each neighbor session. The PE router typically peers with external neighbors using the connected subnet address of the neighbor while for iBGP; neighbor relationship is established using loopback addresses. For next-hop reachability, the PE router also runs an Interior Gateway Protocol (IGP). Each PE router may receive more than one copy of the same prefix from different external peers or route-reflectors. As a result, it calculates the best path for installation in the routing/forwarding table while maintaining multiple copies of prefixes in the BGP database.

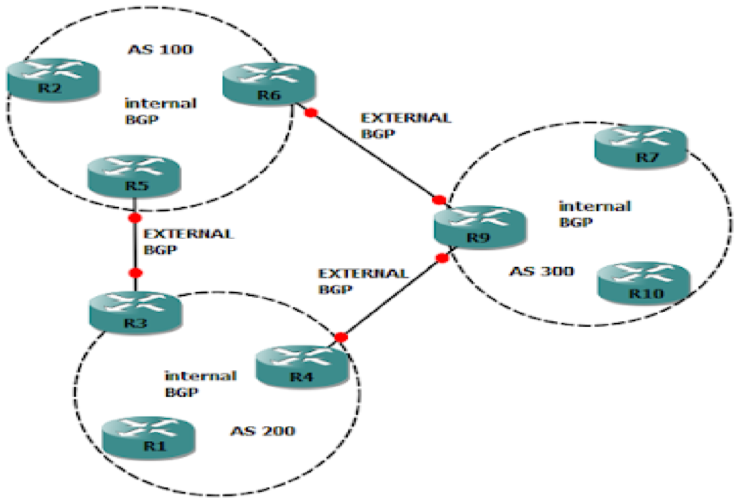


Figure 9-1 BGP Database

Global Control Flow

Time Dependency

There is time dependency in the system. The application involves real-time dependency rather than event-based time dependency. When a customer decides to make a reservation, they will be picking a real time of the day. The system will record this and keep that certain parking spot blocked off for the reserved time. Even after the customer checks-in, the system will track how much time the customer has remaining (all the calculations will still be based on real clock time).

Execution Orderliness

There is linear execution in the system. Customers will sign-up for an account online, enter their payment information, make a reservation online, have their car license plate number read, have their reservation verified, be guided to their parking spot, have their remaining time information be available, and in the end, be guided out of the parking lot. This will happen to every customer, every time, even if a customer is walk-in. A walk-in customer will still be forced to follow the same sequence of steps by the system; they will still have to sign-up and reserve before being allowed to enter the lot.

Concurrency

There will be major concurrency in the system. Each car and parking spot will have threads of their own. Each car will have driver data, license plate info and remaining time information attached to it. Each parking spot will have next availability, spot size, location in lot information attached to it. All the threads will be unique and will have to be handled on a case-by-case basis.

Hardware Requirements:

1. **Tablet 8GB(min):** as the Visual User Interface
2. **LED Display:** for outside to show vacancy of the garage also inside elevator
3. **Cameras:** 10MP for security check
4. **Driver's License Reader:** to quickly obtain information with one swipe
5. **License Plate Reader:** 10MB min to read the plate
6. **Database:** 100GB at least to store all the information.
7. **Hard-drive:** At least 50 GB of space. This disk will store only cached information. With an expectation of supporting 1M users, we have the advantage of shutting down the containers for storage availability.
8. **Server:** to process the information valet interface and the license plate reader
9. **Network Bandwidth:** to connect the garage system with the garage. (At least 2Mbps)

10. Algorithm and Data Structure:

Algorithms

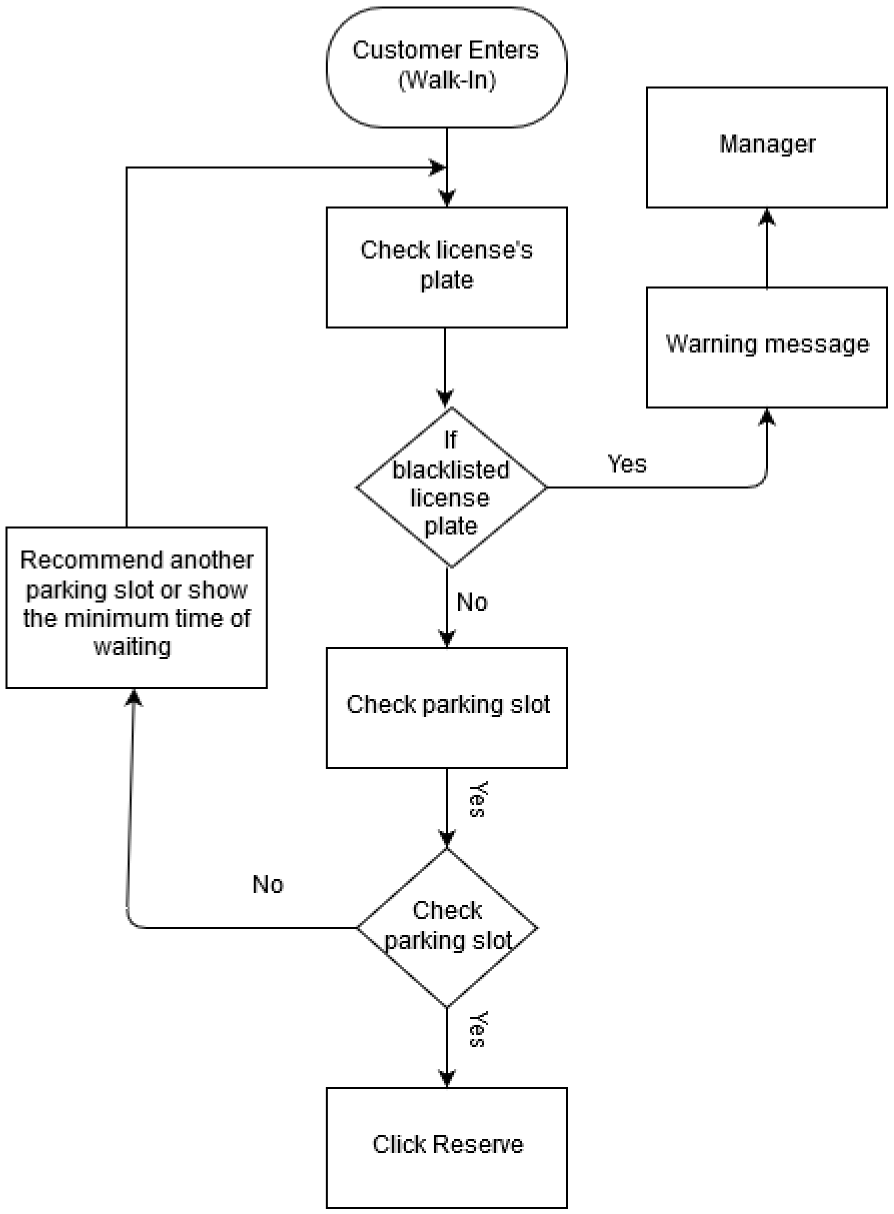
a. Garage

The mathematical model for the simulation for arrivals and departures will be tested in order to check if the system operates normally. The algorithm will be closely described in the following statements. This algorithm will execute in an infinite loop until the garage is full or until the working hours has exceeded.

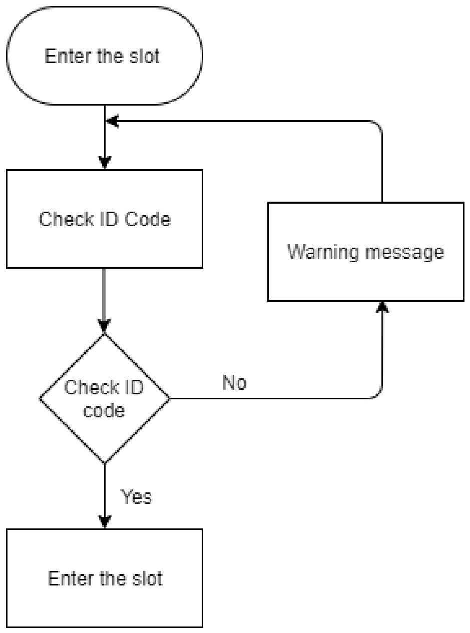
There will be an algorithm to arrange the slot in the garage. Each slot of the parking lot will be assigned as an integer item of the array and each of the items in the array will be filled when a car enters and will be removed when that car exits. However, the three critical for the system would be the beginning time, ending time and the QR code when entering the garage. There will be an algorithms to check if the car is exceeding the parking time to send a warning message regarding that. Additionally, when the code is giving to the customer, there will be another algorithm to check whether the QR code is fit with this slot or not.

Another important algorithm is to check whether the parking garage has slots or not, if does, we assign the slot to each customer, if doesn't, the system will show the minimum time to wait for the current customer to leave or recommend another nearing parking lot that implements our system.

- **Enter the garage:**


Customer Enters (Walk-In)
Check license's plate
If blacklisted license plate
Yes
Warning message
Manager
No
Check parking slot
Yes
Check parking slot
No
Recommend another parking slot or show the minimum time of waiting
Yes
Click Reserve

- **Enter the slot of the garage**



b. <u>Frequent Customer(Silver,Gold,Platinum)</u>

There will be an algorithm to detect if the customer with that license plate is coming here frequently or not in order to know each rank which should be assigned to each customer and assigned a reduction in price for each rank respectively.

For Silver, a reduction in price will be 0.05 when customer comes in at the second time.

For Gold,a reduction in price will be 0.1 when customer comes in at the third time.

For Platinum,a reduction in price will be 0.15 when customer comes in at the fourth time.

c. <u>Encryption:</u>

An average transfer of data between any two actors has a man-in-the-middle risk, where a third bad actor can tap the message midway and see what is inside. To make such attempts at theft meaningless, we will use asymmetric encryption to secure various forms of transactions between us and the customers; this will protect key customer information by making any records unreadable and virtually uncrackable.

We will accomplish this by using the following mathematical technique:

1. We will give the customer a *public key,* which is the left side of the equivalency:

   $m^{E}\ mod\ N^{■} \equiv c^{■}$

2. The customer will replace “m” with their intended message using a padding scheme and send us back the encrypted message “c”.
3. We will decrypt the message using our *private key* and use the content of the message. The message could be the license plate number of the customer’s car, for example.
4. Following the example in step 3, we would then record this transaction in our entry/exit ledger (in its encrypted form).
5. In the event of the theft of this ledger, the customer’s privacy would be safe and their travel history will not be exposed. This is due to the fact that the bad actor does not have the *private key* required to make actual meaning of all the data in the ledger.
6. Any bad actor who decides to brute-force decrypt the data would be stuck solving the puzzle for decades or longer.

The determination of the perfect private key is where the largest amount of mathematical modeling is required. We have to make sure that even if the public key is known, the private key cannot be cracked using it. We also have to make sure that the private key cannot be easily cracked. In order to achieve these objectives, we would have to pick the right values for “N”, “E” and a third letter “D”, which will be used to undo the effects of “E”.

“E” can be any small number greater than 1 of our choosing, with the only condition being that it cannot share any factors with $\phi(N)$, which is something we’ll explore soon.

To find “N”, we will pick any two really large, preferably 100 digits or more, prime numbers and multiply them together:

$$N = P_1 * P_2$$

To understand why this is the best idea, we will first have to define something called the breakability of a number “X”. The breakability of a number is the amount of numbers ≤X, but greater than 1, that do not share any common factor with X.

The best “X” to pick therefore is a prime number. This is because a prime number cannot be broken up more than twice, once with a factor of 1 and another with a factor of itself. This property makes it so that there will be no ambiguities in future calculations.

Here is the formal definition, when X is a prime number:

$$\phi(X) = X - 1$$

The following property is also valid for the multiplication of two prime numbers :

$$\phi(A * B) = \phi(A) * \phi(B)$$

We can then port this information with “N” this way:

$$\phi(A * B) = \phi(A) * \phi(B)$$

$$\phi(P_1 * P_2) = \phi(P_1) * \phi(P_2)$$

$$\phi(N) \ = (P_1 - 1) * (P_2 - 1)$$

This part is extremely difficult to do in the other direction and is the reason, why we would want to keep it $\phi(N)$secret.

The number “D” is another number we would want to keep secret. Why? Because as we will see, this number is really the one that will unlock the encrypted message easily and in one calculation. This is our *private key code*.

To begin to find “D” we must define Euler’s Theorem. This theorem utilizes $\phi(N)$, the breakability of N, that we found from a previous part in its definition.

Here is how the Theorem is defined, using our variables:

$$m^{\phi(N)} \equiv 1 \, mod \, N$$

Using this and some properties of modulus mathematics, we can logically deduce the following:

$m^{k*\phi(N)} \equiv 1 \, mod \, N$; because 1^{k}■always equals 1, not matter the k.

$m * m^{k*\phi(N)} \equiv m \, mod \, N$; because any number times 1 equals itself.

$m^{k*\phi(N)+1} \equiv m \, mod \, N$; using basic rules of exponents

We can use this definition on the original public key: $m^E \, mod \, N$■ $\equiv c$■, and find the following, assuming that we’re looking for a “D” that depends on $\phi(N)$:

$$m^{D*E} = m \, mod \, N$$

Then, $D = (k * \phi(N) + 1)/E$

All this means is that our secret $\phi(N)$factorization has been utilized in a way that incorporates the non-secret parts E and N. We can now use D to transform the encrypted “c” message to the decrypted message “m”.

In conclusion, by incorporating all the steps, we will have an extremely strong encryption system. This will put the customer’s mind at ease, knowing that even if ledger data is leaked, their information is safe.

d. <u>Dynamic Pricing:</u>

Price setting is one of the most important problems for any entrepreneur because any price setting error directly results in lost profit. However, traditional price management methods almost never achieve optimal pricing because they are designed for traditional environments, where the frequency of price changes is inherently limited and the complexity of pricing models is constrained by the capabilities of off-the-shelf tools and manual processes.

At the beginning of each time period the firm determines a selling price $pt \in [pl\ ,$ $ph]$. The prices $0 < pl < ph$ are the minimum and maximum price that are acceptable to the firm. After setting the price, the firm observes a realization dt of the demand Dt(pt), which is a random variable, and collects revenue $pt \cdot dt$. We assume that the inventory is sufficient to meet all demand, i.e. stock-outs do not occur. The random variable Dt(pt) denotes the demand in period t, against selling price pt. Given the selling prices, the demand in different time periods is independent, and for each $t \in N$ and $pt = p \in [pl\ , ph]$, Dt(pt) is distributed as D(p), for which we assume the following parametric model: $E[D(p)] = h(a\ (0)\ 0 + a\ (0)\ 1\ p)$, (3.1) $Var[D(p)] = \sigma\ 2\ v(E[D(p)])$. Here $h : R+ \rightarrow R+$ and $v : R+ \rightarrow R++$ are both thrice continuously differentiable known functions, with $h˙(x) = \partial h(x)\ \partial x > 0$ for all $x \geq 0$. Furthermore, σ and $a\ (0) = (a\ (0)\ 0\ , a\ (0)\ 1\ )$ are unknown parameters with $\sigma > 0$, $a\ (0)\ 0 > 0$, $a\ (0)\ 1 < 0$, and $a\ (0)\ 0 + a\ (0)\ 1\ ph \geq 0$. Write $et = D(pt) -$ $E[D(pt) \mid p1, . . . , pt-1, d1, . . . , dt-1]$. We make the technical assumption on the demand that for some $r > 3$, $\sup\ t \in N\ E[|et|\ r \mid p1, . . . , pt-1, d1, . . . , dt-1] < \infty$. The expected revenue collected in a single time period where price p is used, is denoted by $r(p) = p \cdot h(a\ (0)\ 0 + a\ (0)\ 1\ p)$; to emphasize the dependence on the parameter values, we write r(p, a0, a1) = p · h(a0 + a1p) as a function of p and (a0, a1). We assume that there is an open neighborhood U ⊂ R 2 of (a (0) 0 , a (0) 1) such that for all (a0, a1) ∈ U, r(p, a0, a1) has a unique maximizer p(a0, a1) = arg max pl 0 can easily be captured by replacing p by p − c. A pricing policy ψ is a method that for each t generates a price pt ∈ [pl , ph], based on the previously chosen prices p1, . . . , pt−1 and demand realizations d1, d2, . . . , dt−1. This pt may be a random variable. The performance of a pricing policy is measured in terms of regret, which is the expected revenue loss caused by not using the optimal price popt. For a pricing policy ψ that generates prices p1, p2, . . . , pT , the regret after T time periods is defined as Regret(T, ψ) = E "X T t=1 r(popt, a(0)) − r(pt, a(0)). The objective of the seller is to find a pricing policy ψ that maximizes the total expected revenue over a finite number of T time periods. This is equivalent to minimizing Regret(T, ψ). Note however that the regret can not directly be used by the seller to find an optimal policy, since its value depends on the unknown parameters a (0) .

Data Structures

Our system will indirectly use many data structures such as hash tables, linked lists and bitmaps. Since we are storing huge amounts of data, data structures are a must if we want to achieve high performance and efficiency.

Blockchains use something called a “hash pointer” combined with linked lists. A hash pointer is used as the “previous” pointer in a linked list. The linked list in a blockchain is not like a traditional linked list, where a block holds the data element and a “next” pointer. A blockchain linked list still uses a block to hold data but instead of using a “next” pointer, it uses a “previous” pointer. This is done in order to prevent any tampering. If a block is tampered with, the "previous" pointer of the next block will have a completely different calculated value and that data will be completely inaccessible and unreadable (as will all the blocks in the rest of the chain). What is special about using a hash pointer is used as a previous pointer is that a hash pointer hashes the data of the previous block all the way down to the first ("genesis") block. This means that in order for a bad actor to crack the data in the chain, he has to crack every single block simultaneously. And even when he does, he cannot crack the genesis block because that key is hand-created by the creator.

Another data structure our system is using is a bitmap. A bitmap is simply a two-dimensional array that we can tweak to store objects. These objects would contain

11. User Interface and Implementation:

Sign up - UC1:

A. Preliminary Design

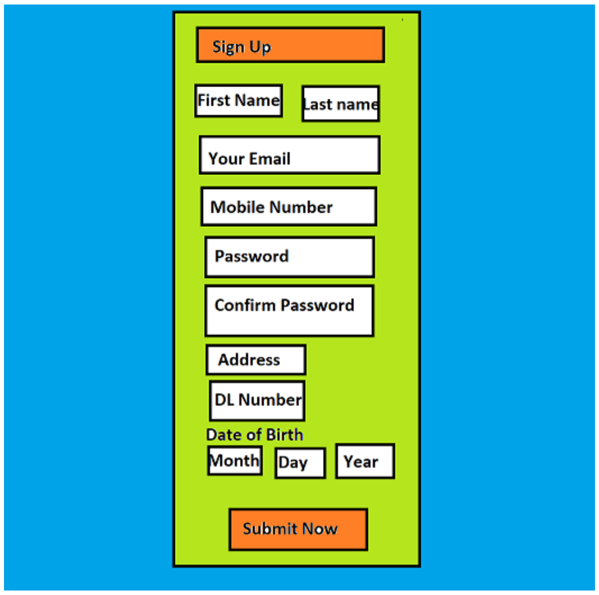


Figure 11-1 Designed Sign up page

Sign in - UC2:

A. Preliminary Design:

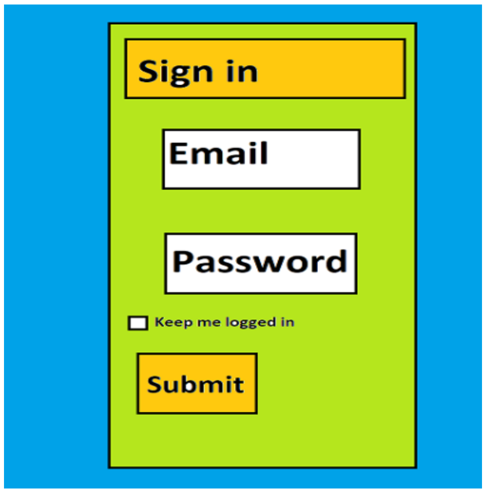
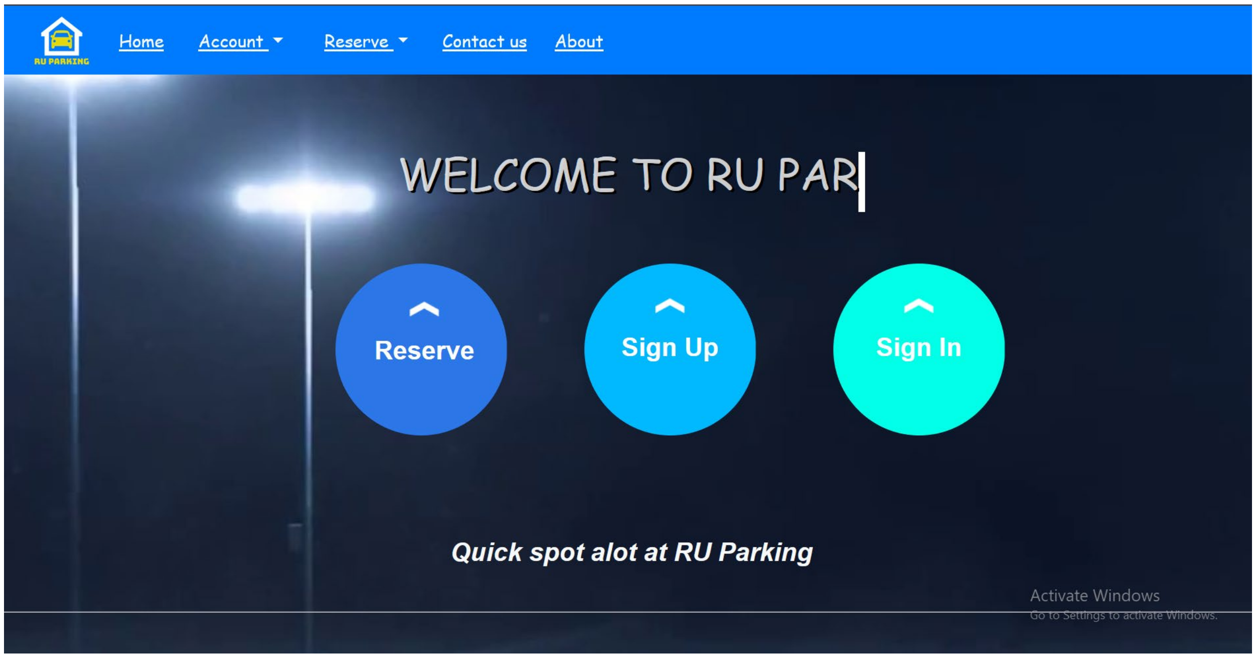


Figure 11-2 Designed Sign in page

Home page:



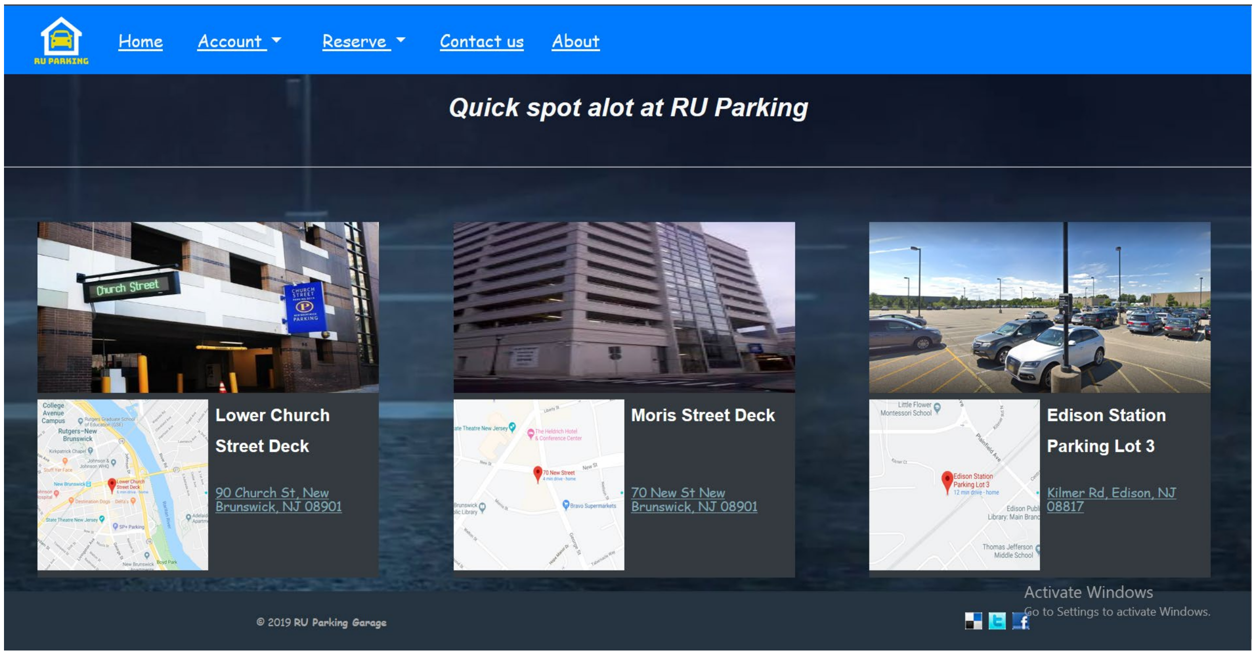

RU PARKING
Home
Account
Reserve
Contact us
About
Quick spot alot at RU Parking
Lower Church Street Deck
90 Church St, New Brunswick, NJ 08901
Moris Street Deck
70 New St New Brunswick, NJ 08901
Edison Station Parking Lot 3
Kilmer Rd, Edison, NJ 08817
© 2019 RU Parking Garage
Activate Windows
Go to Settings to activate Windows.

Figure 11-3 Home page

- In the home page, we have designed new navigation bar so it will become friendlier with users. We design it as fixed position so it will not disappear when users scroll down.
- We also created "Welcome to RU Parking" with animation so the home page will look more fancy.

- We added 3 parking locations with the maps belong to each one. So that when we access the home page. If they do not know where to go, they just need to click the map. (**Total 1 click**)
- When people want to sign in or sign up they just need to click on "Sign Up" or "Sign In" button as shown in the picture. (**Sign Up: Total 1 click, Sign In: Total 1 click**)
- The main feature is reservation. There are 2 kinds: guest and customer. Guest is for walk-in customers and "customer" is for users that have registered. When they click on "Reservation" button it will show as below:

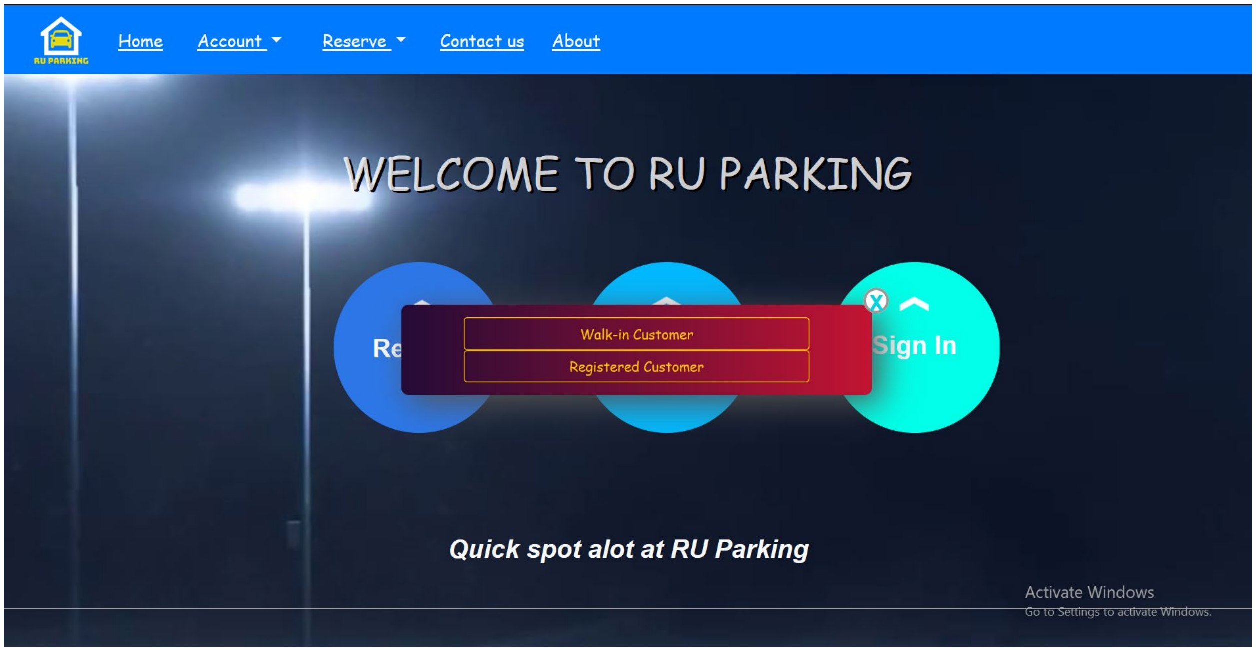


Figure 11-4 Reservation button

- If users are in rushed, they can choose "guest" so that I will jump directly to reservation page for walk in (**Walk-in: Total 2 clicks**)
- If users want to create an account so that it helps them easier next time reservation, they just need to click "customer" and it will jump to "Sign Up" page (**Customer: Total 2 clicks**)

<u>Sign Up Page:</u>

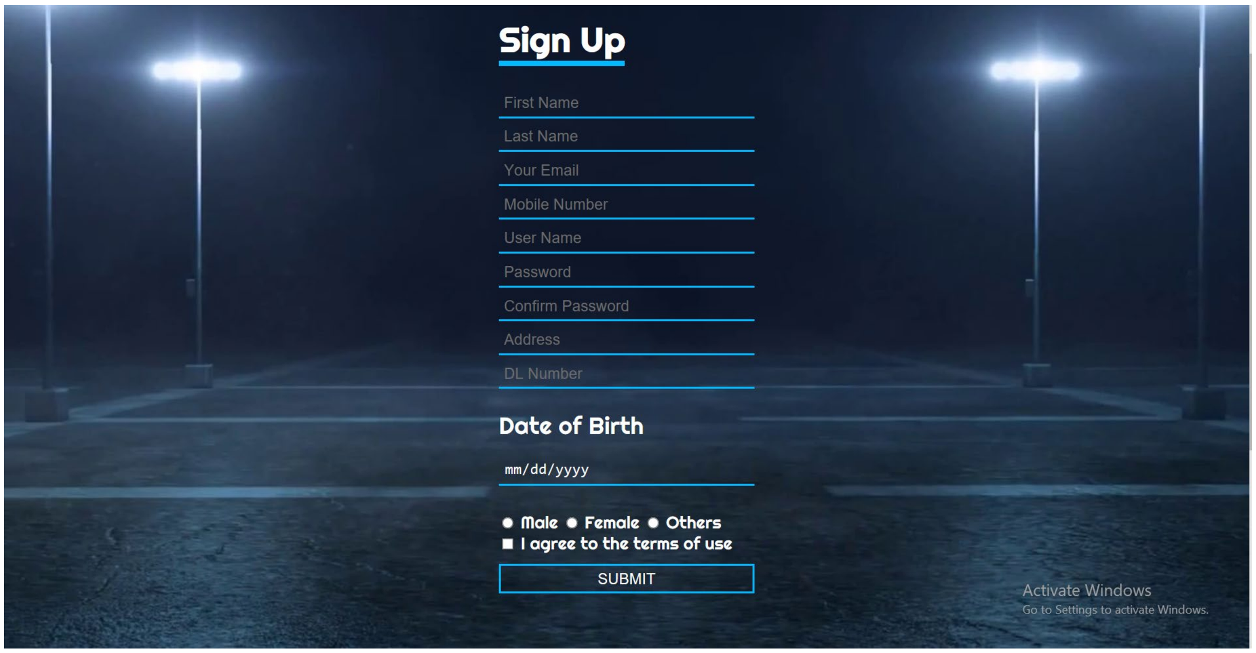


Figure 11-5 Sign up Page

Data Entry: Total 1 click on Sign Up and 14 keystrokes as follow:

- First Name
- Last Name
- Email
- Mobile Number
- Username
- Password
- Confirm Password
- Address
- Credit Card Number
- CVV
- Driver License Number
- Date of Birth
- Gender
- Agreement

2. Click button "Submit"

- This sign up page we only add 3 inputs: Username, Credit Card Number and CVV. Because the customers will be charged via their credit card.

Sign In Page:

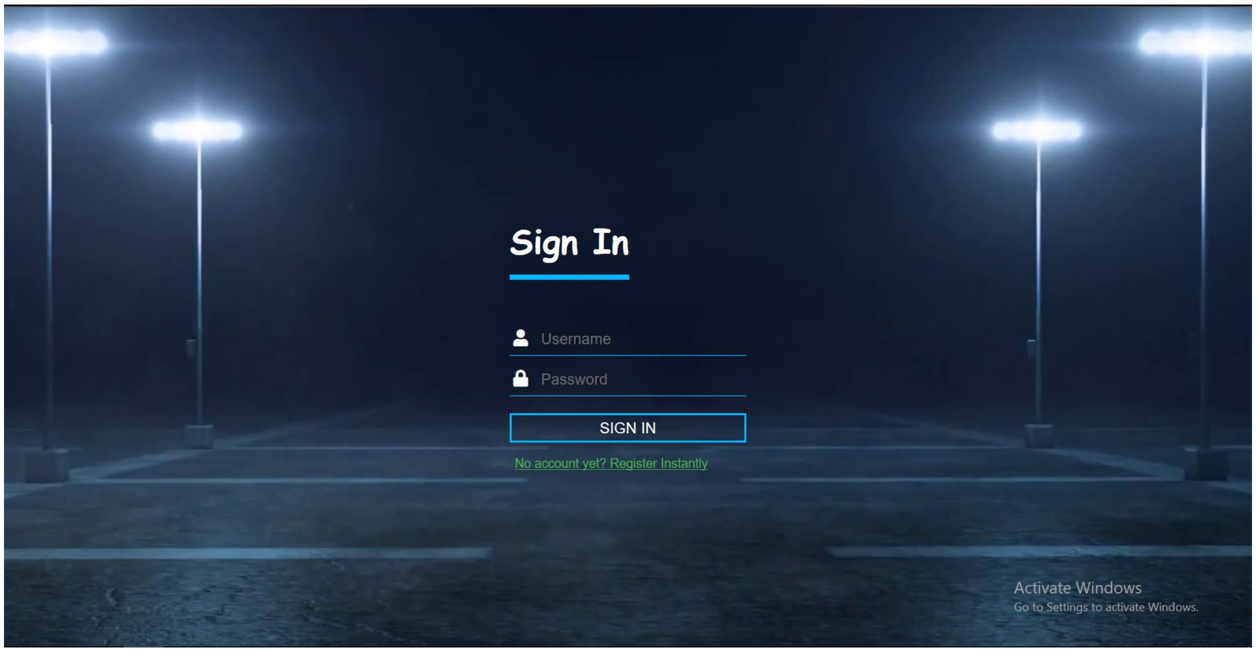

Sign In
Username
Password
SIGN IN
No account yet? Register Instantly
Activate Windows
Go to Settings to activate Windows.

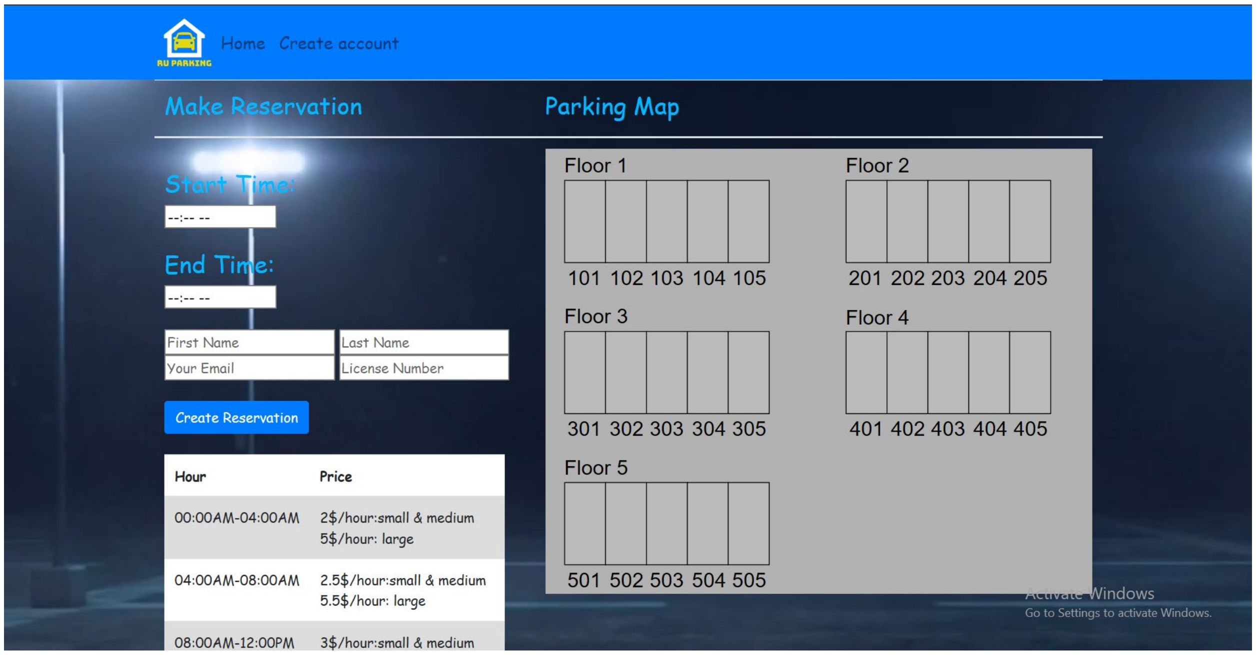
Figure 11-6 Sign in page

Data Entry: Total 1 click and 2 keystrokes as follows:

- Email (This is Username)
- Password

Then click button "Sign In". However, if users accidentally click on the sign in page when they have not signed up yet, they can click on "No account yet? Register instantly", so that with **only 1 click** it will bring user to sign up page.

Walk-in page:



Hour	Price
00:00AM-04:00AM	2$/hour:small & medium 5$/hour: large
04:00AM-08:00AM	2.5$/hour:small & medium 5.5$/hour: large
08:00AM-12:00PM	3$/hour:small & medium

Figure 11-7 Walk-in page

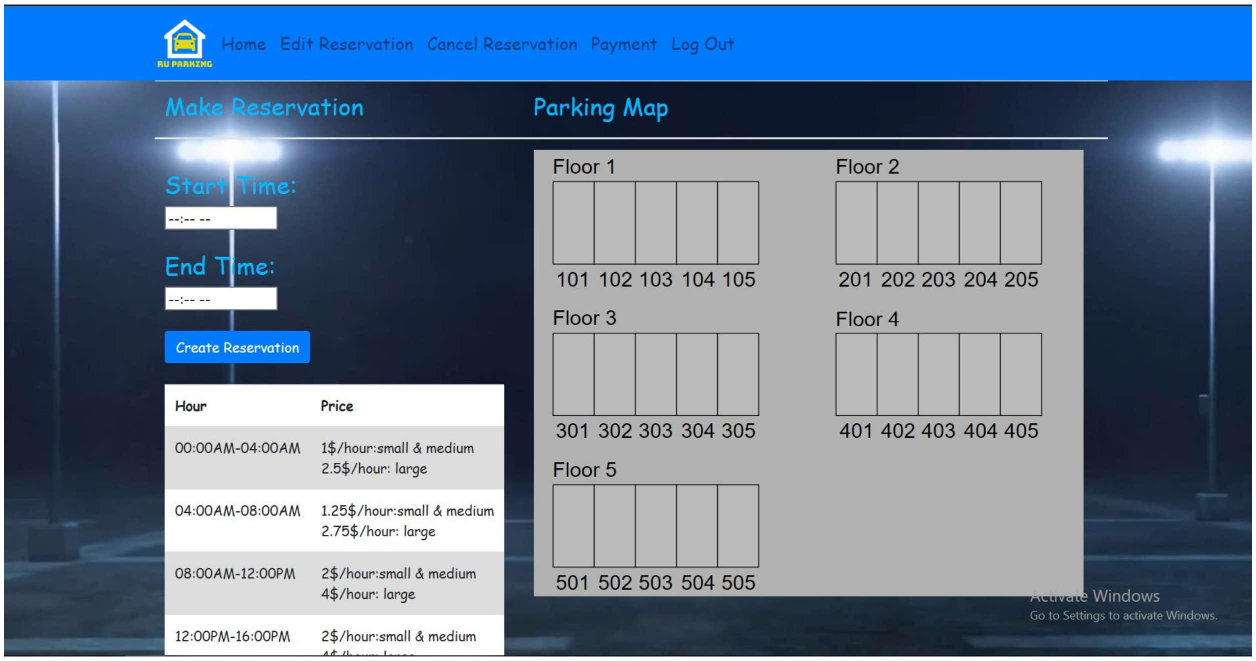
Data Entry: Total 1 click and 5 keystrokes as follows:

- Time and Date
- Slot Number
- License Plate Number
- Credit Card Number
- CVV

Then click button "Create Reservation"

This walk-in page is for walk-in customers, if they are in rushed they can reserve a slot quickly, only 5 keystrokes so it will not take much time. The parking map is just a "prototype" for the purpose of time. Therefore, in reality it will be bigger and more space.

Online Customer Reservation page:



Hour	Price
00:00AM-04:00AM	1$/hour:small & medium 2.5$/hour: large
04:00AM-08:00AM	1.25$/hour:small & medium 2.75$/hour: large
08:00AM-12:00PM	2$/hour:small & medium 4$/hour: large
12:00PM-16:00PM	2$/hour:small & medium

Figure 11-8 Online customer reservation page

Data Entry: Total 1 click and 2 keystrokes as follows:

- Time and Date

Then click button “Create Reservation”

This walk-in page is for online customers. The parking map is just a “prototype” for the purpose of time. Therefore, in reality it will be bigger and more space.

Payment page:

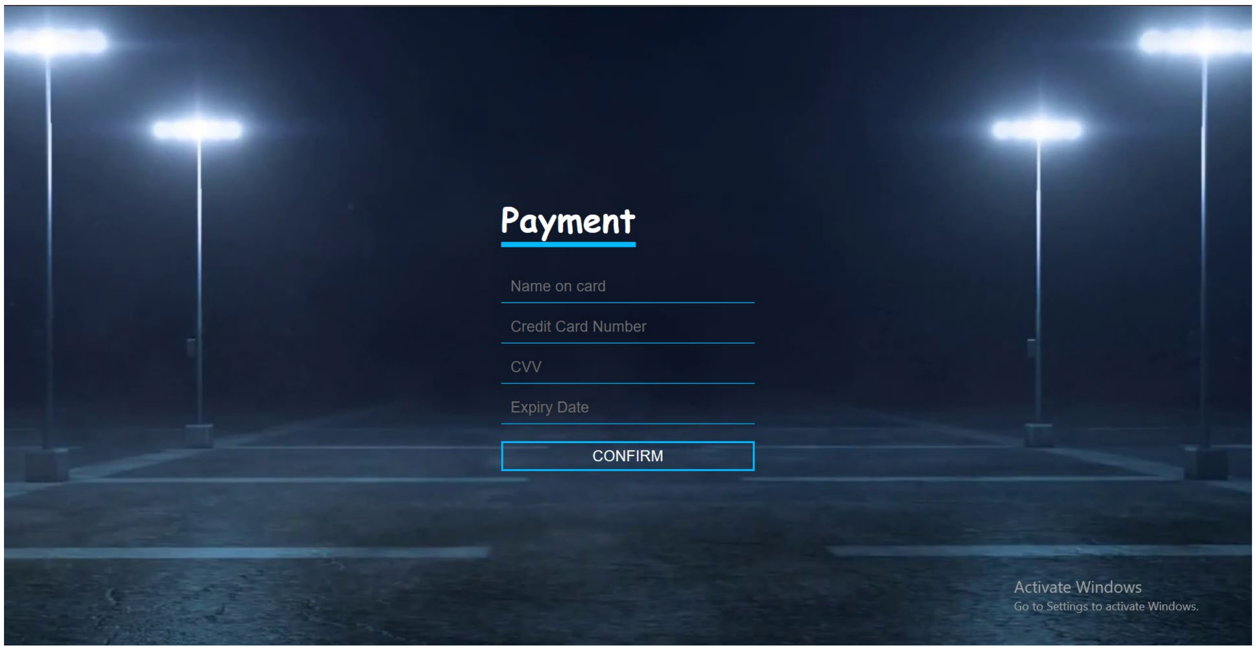


Figure 11-9 Payment

Data Entry: Total 1 click and 4 keystrokes as follows:

- Name on credit card
- Credit Card Number
- CVV number
- Expiry Date

Then click button "Create Reservation"

12. Design of Tests:

Each unit will be tested before merge into the system. The units that need to be tested are:

- Website
- Database
- Plate Recognition System
- Blockchain

Website:

Goal: The website is responsible for user to sign up, sign in and make reservation for both frequency users and walk-in users. Even for the walk-in customers who have not made a registration before, they can easily register an account with the help of managers. As a result, the website is also a place where the garage manager can check the information about the reservation and also do some necessary changes among different parking slots.

Test cases:

- Creating an account
  - To check if the users can create a new account with the right personal information.
- Making a reservation (by registered-customers)
  - When there are some parking slots available, this step is to check if the registered-customers (no matter walk-in customer or not) can make a reservation with: spot number; mobile number; credit card number and CVV. This step also aims to check if the registered-customers can choose each of the available slots they want.
  - When there is no parking slot available, this step is to make sure that the registered-customers cannot make a reservation in this situation.
  - Also to check if the customers can log out when finishing the reservation.
- Making a reservation (by unregistered-customers)
  - When there are some parking slots available, this step is to check if the walk-in unregistered-customers can make a reservation (by the help of managers) with: spot number; mobile number; credit card number and CVV. This step also aims to

check if the unregistered-customers can choose each of the available slots they want.

- When there is no parking slot available, the walk-in unregistered-customers cannot enter the garages. For that reason, this situation does not need to be tested.
- Also to check if customers can log out when finishing the reservation.

- View reservation status
  - To check if managers can view the reservation status of different parking slots.
- Walk-in registration
  - To check if walk-in customers can have an immediate registration and reservation.
- Edit information
  - To check if customers can edit their personal information and reservation information.
- Log-in
  - To check if registered-customers can login successfully with their correct username and password.
  - Also to check if the website can report an error if the customers fill in a wrong username or password.
- Dynamic parking price
  - To check if there shows a dynamic parking price when opening the reservation webpage.
- Payment
  - To check if the customers' can make a payment via the website with their bank cards or mobile phones.

Database:

Goal: The database is responsible for storing user information and keeping track of the time elapsed for each user within combination parking. The database is also responsible for updating the available slots for customers to book.

Test cases:

- Calculating time remain
  - To check if the database can receive the time elapsed of the customers' parking and make calculations about the remaining time for a next parking.
- Comparing license
  - To check if the database can receive the data of the plate recognized from the Plate Recognition system and make comparison with the plates from blacklisted cars.
- Reservation
  - To check if the data of the slots can be updated in real-time in order to help new-coming customers to make a reservation and prevent them from booking the reserved slots.

Plate Recognition System:

Goal: The plate recognition system aims to reduce the check-in time and increase the security for the customers.

Test cases:

- Recognition
  - To check if the system can detect the license via camera to capture the plate instantly.

- Send data
  - To check if the system can send data to the database immediately to start or end the time elapsed of the customers' parking and make comparison with the plates from blacklist cars.
  - To check how will the system handle it if there is no connection with the database.

Blockchain:

Goal: The purpose of using blockchain is to make sure that the customers' information is secure from and unreadable by hackers.

Test cases:

- Unreadability
  - To check if the Blockchain can be applied successfully in securing customer data by making sure the data on the blockchain ledger and hashtable is encrypted and therefore completely unreadable by any unintended party.
- Payment System
  - To check whether or not the blockchain and it's ledger are up-to-date with the latest transactions. Transactions such as making a reservation online, walking-in to the lot, canceling a reservation, editing a reservation, and staying overtime.
  - To check whether or not the blockchain will accurately calculate the monetary balance owed by each customer.
- Tamper-Proof
  - To check whether or not the blockchain and it's ledger can sustain any tampering of the database such as deletion of any transactions or manual changes to cost of service. To ensure that despite any tampering, the total balance owed by the customer reflects reality.
  - To check whether or not any direct tampering to the ledger will be accepted to the blockchain or not. To test the strength of the system of hash-pointers, a feature most prominent in blockchains.

13. History of work, Current Status, and Future Work:

History of work, current status:

In the report #1 and report #2, group 4 had set a bunch of goals for the project. With a deeper understanding of the project, group 4 made some adjustment about the goals and worked on them.

As for now, group 4 has already created a website for the RU parking. In the website, customers can create their own account to start their RU parking. Once creating an account, the customers just need to log in with the username and password and then make a reservation. For the walk-in customers, they can also make a reservation by using first name, last name, email and license number. The database in this project is used to contain the information of the customers and make sure the safety of the customers' information by using SQL Server Authentication. What is more, the database also contains the information for every reservation of the parking and it calculates the dynamic parking charge based the parking time and the size of the parking slot (size of the vehicles). This work also includes blockchain method of data privacy. Customers information is secured with the use of public and private keys. An asymmetric cryptography is used to secure transactions between users. The public keys are shared since they give away no data . Each user has an address that is derived from the public key using a hash function. These addresses are used to send and receive assets on the blockchain, with this technique the users can view past transactions and activity that has occured on the blockchain. If the database is compromised by an attacker and information is deleted or manipulated, even then the original data is available and stored in blockchain and hence the owner is at no risk of being vulnerable to any form of malicious attacks. We have also included the usage of docker migration assisted real-time information exchanges between garages for systematic functioning of multiple garage locations and without having a single point of failure of a database. Information such as combined payment, blacklisted car at a particular garage location and slot availability information are exchanged between the garage locations via docker migration such that the garages are aware of the status of the other locations.

Future Work:

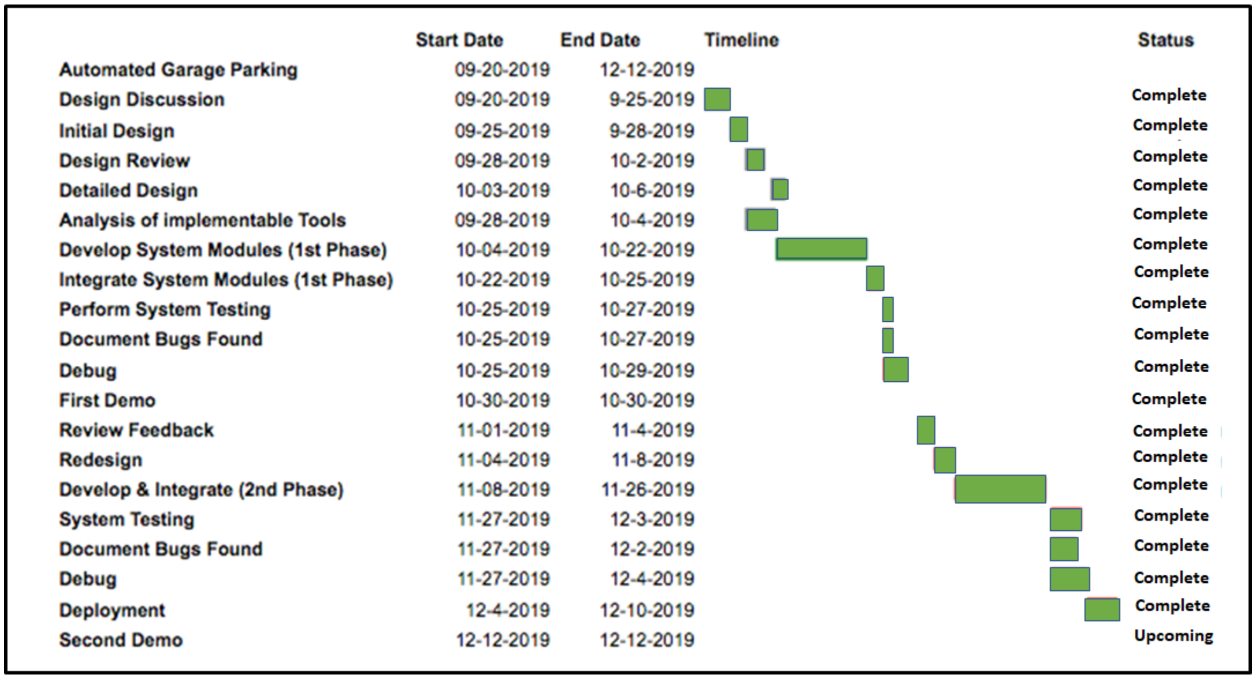
For the future work, now this project only involves three parking garages. To make the project more professional and scalable and make the test more trustworthy, the group will try to involve more general parking garages in order to verify that our software product can support huge scale emulation. Furthermore, when doing the detection for the size of the vehicles, this project only use small, medium and large to describe the size. In order to make more specific and more digitized description of the size, the group 4 will try to find other ways or improve the current shape detection algorithm for specific dimension detection. Also the face recognition of the customers is yet to be implemented for added security purposes which we would like to implement in the next phase.

14. Project Management:

To develop the report all team members were independently developing their assigned tasks of the report in one shared Google document for each part of the report. Once the two parts of each report is completed, they will be merged into a full report, shared by all members in a Google document.

Below is the gantt chart diagram which enlists detailed task breakdown and the corresponding estimated start date and end date. It should be noted that demo 1 and demo 2 has task duration of a day and hence has no visual bar. Description of each task is given below. All the bugs of our software are tracked and fixed until we obtain our desired product with the defined quality standards

Gantt Chart



	Start Date	End Date	Timeline	Status
Automated Garage Parking	09-20-2019	12-12-2019		
Design Discussion	09-20-2019	9-25-2019	🟩	Complete
Initial Design	09-25-2019	9-28-2019	🟩	Complete
Design Review	09-28-2019	10-2-2019	🟩	Complete
Detailed Design	10-03-2019	10-6-2019	🟩	Complete
Analysis of implementable Tools	09-28-2019	10-4-2019	🟩	Complete
Develop System Modules (1st Phase)	10-04-2019	10-22-2019	🟩	Complete
Integrate System Modules (1st Phase)	10-22-2019	10-25-2019	🟩	Complete
Perform System Testing	10-25-2019	10-27-2019	🟩	Complete
Document Bugs Found	10-25-2019	10-27-2019	🟩	Complete
Debug	10-25-2019	10-29-2019	🟩	Complete
First Demo	10-30-2019	10-30-2019		Complete
Review Feedback	11-01-2019	11-4-2019	🟩	Complete
Redesign	11-04-2019	11-8-2019	🟩	Complete
Develop & Integrate (2nd Phase)	11-08-2019	11-26-2019	🟩	Complete
System Testing	11-27-2019	12-3-2019	🟩	Complete
Document Bugs Found	11-27-2019	12-2-2019	🟩	Complete
Debug	11-27-2019	12-4-2019	🟩	Complete
Deployment	12-4-2019	12-10-2019	🟩	Complete
Second Demo	12-12-2019	12-12-2019		Upcoming

Task Phase Description

Design Discussion and Initial Design

- Basic UI design for sign up, sign in, registration and reservation page.
- Establish routes for each user case (determine which page goes to where)
- Preliminary ideas for blockchain assisted security of user information
- Determine the containerization technique for information exchanges

Design Review and Detailed Design

- Finalize the UI (HTML & CSS)
- For each use case determine user page interaction flow
- Research and simulation of the analytical model for blockchain assisted information privacy
- Blacklisted car detection
- Determine detailed module interaction and object finalization of each module

Analysis of Implementable Tools

- Determine the tools that would be used in container deployment (docker, lxd, kubernetes) and thoroughly understand the features of each of these platform-as-a-service product.

Develop and Integrate System Modules

- REQ1 and REQ2
- Car detection at the entrance and customer entry after reservation confirmation
- Depending on the spot choice bring the customer to the correct level
- Verify if the vehicle is parked in the reserved slot
- Offer slot choices to walk-in customers
- Email notification with reservation ID to the customer
- Charge customers depending upon dynamic pricing
- Encrypt customer payment and information details
- Depending upon customers parking history for the day provision combined pricing
- Provide information to the customer about parking slot availability at other garage location

System Testing

- Code is implemented completely and checked against our requirements whether it is addressing our needs which are gathered in the design and analysis phase or not.
- All the bugs of our software are tracked and fixed until we obtain our desired product with the defined quality standards.

Deployment

- Our product will be deployed for beta testing for finding bugs
- Depending on the users feedback the our system can be improved

Product Ownership

Member	Done		
Shalini & Khanh	Initial container configuration Creating containers Pushing files into the container Car number plate detection	Number plate recognition from video streaming	Running the automatic number plate detection app in the container Storing pricing and slots availability information in the container Migrating the container from one node to another (for information availability at other garage locations)
Luan, Zhuoyang	Creating website contains the main page with sign up page, sign in page.	Coding the map for parking lot and create reserve page (reserve form), mark the lots that have been reserved. Adding well designed web page layout for user friendly experience	Create confirmation page and successful reservation notification, log out features. Create pages for admin. Add more animations to make website more fancy. Integrating all the modules

Khanh, Duc, Tan	Doing research about database, back-end technology.	Building a Database using MongoDB to store key information for all parts of the project.	Using Node.js to deal with back-end. Process database. Connecting the database with the finished website and plate recognition systems.
Nainil	Creating the barebones of the post-sign page, where payments and reservations will be made.	Building an encryption system to protect customer data.	Building a fully-functioning blockchain to update and maintain the ledger of all transactions. Building solutions to keep the ledger secure, immutable & tamper proof. Connect blockchain to the database.

15. References:

1) Blacklisted car webpage: https://www.stolencar.com/Report/Search
2) Parking lot offences: https://www.bjs.gov/index.cfm?ty=tp&tid=44

Books:

- https://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

  Reports:

- Group#3_2018_project
- Group#5_2013_project

Web Resources:

- https://www.devteam.space/blog/how-to-integrate-a-blockchain-technology-into-your-project/
- https://www.redhat.com/en/blog/container-migration-around-world
- https://www.netapp.com/us/info/what-are-containers.aspx
- https://lxd.readthedocs.io/en/latest/
- https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work
- https://fortune.com/2018/06/27/facebook-data-privacy-blockchain/

UML Design:

- https://www.draw.io/