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
Zhuoyang Xiao - zx150
Nainil Patel - npp71

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

SCAN ME

Individual Contribution Breakdown

Project Category		Team Members						
		Shalini	Luan	Khanh	Duc	Tan	Nainil	Zhuoyang
Interaction Diagrams	UML diagram		16.6%	16.6%	16.6%	16.6%	16.6%	16.6%
	Description of Diagrams		16.6%	16.6%	16.6%	16.6%	16.6%	16.6%
	Alt Solution Descriptions		16.6%	16.6%	16.6%	16.6%	16.6%	16.6%
Class Diagram & Interface Specification	Class Diagram & Description		80%	20%				
			50%	50%				
	Traceability Matrix					100%		
System Architecture and System Diagram	Styles							100%
	Package Diagram				100%			
	Map Hardware			100%				
	Database			25%	25%	25%	25%	
	Network Protocol	100%						
	Global Flow Control						100%	
	Hardware Requirement	100%						
Algorithms				25%	25%	25%	25%	
User Interface	Design		100%					
	Description		100%					

	Testing Design								
	Project Management	Document Merge	100%						
		Project Coordination/ Progress	50%						
		Plan of Work	100%						

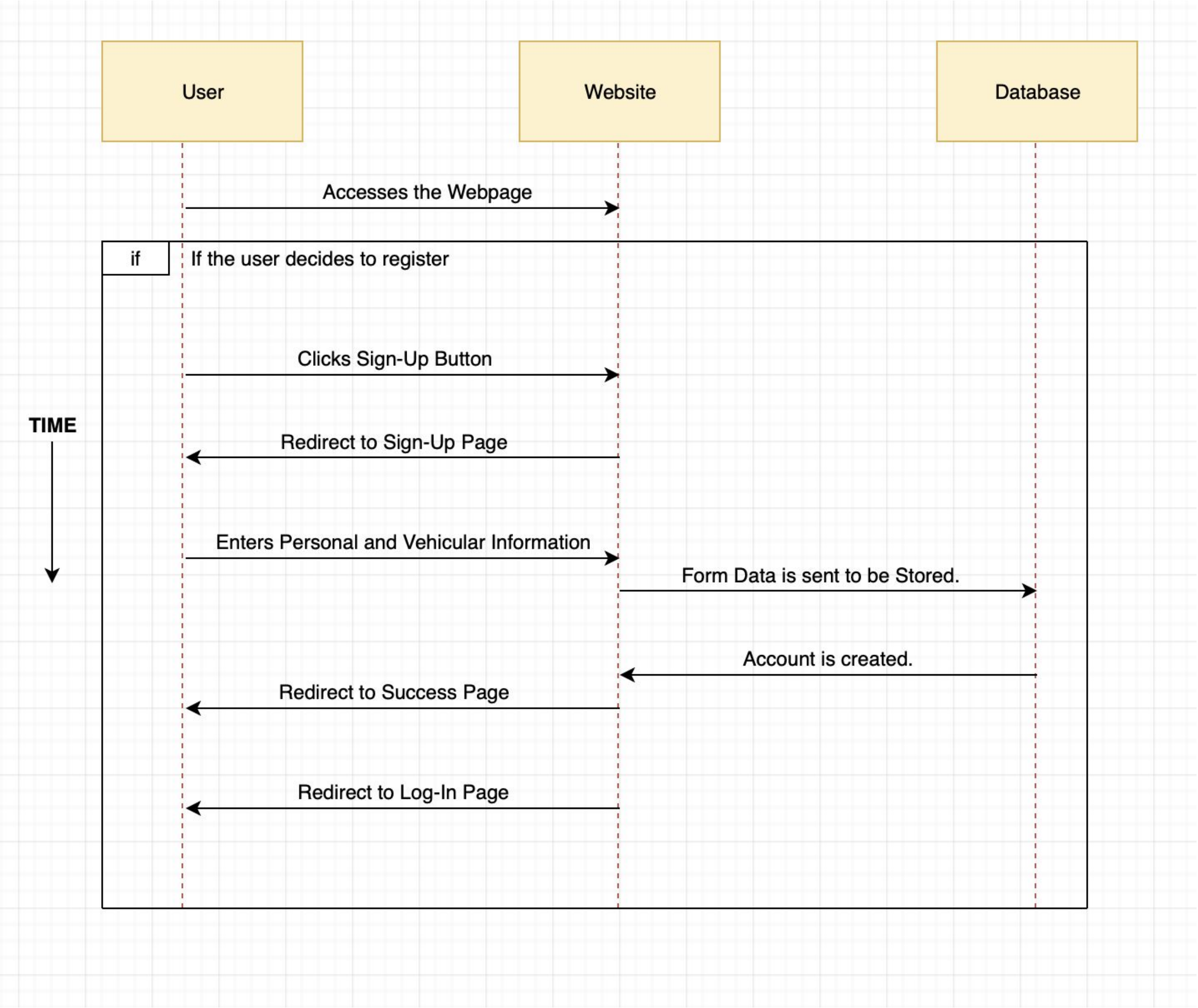
Note: This contribution table will be gradually filled as we progress with the tasks in the upcoming weeks.

Table of Contents

1. Interaction diagram

1.1 Registering Online:

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.



1.2 Login:

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:

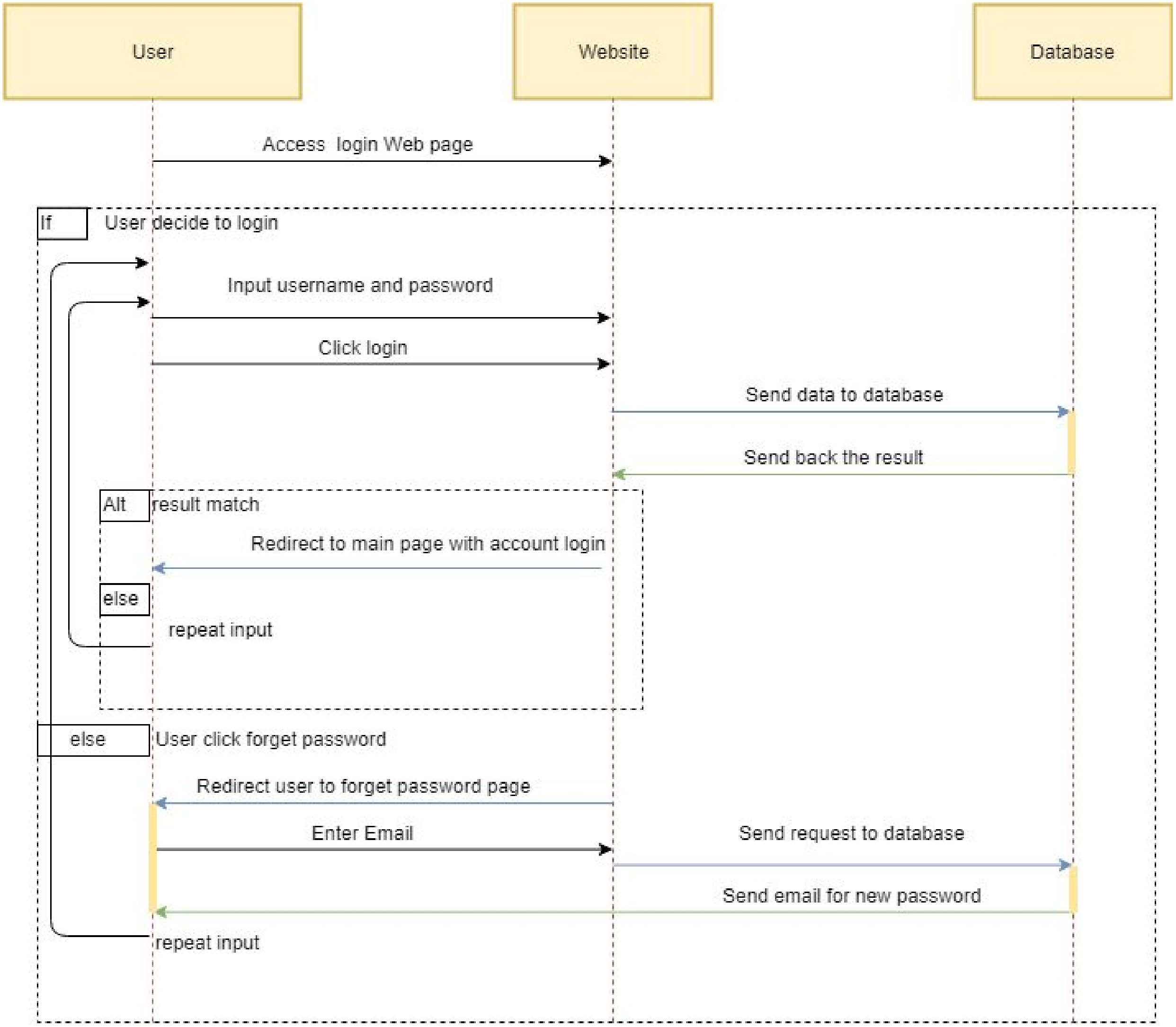
1. If user decide to login

   g) They try to input their username and password
   h) Click login
   i) Website will send the data to verify
   j) The database send back the result
   k) If the user id and password match in the system, website redirect user to homepage with their account login

   Else repeat step a)

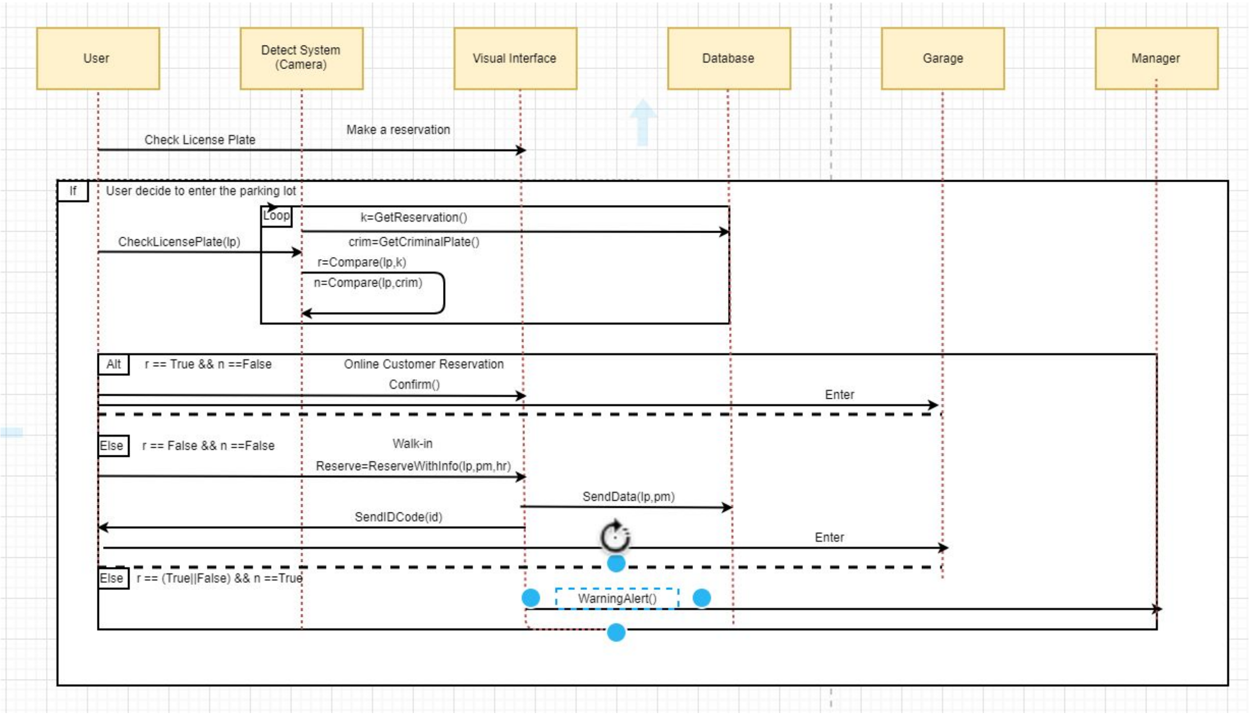
2. If user click forget password

   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


User
Website
Database
Access login Web page
If
User decide to login
Input username and password
Click login
Send data to database
Send back the result
Alt
result match
Redirect to main page with account login
else
repeat input
else
User click forget password
Redirect user to forget password page
Enter Email
Send request to database
Send email for new password
repeat input

1.3 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.



1.4 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.

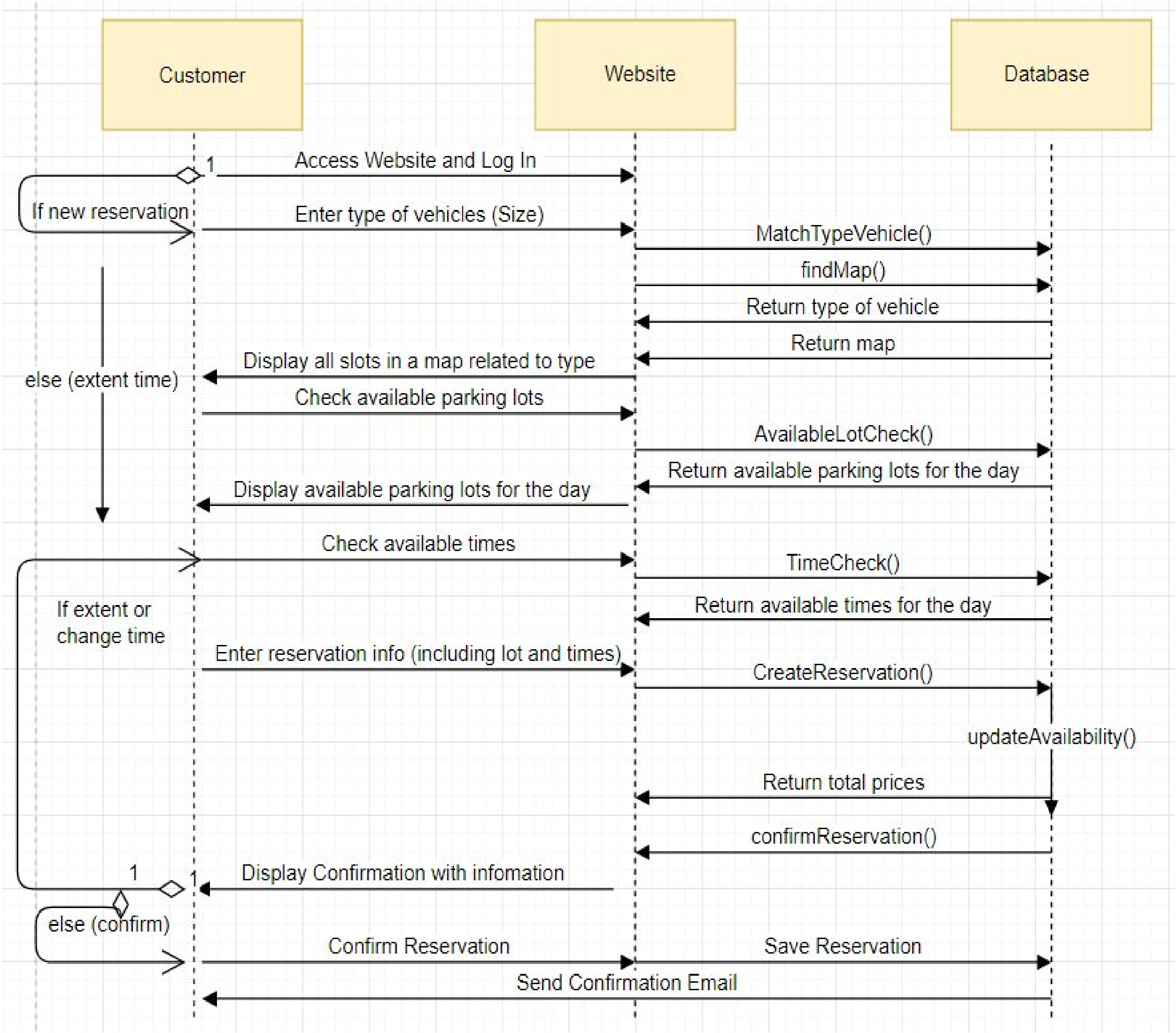
Flowchart of reservation:

1. If user decides to make a new reservation:

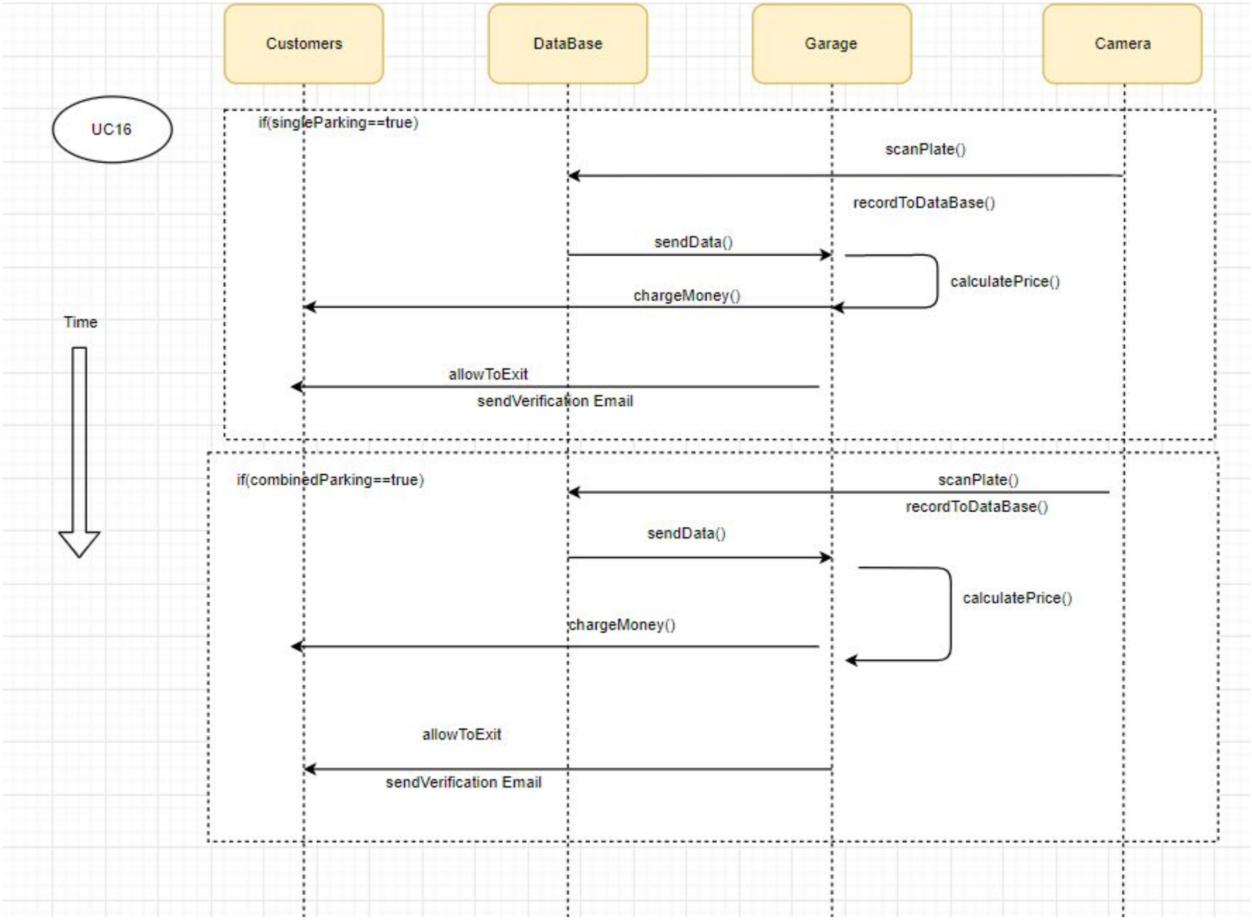
   a) They try to log in and access to the website
   b) They will enter the type of vehicle (size of vehicle)
   c) Database will match the type of vehicle and return the map related to it to the website
   d) The website will display the map with all slots including available and unavailable
   e) They 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
1
Access Website and Log In
If new reservation
Enter type of vehicles (Size)
MatchTypeVehicle()
findMap()
Return type of vehicle
Return map
Display all slots in a map related to type
else (extent time)
Check available parking lots
AvailableLotCheck()
Return available parking lots for the day
Display available parking lots for the day
Check available times
TimeCheck()
If extent or
change time
Return available times for the day
Enter reservation info (including lot and times)
CreateReservation()
updateAvailability()
Return total prices
confirmReservation()
1
1
Display Confirmation with infomation
else (confirm)
Confirm Reservation
Save Reservation
Send Confirmation Email

1.5 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.

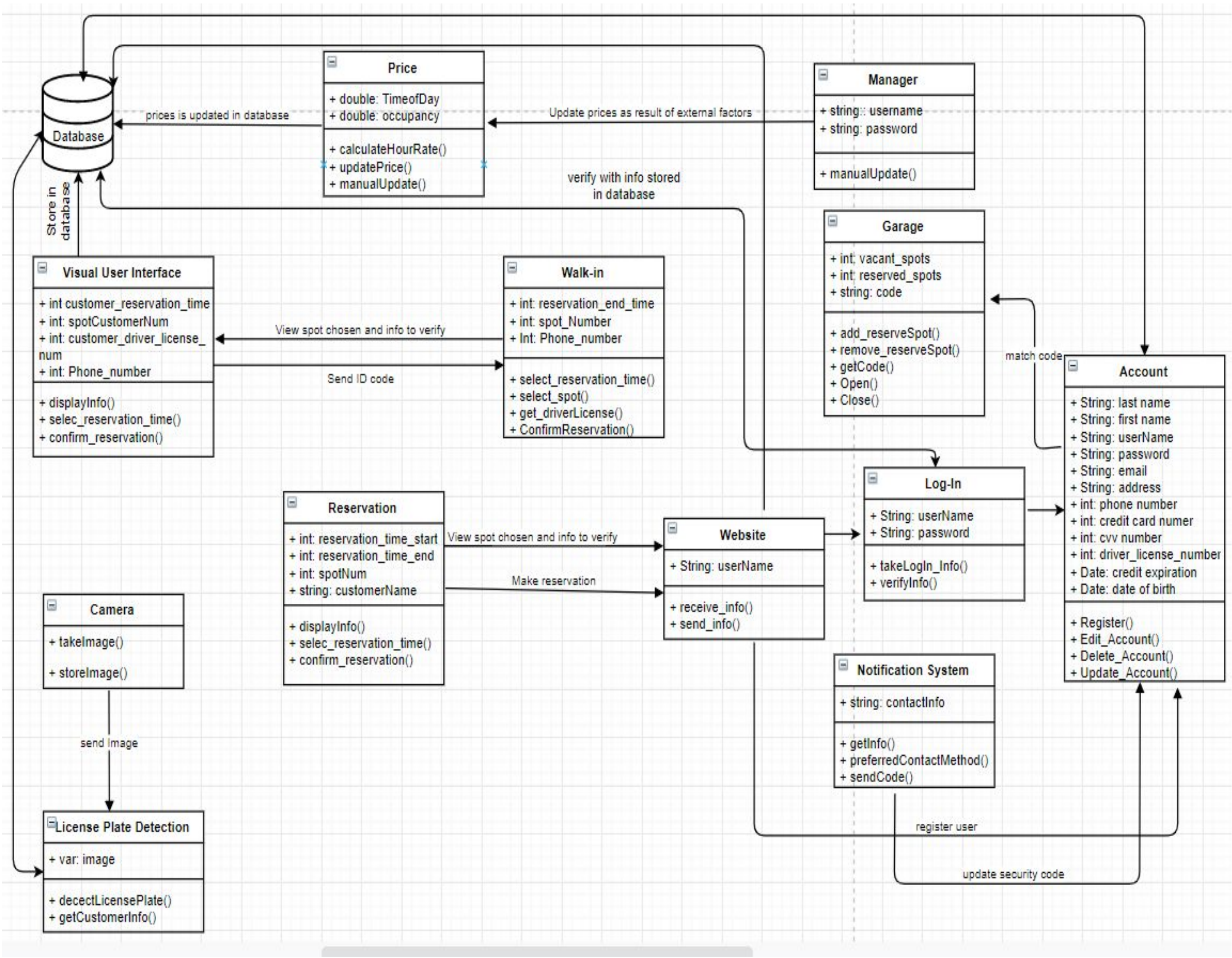
   d) System will base on arriving and leaving time of that vehicle to calculate price.

   e)Customers do payments

After leaving garage, customers will get confirmation email.

2. Class Diagram and Interface Specification:

2.a Class Diagram:



2.b 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.

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()

3. System Architecture and System Design

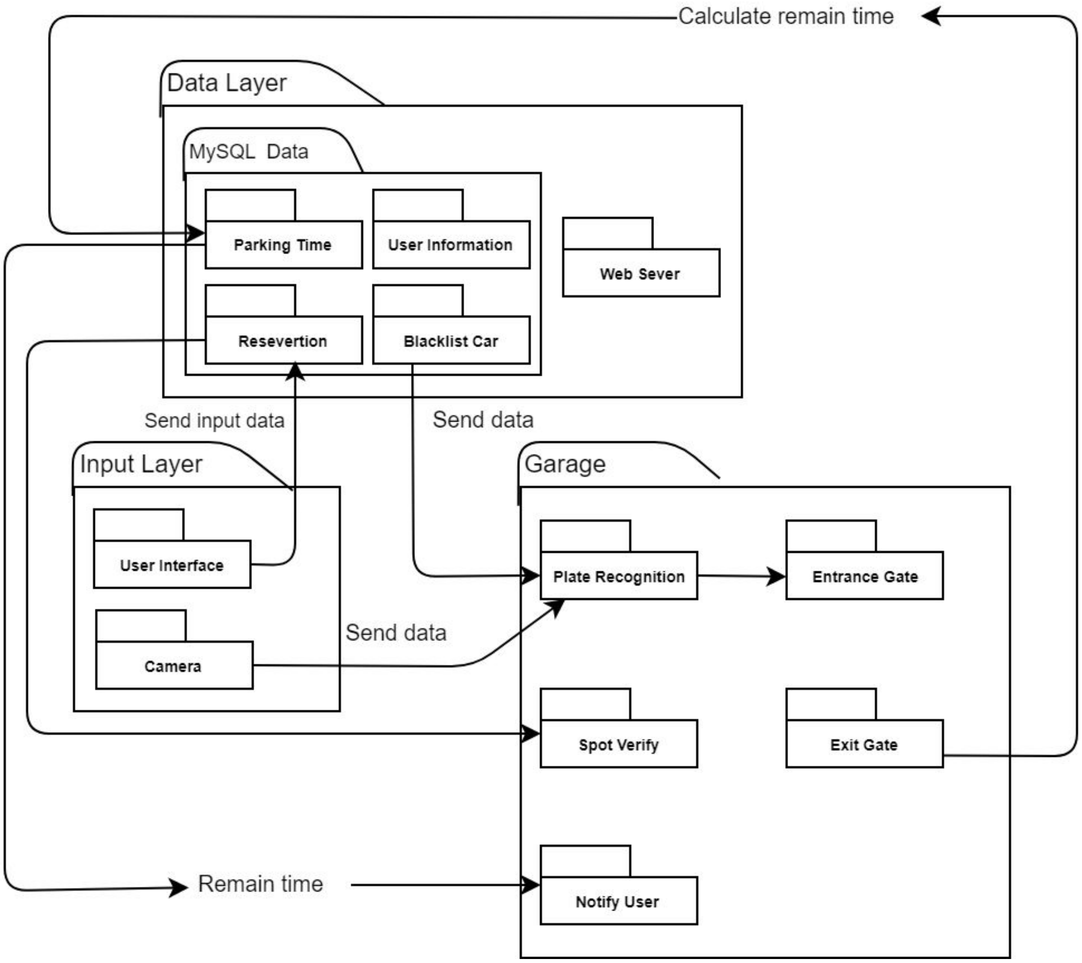
3.a Architectural Styles

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.

3.b 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.

3.c. 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.

3.d Persistent Data Storage

Most data collected by the system will be stored in a mySQL database while some 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.

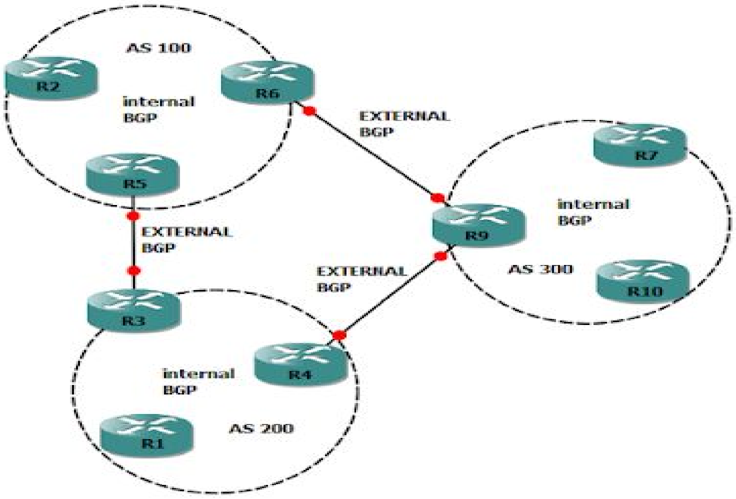
Such a requirement can be satisfied using a blockchain. We can store key data such as transactions, reservation IDs, and lot visit history in a permanent, immutable manner. This is possible because the blockchain has a feature built-in that makes it so that no data stored on it can be overwritten, ever. 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.

3.e <u>Network Protocol:</u>

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.



3.f Global Control Flow

Time Dependency

There is time dependency in our system. Our 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. Our 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 our 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 our 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.

3.g 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)

4. Algorithms and Data structures

4.1. Algorithms

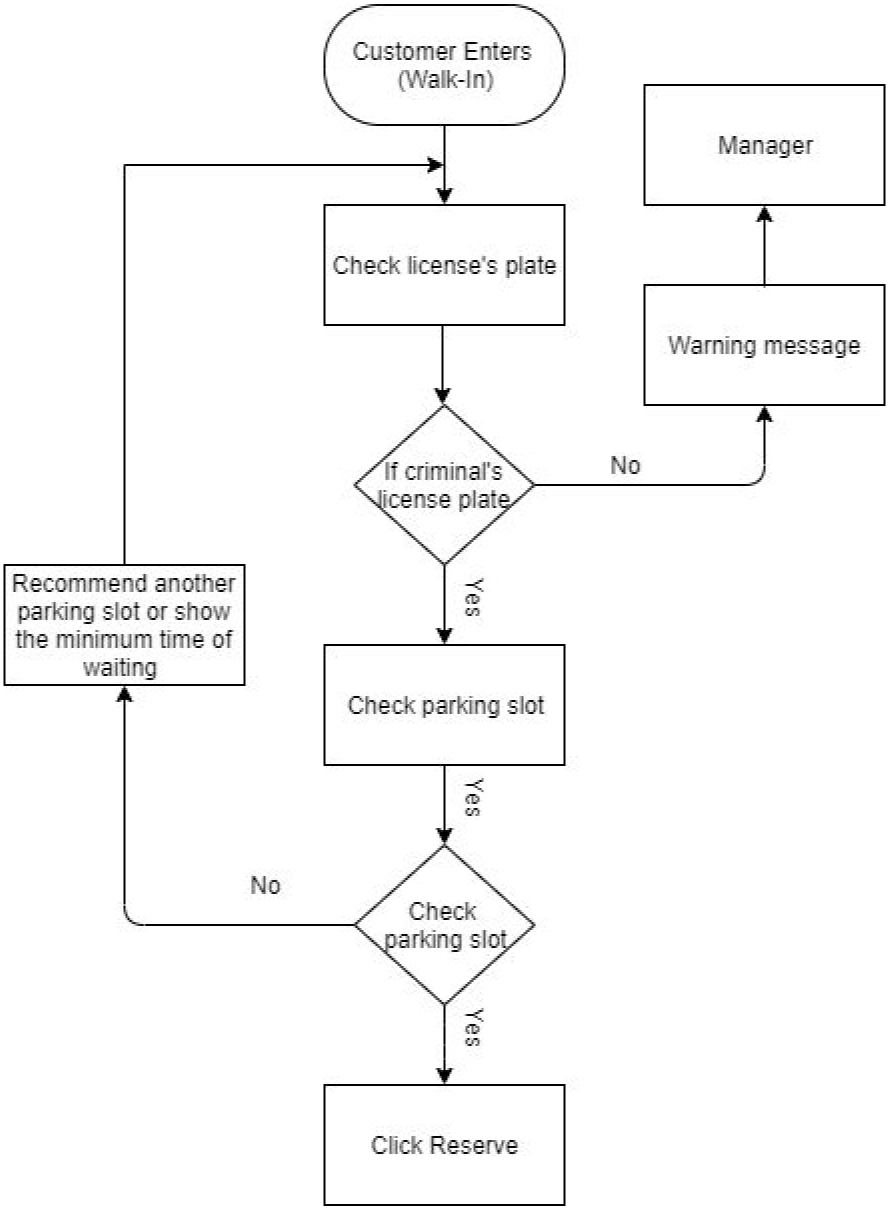
4.1.a Garage

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.

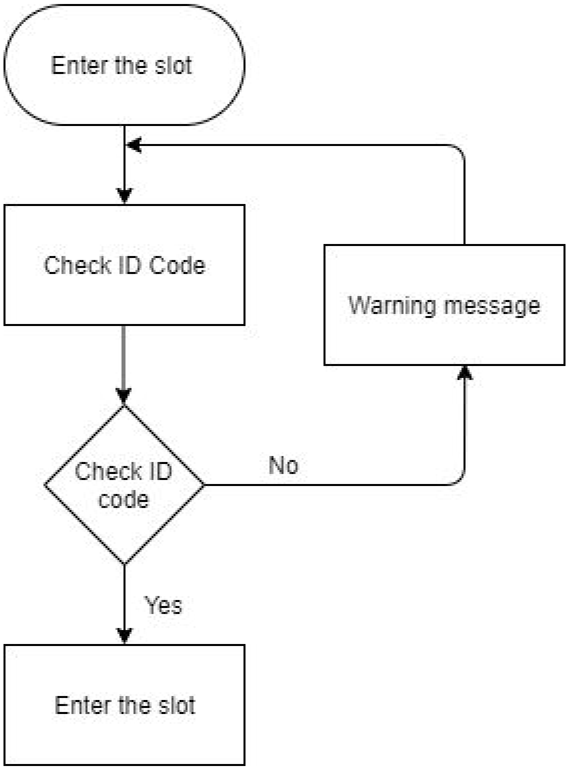
There will be an algorithm to arrange the slot in the garage. Each slot of the parking lot will be assigned as an integer item of the array and each of the items in the array will be filled when a car enters and will be removed when that car exits. However, the three critical for the system would be the beginning time, ending time and the 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 code is fit with this slot or not.

Another important algorithm is to check whether our parking slot has slots or not, if does, we let the customer choose, 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 criminal's license plate
No
Warning message
Manager
Yes
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**



4.2.Data Structures

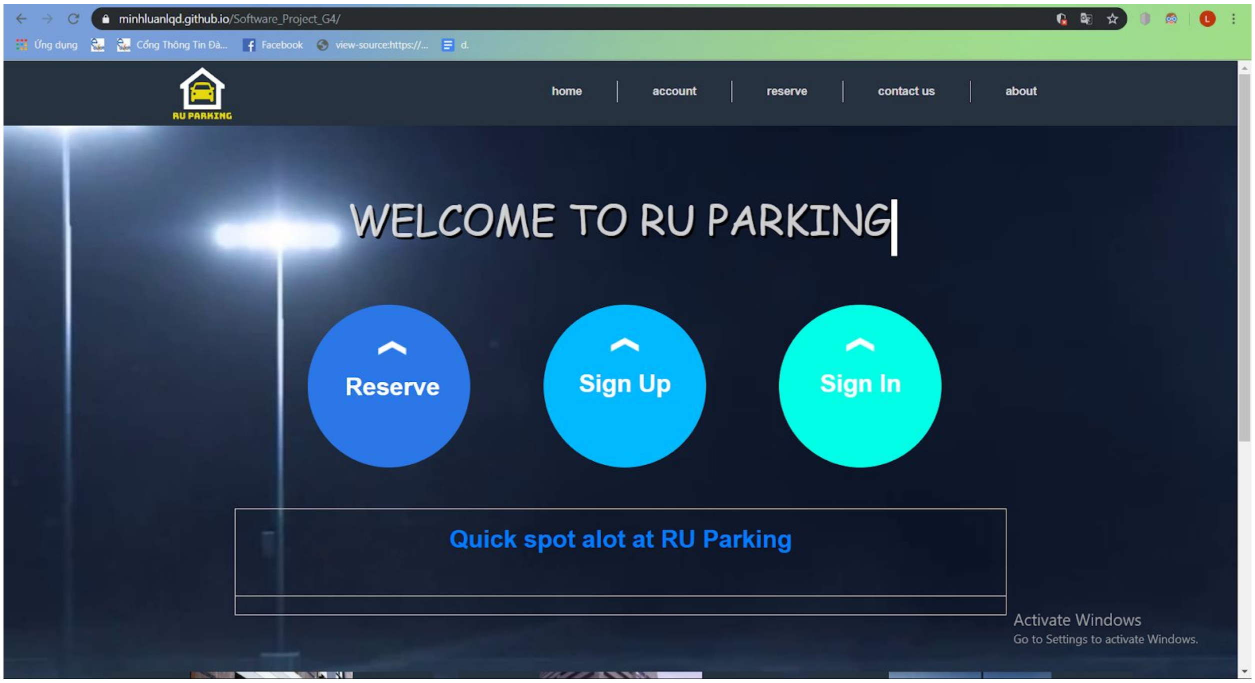
Our system will indirectly use many data structures such as hash tables, linked lists and bitmaps. Since we are storing huge amounts of data, data structures are a must if we want to achieve high performance and efficiency.

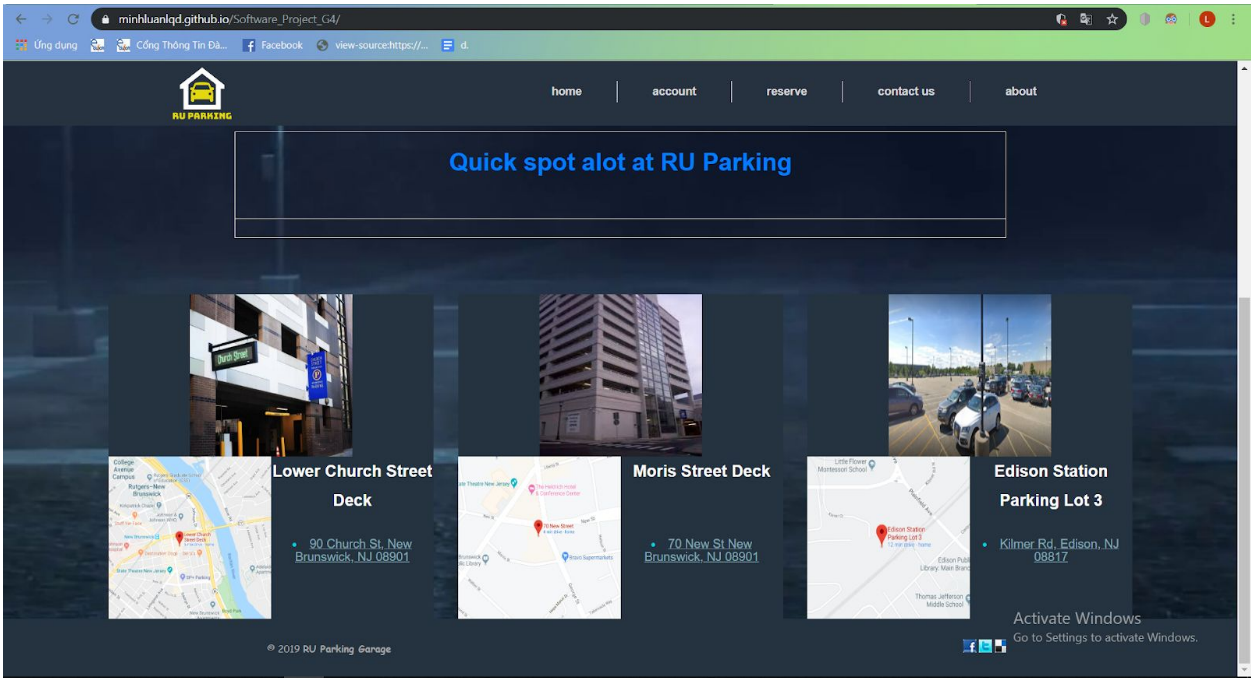
Blockchains use something called a “hash pointer” combined with linked lists. A hash pointer is used as the “previous” pointer in a linked list. The linked list in a blockchain is not like a traditional linked list, where a block holds the data element and a “next” pointer. A blockchain linked list still 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 information such as slot availability, reservation times, current price, slot size, time remaining, and much more.

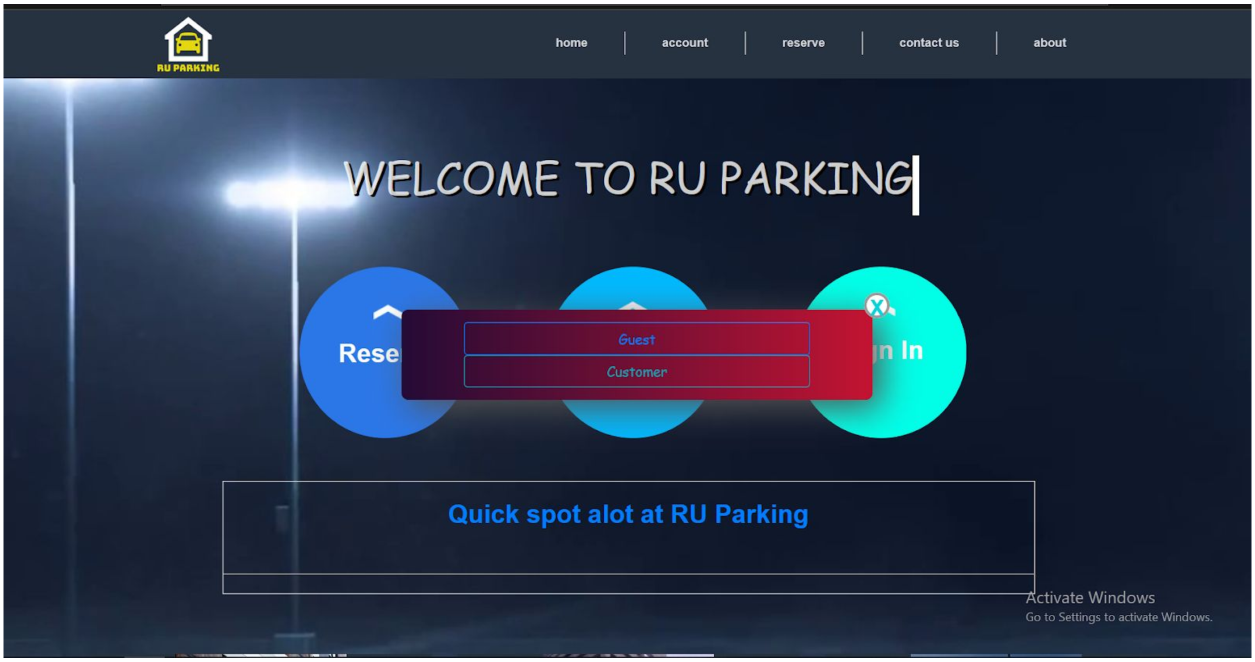
5. User Interface and Implementation

5.1. Home page:


minhluanlqd.github.io/Software_Project_G4/
RU PARKING
home
account
reserve
contact us
about
WELCOME TO RU PARKING
Reserve
Sign Up
Sign In
Quick spot alot at RU Parking
Activate Windows
Go to Settings to activate Windows.

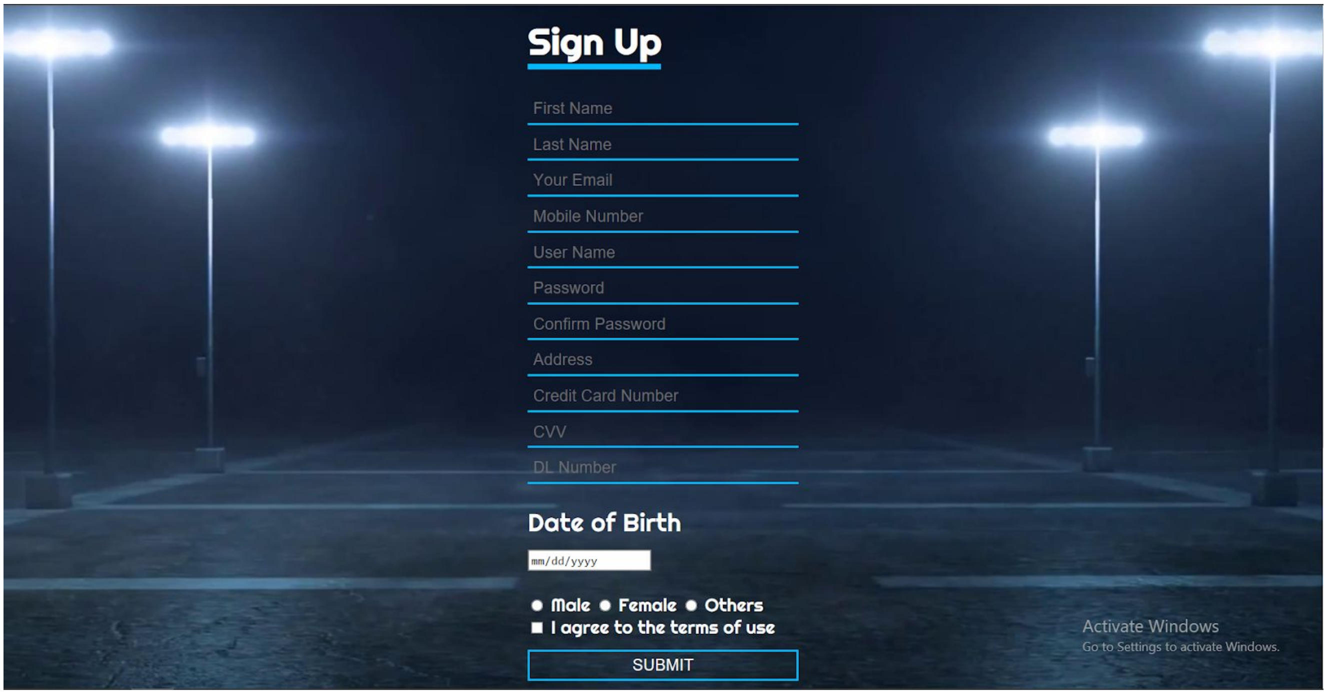

minhluanlqd.github.io/Software_Project_G4/
Ứng dụng
Cổng Thông Tin Đà...
Facebook
view-source:https://...
d.
RU PARKING
home
account
reserve
contact us
about
Quick spot alot at RU Parking
Lower Church Street Deck
90 Church St, New Brunswick, NJ 08901
Moris Street Deck
70 New St New Brunswick, NJ 08901
Edison Station Parking Lot 3
Kilmer Rd, Edison, NJ 08817
© 2019 RU Parking Garage
Activate Windows
Go to Settings to activate Windows.

- 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:



- 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**)

5.2 Sign Up Page:



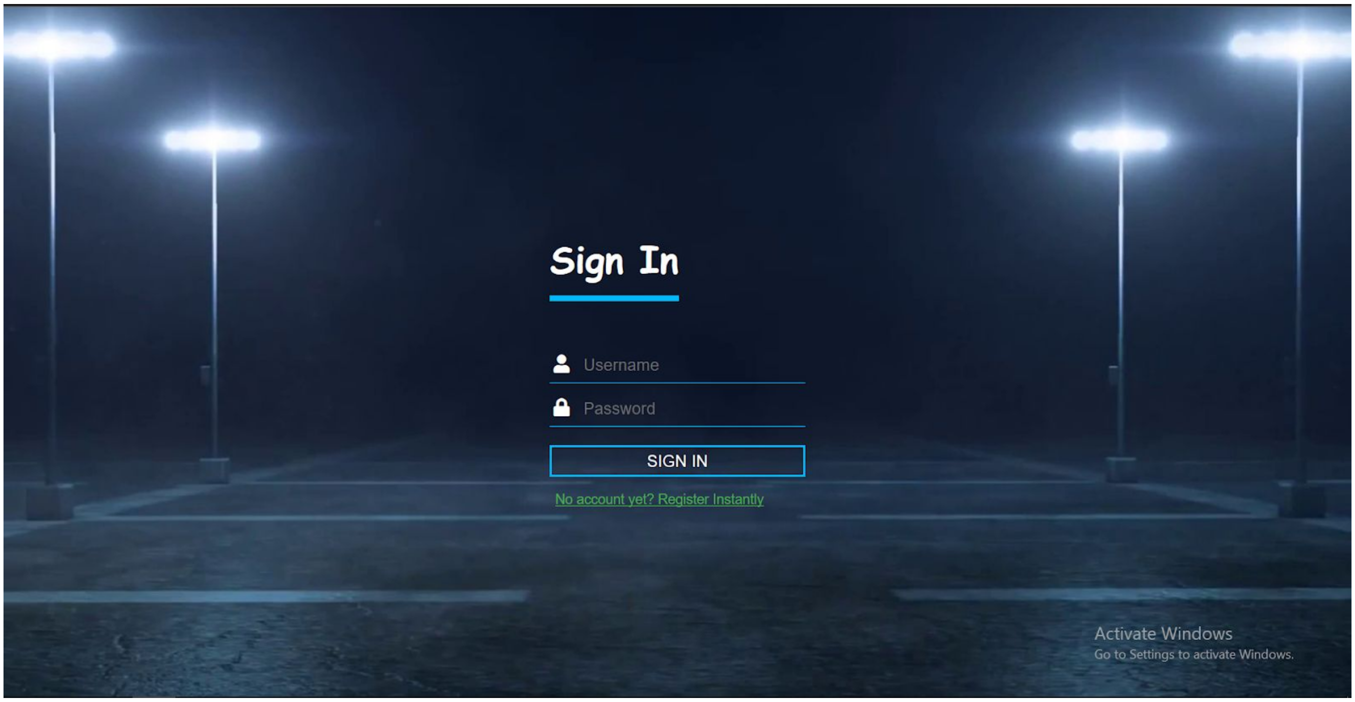
Data Entry: Total 1 click on Sign Up and 14 keystrokes as follow:

- 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.

5.3 Sign In Page:

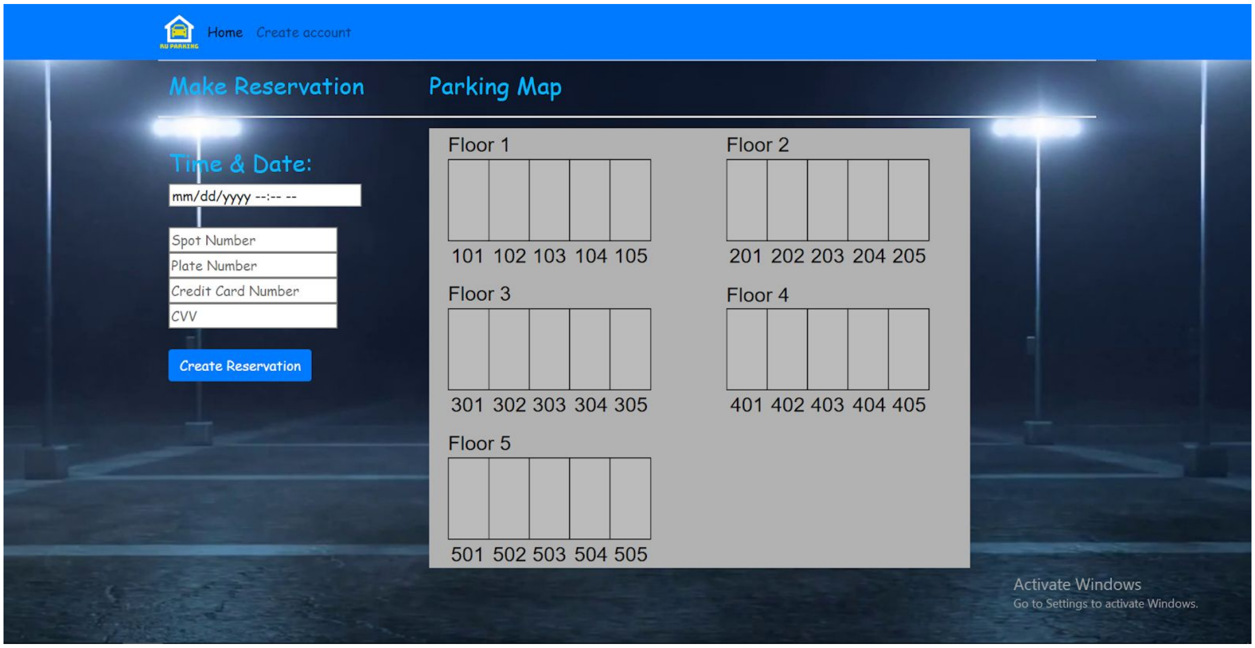

Sign In
Username
Password
SIGN IN
No account yet? Register Instantly
Activate Windows
Go to Settings to activate Windows.

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.

5.4 Walk-in page:



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

- Time and Date
- Spot Number
- 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.

6. Design of Test

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 hackers.

Test cases:

- Implement
  - To check if the Blockchain can be applied successfully in personal information protection.

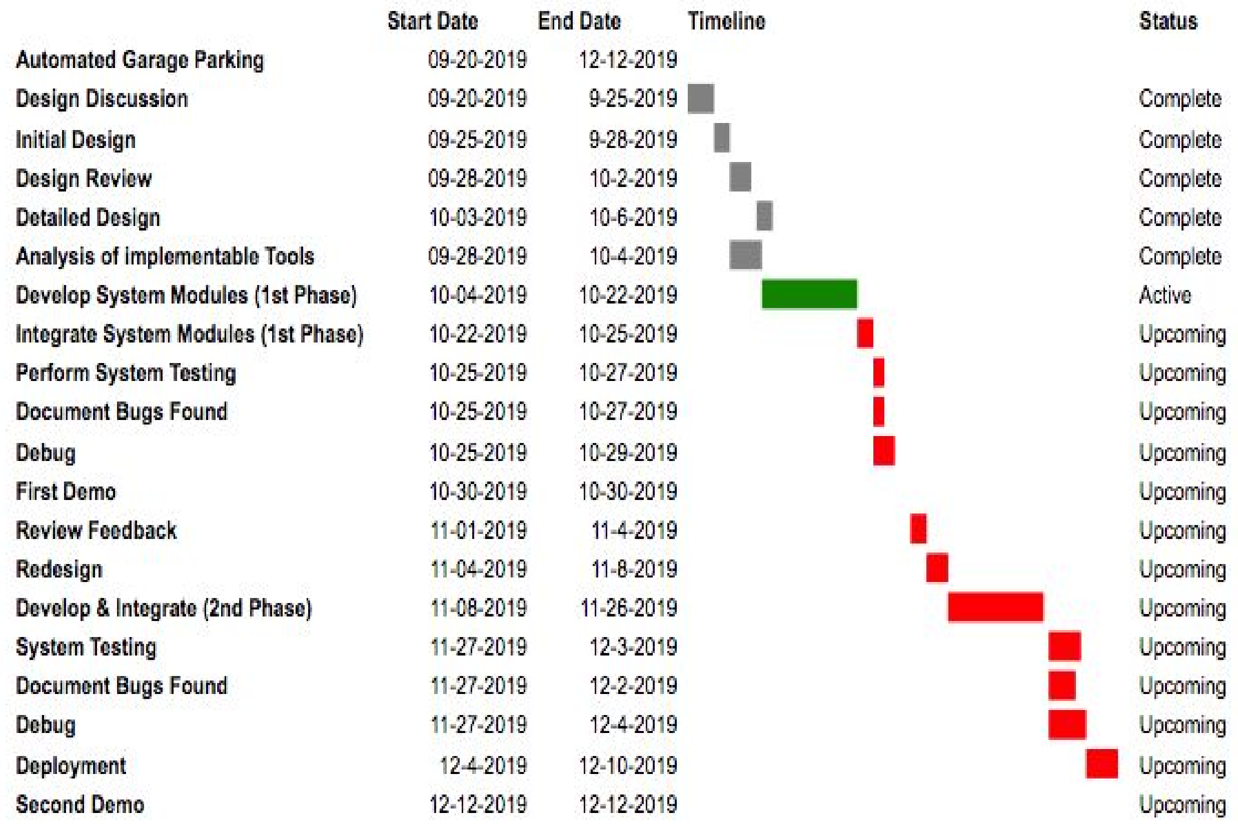
7). Project Management & Plan of Work

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

1. Gantt Chart

Task Phase Description



	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		Active
Integrate System Modules (1st Phase)	10-22-2019	10-25-2019		Upcoming
Perform System Testing	10-25-2019	10-27-2019		Upcoming
Document Bugs Found	10-25-2019	10-27-2019		Upcoming
Debug	10-25-2019	10-29-2019		Upcoming
First Demo	10-30-2019	10-30-2019		Upcoming
Review Feedback	11-01-2019	11-4-2019		Upcoming
Redesign	11-04-2019	11-8-2019		Upcoming
Develop & Integrate (2nd Phase)	11-08-2019	11-26-2019		Upcoming
System Testing	11-27-2019	12-3-2019		Upcoming
Document Bugs Found	11-27-2019	12-2-2019		Upcoming
Debug	11-27-2019	12-4-2019		Upcoming
Deployment	12-4-2019	12-10-2019		Upcoming
Second Demo	12-12-2019	12-12-2019		Upcoming

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

3. Product Ownership

Member	Done	In Progress	Future Responsibility
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 & Nainil	Doing research about database, back-end technology.	Building a Database using SQLite 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 ledgers of all transactions.

8. References:

1. Blacklisted car webpage: https://www.stolencar.com/Report/Search

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/