Team Full Circle

Phuc (Billy) Huynh (1775946)
phuch@uw.edu

Aaron Handjojo (1875857)
aarhan7@uw.edu

Pratit Vithalani (1514123)
pratitv1@uw.edu

Sesario Imanputra (1874080)
sesari@uw.edu

Project Proposal

Our revolutionary idea is to combine the future of driving with the current demand for cheaper taxis. We will utilize the ever-expanding technology of self-driving vehicles and bring it to the existing rideshare service customers. Our app will allow owners of self-driving vehicles to lend their car to the users so that they can get to where they want to go. Our main goal is to use the power of self-driving cars to give a better and cheaper option to users of other rideshare apps.

Vision

To create the world's most compelling rideshare service that combines the ever-expanding technology of self-driving vehicles.

Mission

To bring smarter and safer transportation to everyone with the lowest possible prices.

Goals and Objectives

Goal 1: To maximize the potential of self-driving cars in ridesharing.

Objectives:

- Create a rideshare app specifically made for self-driving cars with functionality only available for said vehicles .
- Work closely with self-driving car manufacturers to influence the addition of required app functionalities (i.e. internal camera, QR code, etc.).
- Enable owners to command their car through the app to go refuel to ensure owners can get their car back fully refueled.
- Allow the cars to automatically receive requests without owner oversight so the owner does not have to.

Goal 2: To provide a cheap and reliable rideshare service for users.

Objectives:

- Eliminate driver service fees to bring the overall cost of using the app down.
- Incentivize 5000 self-driving car owners total to enroll their vehicle into this project for availability in the first month of the app's lunch.
- Maintain a service uptime of at least 99.99%.

Goal 3: To provide a secure service for self-driving car owners to share their car

Objectives:

- Encrypt user and owner financial details when storing and processing payments.
- Create a reporting system in the app for owners to report disruptive users so that damages would be prevented or fined.
- Provide a QR code to the customer to allow them to enter the car to ensure that the right customer comes into the right car.

Stakeholders

Car Owners (Primary Stakeholders)

Major interest in our services since they are one of our main audience.

Customer/Passenger (Primary Stakeholders)

Major interest in our services since they are one of our audience.

Car Manufacturers (Secondary Stakeholders)

The services that we will provide will depend on eligible autonomous cars supplied by car manufacturers.

Shareholder

While they may not be an active member of the business, they will still be very interested on how our project performs.

Local Communities

The community will be heavily affected by our service, especially local, existing rideshare services. Thus, we have to tread carefully on how low we decide to set our prices.

Business Analyst

An organizational role that is also a stakeholder is a business analyst. They compile requirements from stakeholders and rank the importance of features. This is important because requests will continue to evolve as autonomous cars are still evolving in the automotive industry. Thus, if any mishaps occur with driverless cars, the organization

would be informed by the analyst.

Designer

An organizational role that is also a stakeholder because the designers implement the analysis gathered into our service. Furthermore, designer acquires an important role in the safety aspect of our service, such as when the doors of the car locks relative to when the passengers enter the car and keeping sensitive information secure.

Regulatory Agency

Government agencies will be interested on how the service is run. We suspect government agencies will be a major stakeholder due to Uber's greyball debacle, in which they denied service to specific customers through mining credit cards and geofencing to avoid interacting with law enforcement agents. Thus, I would assume they will have high interest on how our day to day operations works.

Non-Functional Requirements

Security

Since we will be allowing owners and customers to receive and give payments, our system must remain secure in order to ensure the safeguard of such sensitive information. Also, we must be able to keep the information of each vehicle secure, including its VIN and current location.

Usability

The purpose of this app is to give users the fastest experience in getting around or setting up a vehicle for service. For this, the app should provide a simple and easy-to-use interface.

Availability

Due to the nature of the service we are providing, a transportation product, the system must stay up for at least 99.99% of the time. This will eliminate dropped orders and will keep everyone up to date on where their cars are at the time being.

Portability

We are aiming to provide an easy to use experience and part of that has to do with cross platform accessibility. That means that someone can start a task on one device

and finish it on another. For example, a customer can book a ride from their laptop and then enter the vehicle after scanning the QR code with their phone.

Scalability

Our vision is to, eventually, have this service in use around the world. For that, we have to design our system in a way that allows us to expand from a couple of cities to cities around the world in different countries.

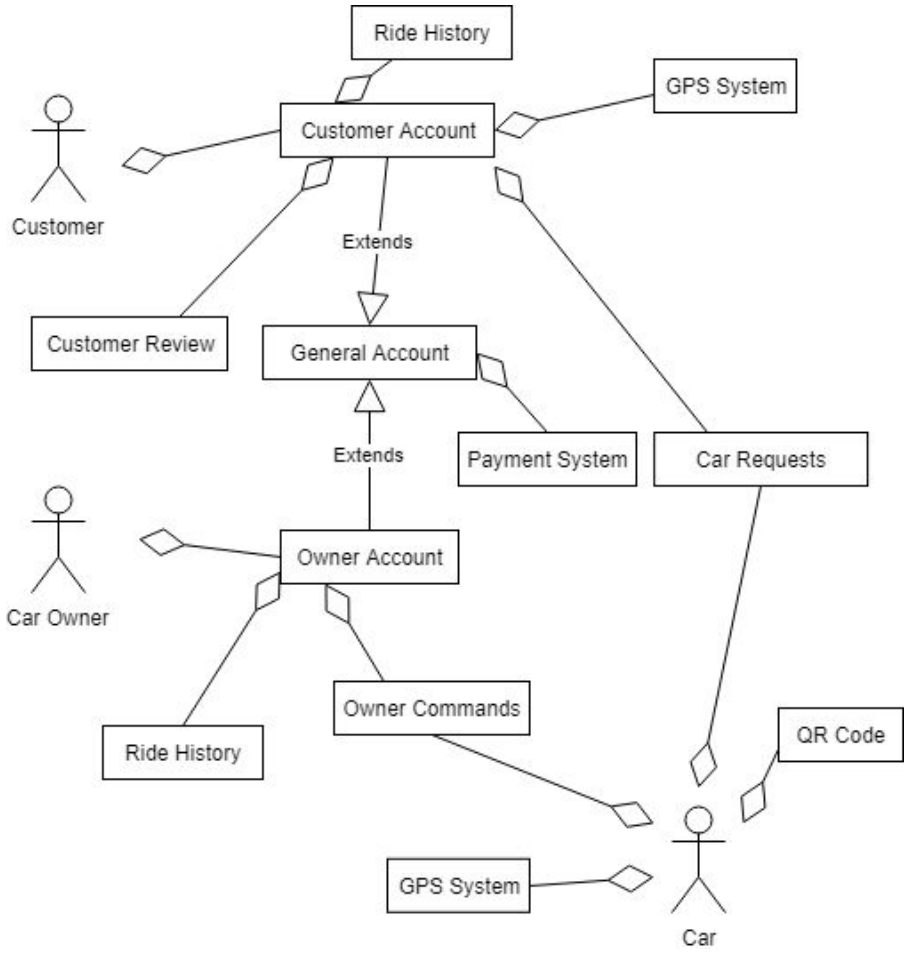
Confidentiality

Because we are looking to become a quick rideshare service, we do not believe that it is necessary for people to know whose car they are getting into. To do this, we keep the identity of the car owner hidden from the customers who use it.

Functional Requirements

- Owners should be able to tell their car to come back to them or pick up/drop off passengers
- Customers should be able to select how many people are with them in order to get a car that can fit all of them
- Owners should have a history of who was in their car
- Customers and owners should be able to pay through this app (debit card, credit card, etc.)
- Owners can report customers for any damage or disruption caused in their vehicle
- Customers must be able to identify themselves to the vehicle before they enter
- Owners should have the option to limit how far their car goes away from them
- Owners should be able to tell their car to refuel/recharge when they are not carrying passengers
- Customers should be able to order a car in advance or on the spot

Domain Model Diagram



When creating this Domain Model Diagram, we considered the fact that this would not be mostly true and decide to spend less than two hours worth of planning to draw this DMD. We focused the model on real world objects that exists within the boundaries of the application we are designing such as a gps system, a rides history, accounts, etc. We also made it as simple as it would look without giving too much details into it, effectively making it a “glossary”.

Use Case Diagram

Use Cases

Authentication

Login

Basic Course

Opening the app prompts the customer to enter their username and password, then click enter. When the customer enters their username and password, the “ENTER” button will be clicked and a loading screen will be displayed. While the loading screen is displayed, the system validates the entered username and password and logs customer into the system.

Alternative Course
If any of the customer’s username or password combination is blank, the customer will be prompted to re-enter their credentials.

Verify Login

Basic Course:
Customer clicks on the ‘LOGIN’ button which then the system verifies the existence of the combination of username and password. After verification, their account information is displayed on the customer interface.

Alternative Course
If the customer’s credentials are invalid, the system returns an invalid input message and prompts the customer to login again through the app

Display Wrong Login

Basic Course
If it is determined that the customer’s login credentials are incorrect, the system replies returns the following warning: “Your username or password is incorrect. Please try again.”

Alternative Course
No message will be displayed if the customer enters valid credentials.

Financial Information

Add Funds to Account

Basic Course

When the customer clicks on the ‘Add Funds’ button in the customer home page, the system prompts the customer to enter their bank name, bank account number, and the amount they want to transfer over. The system validates the bank account with the bank and ensure the customer has enough funds to have their requested amount transferred over. The system will move the funds from the bank account into the app and updates the customer’s account.

Alternative Course

One alternative course is the customer inputs invalid information. This can include an invalid bank, incorrect bank account number, or a negative number of funds. For this, the system will display an error message to the customer, stating where exactly the error came from.

Another alternative course would be if the bank determines that the customer does not have enough funds to complete the transaction. The system will return an error message stating this, and the system will end the process.

Deposit Funds to Bank

Basic Course

When the customer clicks on the ‘Deposit Funds’ button in the customer home page, they will be prompted to enter their bank name, bank account number, and the amount they want to deposit to the bank from their account. The system contacts the bank and validates the bank account detailed. After the account number is validated with the bank, the system will verify that the customer has enough funds in their account to be transferred to the bank. The funds will then be moved from the account into the bank and the customer information will be updated with funds deposited.

Alternative Course

One alternative course is the customer inputs invalid information. This can include an invalid bank, incorrect bank account number, or a negative number of funds. For this, an error message will be displayed to the customer, stating where exactly the error came from.

Another alternative course would be if the app determines that the customer does not have enough funds to complete the transaction. The customer will have an error message stating this, and the process will be dropped.

View Funds

Basic Course
After the customer logs in, customer's funds is displayed on the screen

Alternative Course
There are no alternatives.

Make Payment

Basic Course
When the customer clicks on the 'Make Payment' button in the customer page, the system will prompt the customer to enter the amount they want to pay to the receiver, as well as the receiver's account name, account number, or email. The system determines whether the customer have enough funds in their app to make the payment and that the receiver's account does exist. Then, the system will move the funds from the customer's account to the receiver's account.

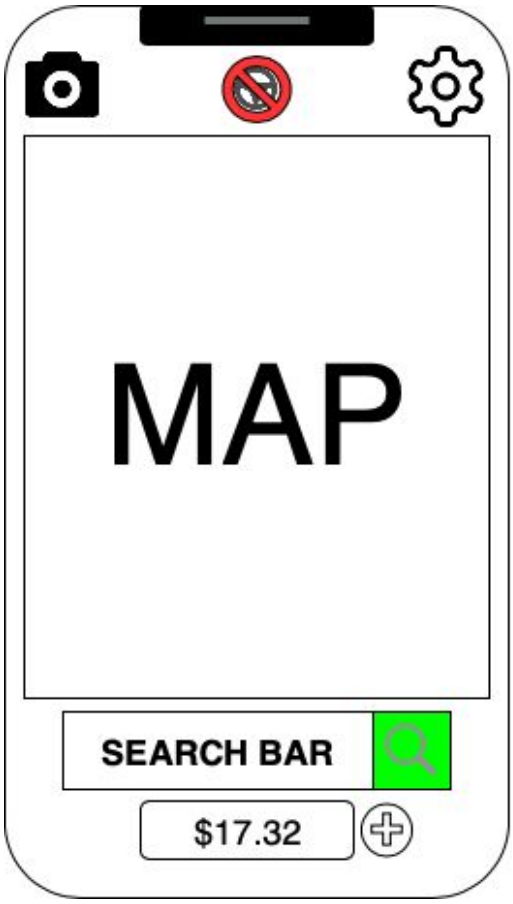
Alternative Course
If the customer does not have enough funds in their account to make the transaction, the system will report the cause to the customer through the app. This allows the customer the option to enter in their bank name and bank account number again to transfer funds. After the account number is validated with the bank, the bank will verify that the person has enough funds to have their requested amount transferred over. The funds will then be moved from the bank account into the app and the customer will be updated.

One other alternative course is the customer inputs invalid information. This can includes an invalid bank, incorrect bank account number, or a negative number of funds. For this, an error message will be displayed to the customer, stating where exactly the error came from.

Another alternative course would be if the bank determines that the customer does not have enough funds to complete the transaction. The system will send out an error message displayed to the customer and then drop the process.

Customer

GUI Prototype:



(This is our initial prototype of the final GUI that may be used in the application)

Request Ride

Basic Course

Customer sets pick up location at desired or current location to request the ride. During this process, the system calculates distance, determines eligibility, and displays it on the interface along with the price for the service. Then, The customer can selects the “CONFIRM” button to initiate service.

Alternative Course
If distance is not eligible, the system would disable the “CONFIRM” button by coloring it gray. In addition, an error message located on top of the button appears which states “Distance is too long”. This prompts the customer to determine another location.

Find The Closest Car

Basic Course
Customer selects “REQUEST” button on the screen which prompts the system to search for vehicles. This is done by collecting all available vehicles within 1 mile raidus and calculates which vehicle’s position is closest to the customer’s pick up point. Once the closest vehicle is found, the system feeds that vehicle with the customer’s pickup point and instructs it to travel.

Alternative Course
In the event in which nearest vehicle is above 1 mile, then the system will identify the nearest vehicle within 2 miles increments until 6 miles. If no available nearest vehicles exist by 6 miles, the service will cancel itself and send out an message to the customer to try again later.

In the event in which the system identifies two vehicles with the same closest distance, the system provides the task to the vehicle with the highest rating.

Cancel Ride

Basic Course
Customer selects the “CANCEL” button before the system finds an available vehicle. The system recognizes that the button is activated before the system finds a vehicle. Thus, the vehicle does not charge the customer for the service.

Cancellation can also occur if the customer fails to enter the vehicle within 2 minutes of the vehicles arrival at the pickup point. Then, the system deducts the customer’s account by the cancellation fee, which is 1.50$.

Alternative Course
Customer selects the “CANCEL” button after a vehicle is chosen by the system. The system recognizes this and deducts the customer’s account by the cancellation fee, which is 1.50$.

Customer's ride is cancelled if the they fail to enter the vehicle within 2 minutes of the vehicles arrival at the pickup point. Then, the system deducts the customer's account by the cancellation fee, which is 1.50$.

Authenticate the Vehicle

Basic Course

Customer is assigned a vehicle. The system provides vehicle information, including license plate, make, and model. Customer finds a car that fits the description by the make and mode. Customer further confirms this by viewing the license plate. Then, the customer approaches the car and scans the QR code on the car. The system obtains the customer's image and compares it to the vehicles QR code to verify. In return, the system unlocks the vehicle for the customer to enter.

Alternative Course

System compares vehicles QR code to verify but it fails. The system notifies the customer and the customer can resend the QR code to refresh the QR code loaded.

File Complaint

Basic Course

When a customer is dissatisfied with the car service, they can file a complaint by selecting on "RATE THE RIDE" button after the ride. After selecting the button, the system will prompt the customer to leave a rating described with stars from 1 to 5 along with a comment in the comment box. To submit, the customer can select the "SUBMIT" button.

Alternative Course:

If the customer wishes a direct response to their complaint, the customer can call the organization, in which they will be directed to a representative. Representative logs customer's location in the system for reference.

Registering as Car Owner

Basic Course

Customer selects "CAR OWNER REGISTRATION" from the interface. This prompts the system to request the customer's driver license and information regarding their vehicle, such as the make, model, available technologies, and recent maintenance report. System evaluates

information and approves.

Alternative Course
Customer goes through the process of registering themselves along with a car but information does not meet the standard of the system. As a result, the system returns a report detailing the factors which made customer ineligible as car owner.

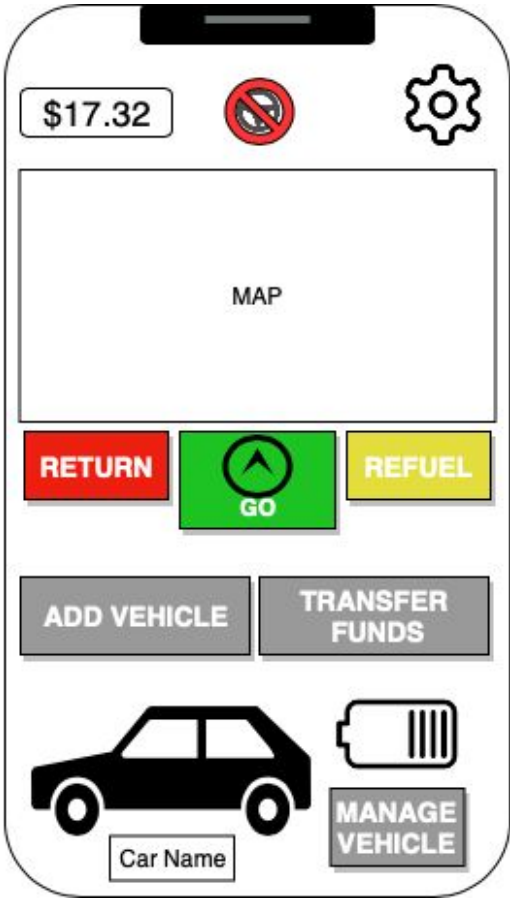
Switch to Car Owner Mode

Basic Course
Customer selects “CAR OWNER” from the interface. System displays a list of vehicles the customer owns. Customer chooses a vehicle, in which the system will bring the customer to the car owner interface.

Alternative Course:
Customer selects “CAR OWNER” but is not registered as car owner. System returns an error which states that customer is not a car owner.

Owner

GUI Prototype:



(This is our initial prototype of the final GUI that may be used in the application)

View Customer History

Basic Course

Car owner selects “HISTORY” in the drop down menu on the screen. The system displays date of ride, time of ride requested, ride cost, and used vehicle information. Customer views the history displayed by the system and it is sorted based on the most recent dates of rides.

Alternative Course

After selecting “History” in the dropdown Menu, the system recognizes that the customer has not initiated any service. Thus, the system displays nothing on the interface.

Add Vehicle

Basic Course

Car owner selects “ADD VEHICLE” button on the owner page. System prompts the customer a list of questions regarding the cars make, model, color, fuel type, front and rear facing cameras, license plate, and a recent maintenance report. System evaluates customer information for service eligibility. The system accepts vehicle information as eligible and updates the customer’s list of available vehicles.

Alternative Course
Car owner selects “ADD VEHICLE” button on the owner page. System prompts the customer a list of questions regarding the cars make, model, color, fuel type, front and rear facing cameras, license plate, and a recent maintenance report. System evaluates customer information for service eligibility. The system denies vehicle information as eligible and notifies the customer on the factors of ineligibility.

Report Customer

Basic Course
If the owner clicks on the ‘Report Customer’ button in the owner page, the system prompts the owner to select which customer is to be reported from the ride history. After clicking on the customer, the system will prompt the customer to enter in the reason they believe the customer should be reported. The system will record the report and then send the report to evaluation team.

Alternative Course
If the owner clicks on the ‘Report Customer’ button in the owner page. The system prompts the owner to select which customer to be reported. The owner changes their mind and leaves the prompts, cancelling the process.

Modify Vehicle Settings

Basic Course
The owner selects “SETTINGS” in the interface. System presents list of options the customer can modify, which includes set distance the car is allowed to travel from their home location, maximum mileage per day, maximum rides per day, and end service time. Car owner modifies their settings and select “SAVE” to save their configuration.

Alternative Course
Car owner sets vehicle settings but did not select “SAVE”. Thus, their configurations are not modified.

Command Vehicle to Go

Basic Course

Car Owners selects “GO” through the interface. This instructs the system to send the vehicle pickup and destination location for the nearest customer in the area. Once a pair of location is selected, it instructs the vehicle to go to the pick up location.

Alternative

Car Owners selects “GO” through the interface. This instructs the system to send the vehicle pickup and destination location for the nearest customer in the area. The system fails to find customer in the area and notifies the customer. Until the system finds a customer, the vehicle stays idle.

Command Vehicle to Return

Basic Course

When the owner wants to take their car off the service and get their car back to the location that they want. The owner login to their account and check whether their cars are in use or not. If not, then they click on “Menu” and choose “service mode” and switch it to off. After that, the system will prompt the owner to enter the location that they want their car to be. The vehicle will return to the requested location.

Alternative Course

One alternative course is the cars keep on accepting the customer’s ride request After the owner switch the service mode to off. Another one is the cars don’t respond to the owners or go to the wrong location after the owner’s request

Command Vehicle to Refuel

Basic Course:

When the car hits the low battery warning dash light, it will send the customers a notification. After receiving a notification, the owner will click on the “Charging station” in the Menu, then the system will display all charging stations that is close by the car’s location. the owner will click on which station they want their car to go to charge their car. When the car is in the charging mode or on the way to the charging station, the service mode will automatically switch to off and switch back on when the car is fully charged.

Alternative Course:

The car doesn't send out a notification to the owner when it hit the low battery warning dash light. The car might die in the middle of the road, It's even more dangerous while having customers in the car.

Another alternative course is the car is still on the service mode and accept the ride, while charging.

Change Vehicle

Basic Course:
Car Owner selects "CHANGE VEHICLE" from interface. System displays list of vehicles the customer owns through the interface. Car Owner selects a different car. System recognizes customer has selected a different car. Thus, system reloads car owner interface with the vehicle selected.

Alternative Course:
Car Owner wishes to change vehicle but changes intentions during the process as the drop down menu appears. As a result, customer selects "CANCEL". Thus, the current vehicle selected is not changed.

Remove Vehicle

Basic Course:
Car Owner selects a vehicle and selects "REMOVE VEHICLE" from interface. System recognizes command and asks car owner to be sure. Car Owner agrees and the system deletes vehicle from car owner's information and interface.

Alternative Course:
Car Owner selects "REMOVE VEHICLE" but regrets during the process. When system asks car owner to be sure, car owner can disagree. Thus, vehicle is not deleted.

Switch to Customer Mode

Basic Course:
Car Owner selects "CUSTOMER" through the interface. System recognizes commands and switches interface into customer mode.

Alternative Course:
There is no alternative to this route since you can't be a car owner without being a customer.

Vehicle

Verify Empty Vehicle

Basic Course

Since a customer needs to scan to enter the vehicle with a QR code, the same logic applies with exiting the QR code. Thus, a vehicle can only be empty if it is scanned twice by a customer.

Alternative Course

Car owner can further verify if a vehicle is empty by viewing a live feed of the vehicle's interior. This can be accessed by the car owner at any point during service.

Validate QR Scan

Basic Course

System obtains vehicle's assigned QR code and compares it with provided customer QR code for eligibility. System recognizes that they are identical and unlocks the vehicle's car.

Alternative Course

System compares QR code of vehicle and customer but fails to verify. The vehicle's car does not unlock until the system feeds the vehicle with the correct QR code.

Find The Closest Request

Basic Course

After a service is terminated and the vehicle is free to leave, the vehicle pulls requests from the system. At first, it starts off with requests in a 1 mile radius, and doubles in size every time there is no request.
The system feeds the vehicles with the nearest available pickup point in 1 mile radius. If no existing customers exist within the 1 mile radius, the system expands its searching radius by 2 miles until 6 miles is reached.

Alternative Course

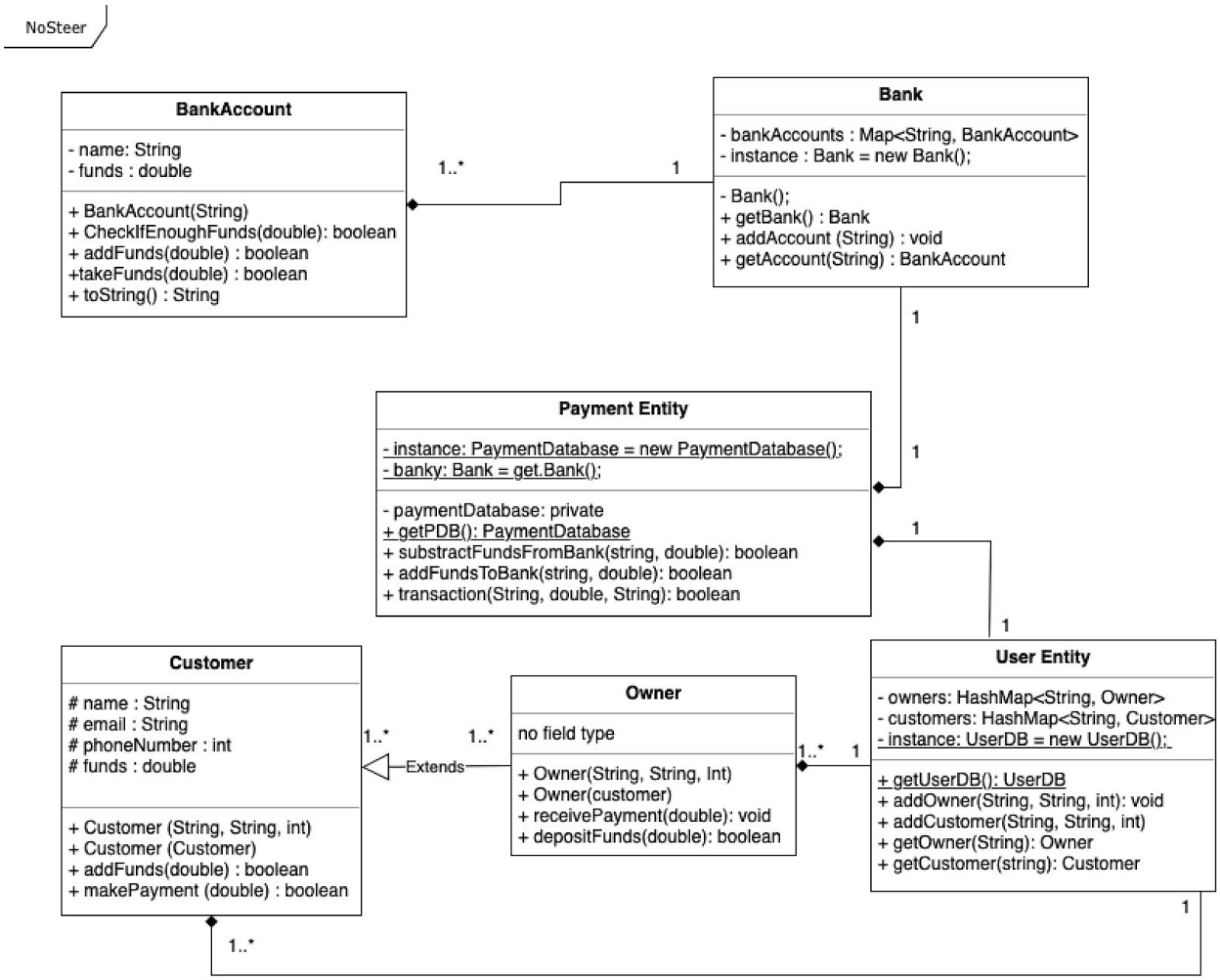
If the system fails to provide the vehicle the closest request, it will notify the car owner and suggest if they want to set a destination for the car in hopes for a higher chance of finding a customer.

Use Case Summary

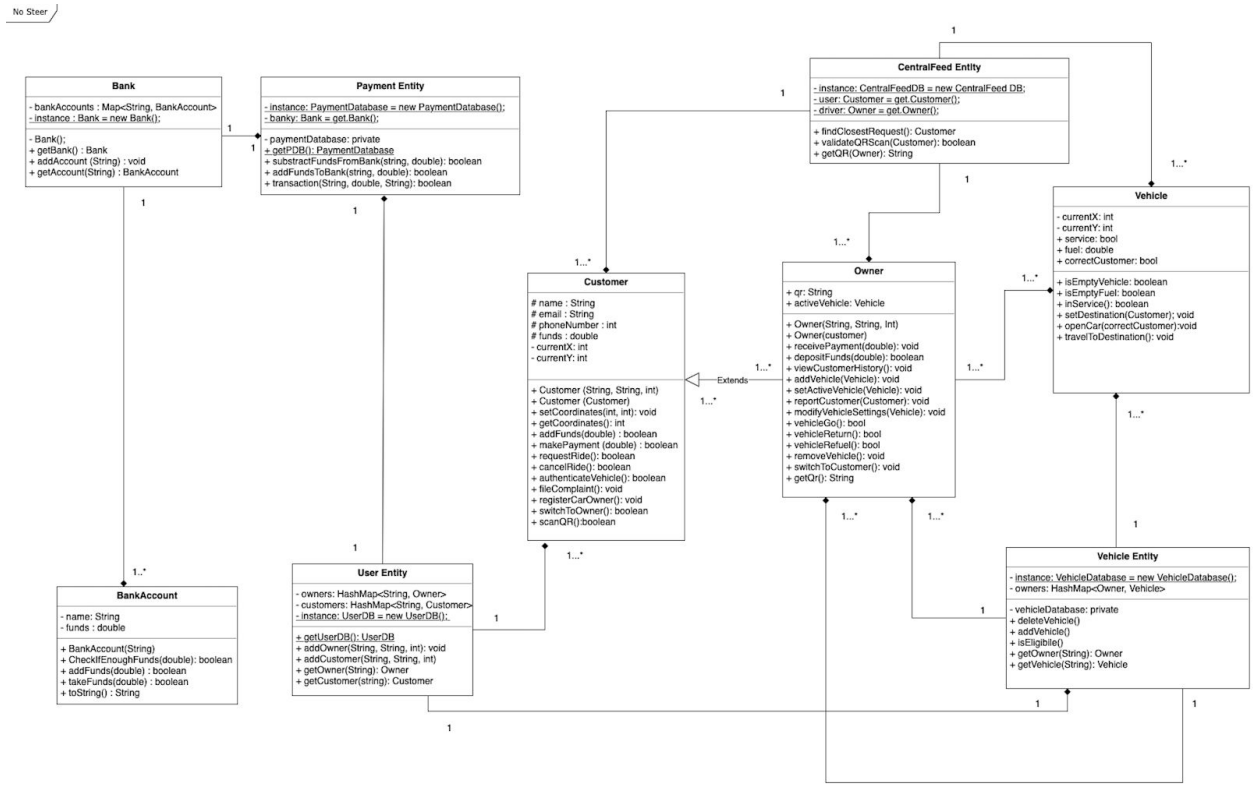
In each of the use cases, we utilized the 2 paragraph method to establish the sunny and rainy day (good and bad) processes that can occur. All our actors were external from the system: customer, owner, vehicle, and bank. Our use cases also use an event-flow system where an actor does something to accomplish a goal (non-verb-noun structure). This allowed us to use active voice in all of our cases, showing immediate action. Also, using the noun-verb-noun structure forced us to use the domain and boundary terminology.

UML Diagrams

Implemented for Workshop 3



Overall Diagram



Relevant Domain Objects and Use Cases

Domain Objects:

- Customer:
    - The app's main entities include customer as they will be the primary users of the app and heavily influences how our app should work. Because of this, we believe that customers are relevant to the ER diagram and needs a data storage.
- Owner:
    - Just like customers, owners are also important in our app as they would be the ones to own cars. As we will be relying on the relationship between the customer, owner, and vehicle for our app, we would need data storage for owners too.
- Account:
    - The account is how the customers will be able to access the app and connect to the vehicle. It is also how they will be able to communicate with the bank for funds. We need a data storage for this entity because we are expecting a 1 to 1 relationship between customers and this entity.
- Vehicle:
    - Since an owner may have multiple vehicles, we need a separate data storage table for the vehicles, as well.
- Ride History:
    - We believe that the ride history is going to be an important part of the security factors of our app. Since the central function of our app is to provide rides for customers, it is important to have a history database to record all of the rides the customer and the vehicles have done.

Use Cases:

- Login
- View Funds
- Add Vehicle
- View Charge Level
- Create Owner Account
- Find Closest Vehicle
- Add Funds from Bank
- Deposit Funds to Bank

Business Rules

Customer:

- Customers are identified through their customer ID
- A customer has one account
- A customer's location should be recorded

Owner:

- All owners are customers but not all customers are owners
- An owner can have more than one vehicle

Vehicle:

- Vehicles are dependent on their owner
- A vehicle's charge percentage should be recorded
- A vehicle's location should be recorded

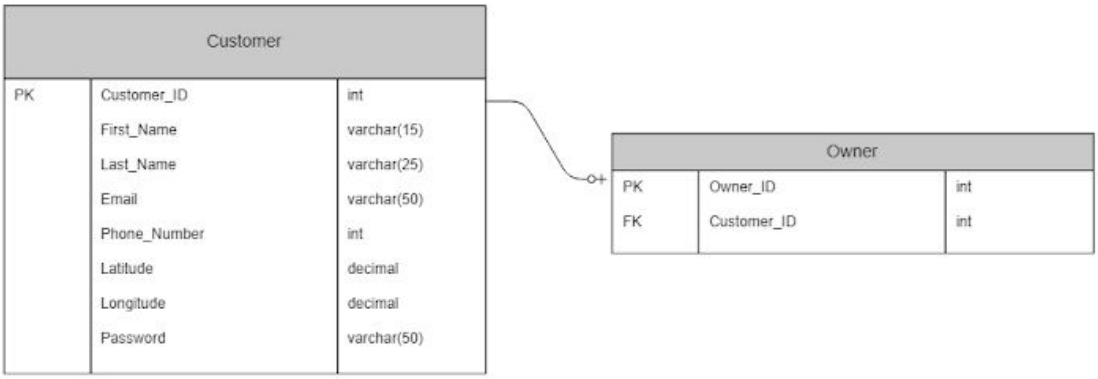
Ride History:

- Ride histories are identified by ride ID
- Ride histories are dependent on the vehicle used and the customer

Sample Business Cases

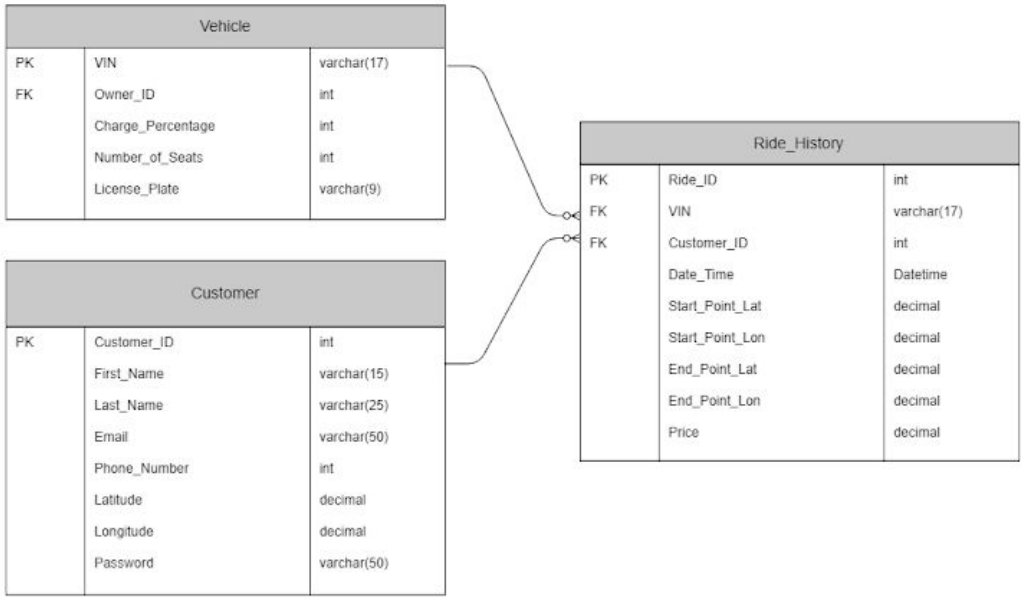
Customers should be able to become owners if they have a vehicle

This means that all owners are customers but not all customers are owners. Because of this, we decided to represent it as a 0 to 1 (optional) relationship between the strong entity, Customer, and the weak entity, Owner, as it would mean that a customer may be an owner.

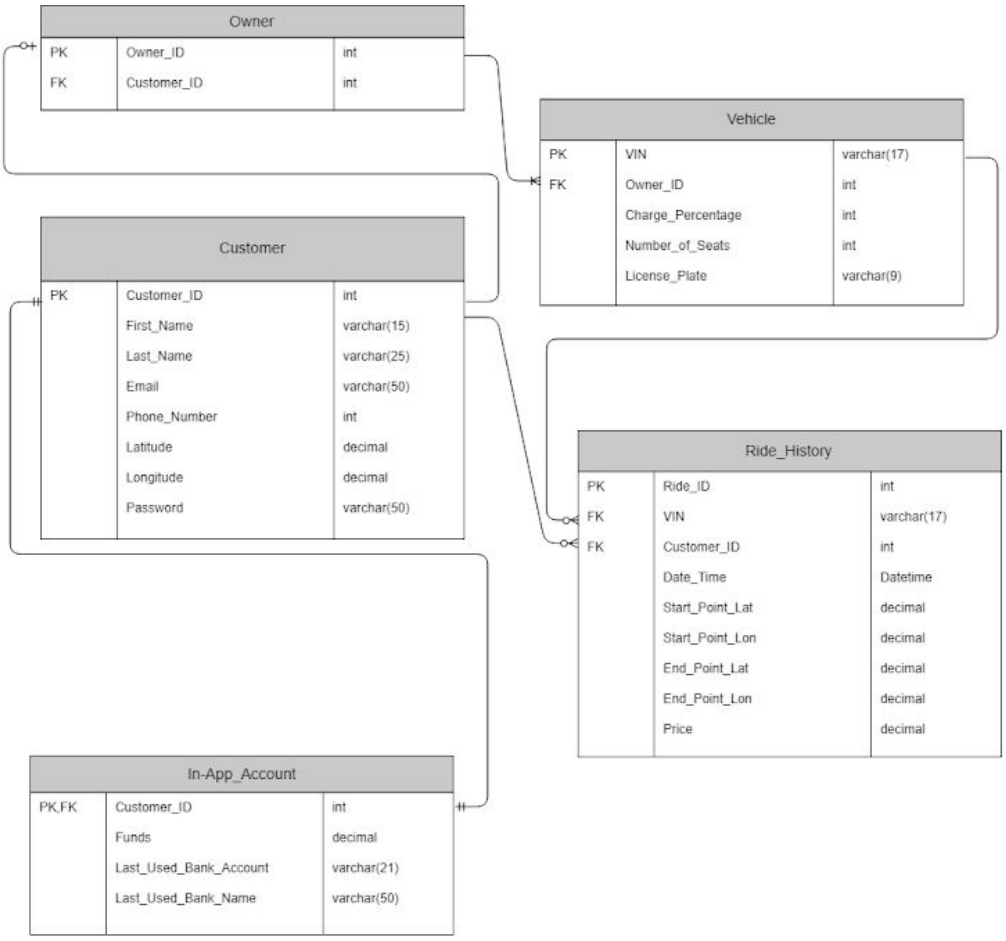


Ride Histories should be recorded and created based on the vehicle used and the customer

We want the ride history of customers to be recorded so the customer or owner of the car can look at the history of rides and report an issue if need be. Because of this, the ride histories will be based on vehicle used and the customer and will have an ID of its own. Since the Ride_History will be dependent on both Vehicle and Customer, it will be a weak entity.



ER Diagram



Storyboard

Scenario: Jim wants to get to the airport but he doesn't want to drive his own car or park his car at the airport. He decides to request a ride to the airport using the no steer rideshare service app.

Step 1:
Because the scenario shows the users how the No Steer rideshare service works.

Step2: Prepare rough presentation outline

Login	Request the ride	Notify the cars after requesting	Accept the request by the car
Display the price	Confirm the service	Scan the QR code	Press “Go” when the customer is ready
Payment	Rate the Service and write the review	Convenient	Car brand
Provide water in the car	Camera inside the car	Send out car’s information	Thanks for using the service

Step 3: Remove the weak parts

Login	Request the ride	Notify the cars after requesting	Aceppt the request by the car
Display the price	Confirm the service	Scan the QR code to get into the car	Press “Go” when the customer is ready
Payment	Rate the Service and write the review	Convenient	Car brand
Water is provided in the car	Camera inside the car	Send out car’s information	Thanks for using the service

Step 4: Add section headers

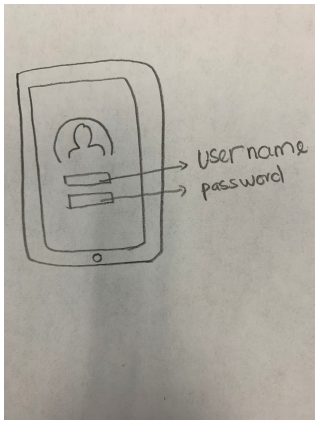
Request the ride	Log into the app	Enter the destination and current location	Confirm the pickup and the price for the service	Notify the cars after the request
Accept the request by the car	Notify the user when the car is about to arrive	Send the user the QR code	Send out the car’s information to	

	in 1 minute		the user including models, license...etc	
Scan the QR code to unlock the car	Put the QR code in front of the car's QR code, placed in the corner of the windshield	Unlock the car if the QR codes are matched	Resend the QR code if the customer request when the old QR codes are not matched	

Step 5: Prepare your final presentation outline

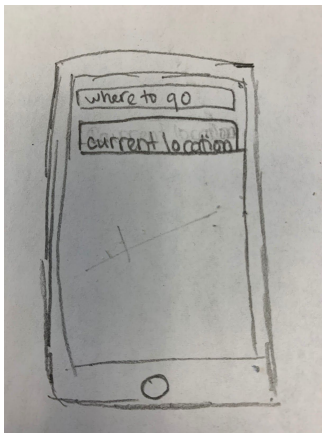
Log into the service	**Request the ride**	Enter the pickup location and destination	Confirm the pickup
Notify all the cars after the request	**Accept the request by the car**	Send out the QR code to the user	Send out all the information about the car the user for authenticity
Notify the users when the car is about to arrive in 1 minute	**Scan the QR code to unlock the car**	Put the QR code in front of the car's QR code, placed in the corner of the windshield to scan	Unlock the car if the QR codes are matched
Resend the QR codes if the user request when the codes are not matched			

Step 6: Storyboard at presentation (rough sketches of slides)

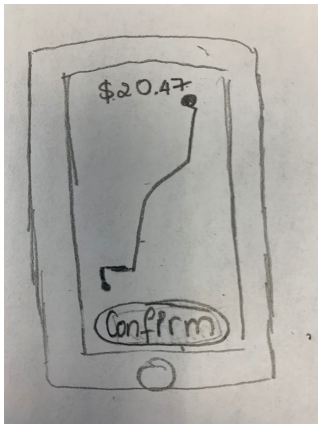


Log into the account

Request the ride



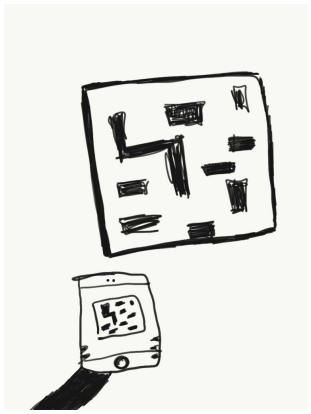
Enter the pickup location
and destination



Confirm the pickup

Accept the request by the car

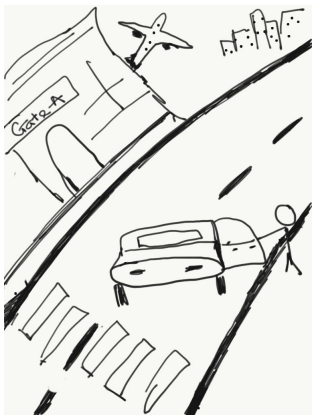
Get the car's information



Scan the QR code to unlock the car

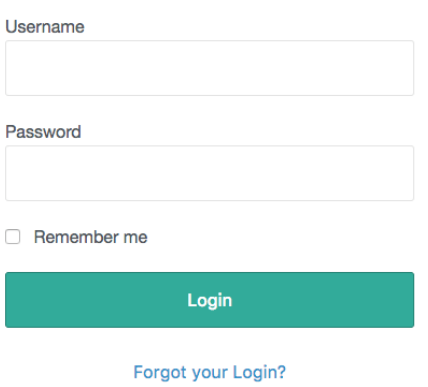
Unlock the car if the codes are matched

Resend the QR code by the user's request if the codes are not matched

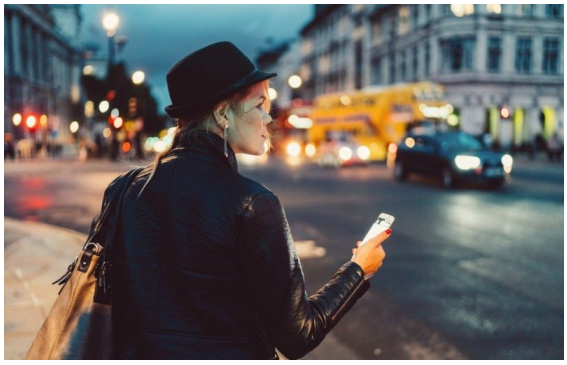


Drop off the user at the destination

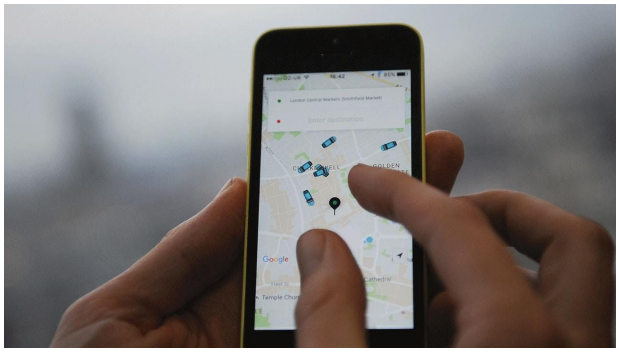
Step 7: Covert sketches into polished slides



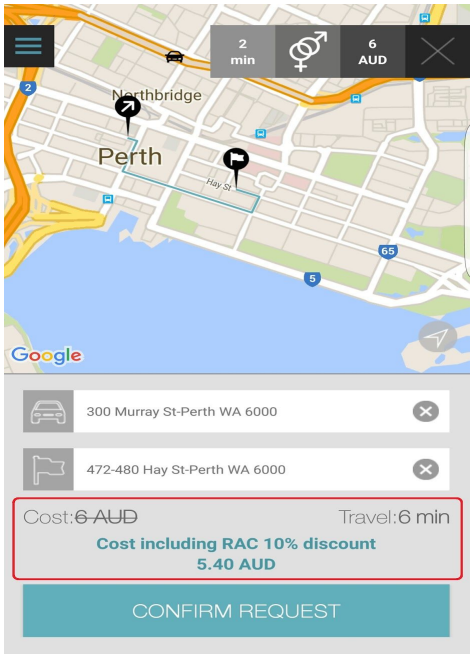
Jim logs into his no steer account by entering his username and password



Jim request the ride to go to the airpot

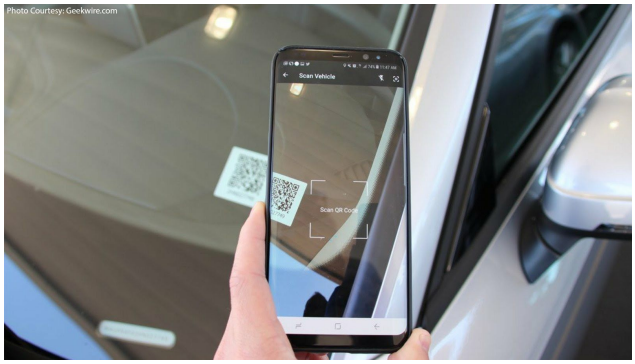


Jim enters the airport's location and his pickup location



After reviewing the price and time, Jim confirms the pickup

The car accepts his request and sends him the car's information and QR code



Jim points the camera in front of the car' QR code, which placed at the corner of the windshield to unlock the car

The car is automatically unlocked when his QR code matches with the QR code on the car

Jim gets dropped off at the airport

Wireframes

Research: Audience and Personas

- Customer - those who require transportation options
    - Personas:
        - Limited Public Transport - I want to go to campus on time but the bus is always late”
            - Individuals who needs to travel to locations where public transport is unavailable for use or any other reasons (such as timeliness).
- Owner - those who own an electric vehicle
    - Supplemental - “I like my job but it’s not enough to support my family of eight kids”
        - Individuals who have occupations but require a reliable method of gaining extra income. Thus, their free time will consist of using this service. They primarily choose this service because of the flexibility it provides them.

    - Commitment on Pay - “I have a car loan so paying it off by driving is convenient”
        - Individuals who use the service to pay off a loan or any other form of paying off something. Thus, once the loan is paid, they won’t use the service anymore until another loan appears

Research: Use Cases

Constructed through generative interviews with customers of ride sharing services and ethnographic observations of drivers of ridesharing services. Through the research, the use cases aligned with what is already established in previous sections. (see “Use Cases”).

- Owner Use Cases
    - Command vehicle to refuel
    - Know # of people within the vehicle at any time
    - Authorization for customers to open the Car
- Customer Use Cases
    - Provide estimated ride route and duration
    - Display traffic in the area
    - Display all cars in the area

Research: Evaluating Competitors

In terms of evaluating competitors, Uber and Lyft is used to compare. The criterias are defined by how each competitors define mandatory aspects of a ride sharing service. The criteria also includes delighter factors, which are factors that exceed the mandatory aspects of a ride sharing service. This is part of the criteria so that delighter aspects of competitors can be implemented in the project if they are relevant.

Criteria	Uber	Lyft
Route information	Provides estimated time of pickup, price, and route taken.	Provides estimated time of pickup, price, and route taken.
Notification	Display’s if driver is nearby and when driver has arrived	Display’s if driver is nearby and when driver has arrived

Range	Also alerts any messages and calls by the driver	No error is shown. Thus, requesting a trip that is out of the country is possible
Recent rides	No limits. Thus, if destinations is another country, it spits out poor connection error	Presents itself as two categories, which are personally paid, or business related. Information of a ride's history provides date, time, miles covered, pickup and drop off time, duration of ride, and receipt.
Trip Planning	Available to plan at any time within the year. Planned trips are stored under the ride history	Available to plan at any time within the year. Planned trips are stored under the ride history
Report Issues	No live feedback. Instead all error related requests goes straight to customer service to be replied later on.	Provides a bot. The bot which redirects to Lyft support team if issue isn't resolved.
Drivers in the Area	Displays only on nearby area	Displays all drivers within 5 miles of the array
Service Variation	Pool, Uber X, Black, Select, UberXL, Black SUV, WAV, and Taxi	Shared, Lyft, Lyft XL, Lux, Lux Black, Lux Black XL, and green mode
Verification	Displays driver's name, car information (license plate, color, make and model), trip record, and time employed	Displays driver's name, and car information (license plate, color, make and model).
Delighters	Loyalty Program which provides benefits to those who use the service often	Local Information such as public transit and allows payment option through employer

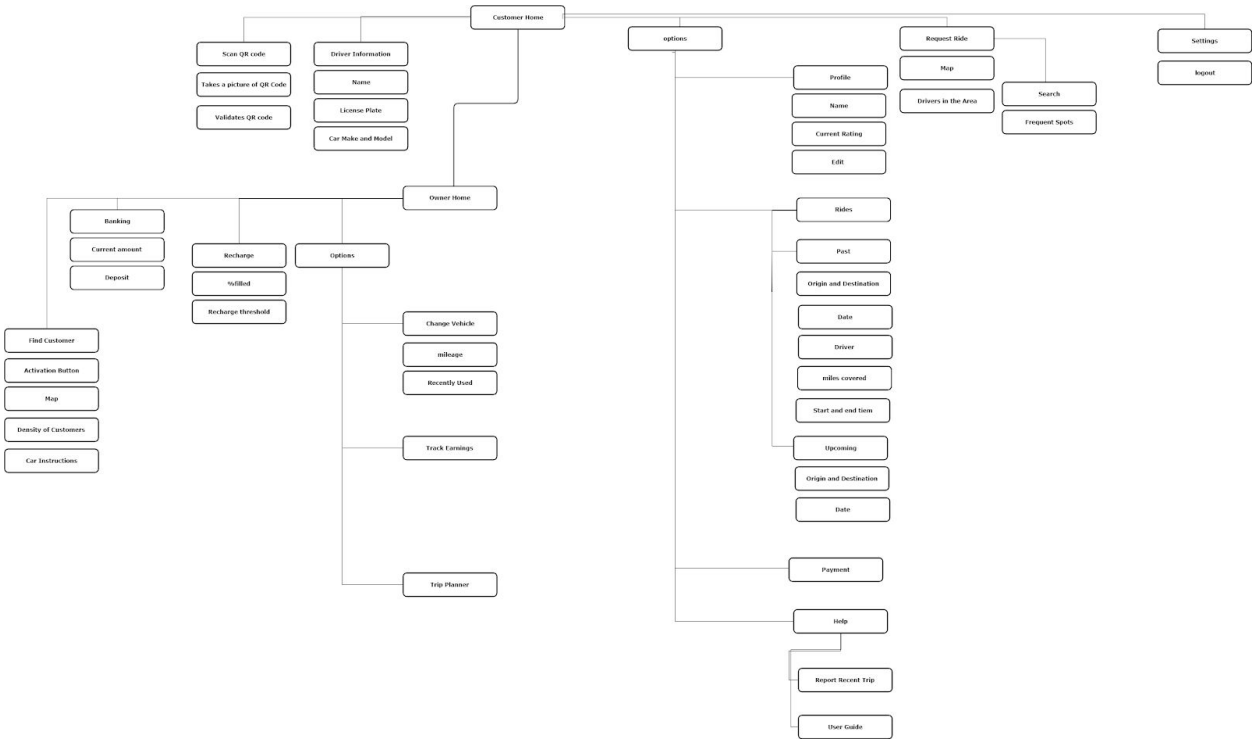
Cheat Sheet

The cheat sheet in this scenario includes a summarized and compact version of the project's business, user personas, use cases, and delighters. In terms of use cases, since it it already well explored in previous sections (see "Use Cases"), a sample is given.

- Business - Rideshare service of self-driving vehicles.
    - Goal 1 - To maximize the potential of self-driving cars in ridesharing.
    - Goal 2 - To provide a cheap and reliable rideshare service for users.
    - Goal 3 - To provide a secure service for self-driving car owners to share their car
- User Personas
    - Customer Personas:
        - Limited Public Transport - "I want to go to campus on time but the bus is always late"
    - Owner Personas
        - Supplemental - "I like my job but it's not enough to support my family of eight kids"
        - Commitment on Pay - "I have a car loan so paying it off by driving is convenient"
- Use Cases
    - Customer
        - Owner Use Cases
            - Command vehicle to refuel
            - Know # of people within the vehicle at any time
            - Authorization for customers to open the Car
        - Customer Use Cases
            - Provide estimated ride route and duration
            - Display traffic in the area
            - Display all cars in the area
- Delighters
    - Uber
        - Loyalty Program which provides benefits to those who use the service often
    - Lyft
        - Local Information such as public transit and allows payment option through employer

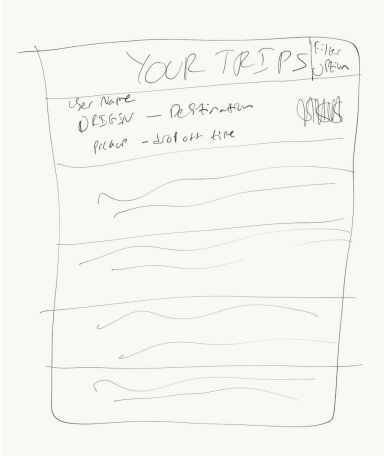
User Flow: Information Architecture

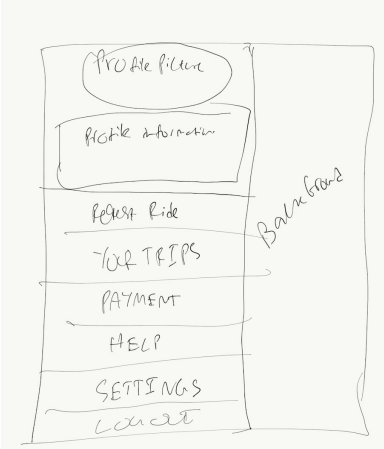
Information architecture is determined through a colleague led card sorting session. The session allows a method of sorting the hierarchy of information presented and the navigation pages expected by the user. With the information architecture, it will optimize the wireframing process as the order of information is already determined.

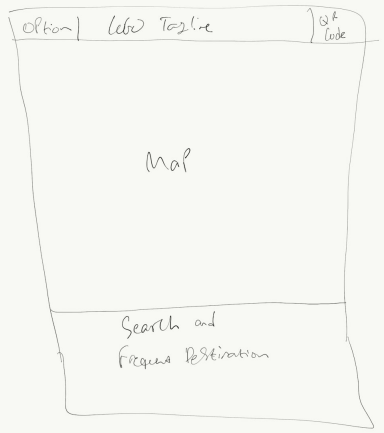


Wireframes Sketches: Drafts

The following are samples of wireframe sketches for “Your Trips”, “User Option”, and “User Hub”. Only samples of sketch wireframes are included to compensate for the large size of high fidelity diagrams. The purpose of this sketches is to decide where information should be placed in the screen. Since an information architecture is constructed, this process was easier to follow. This is because the information architecture already determined which information is prominent. In addition, the sketches also outlines all possible actions in a page. This is presented by the buttons and touch points.

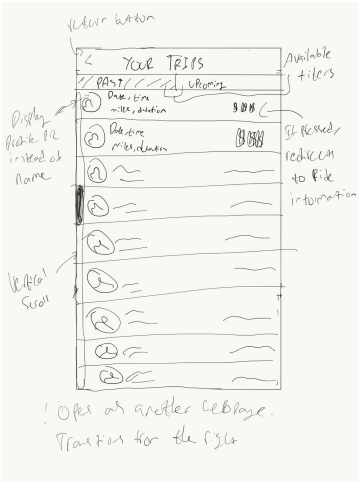


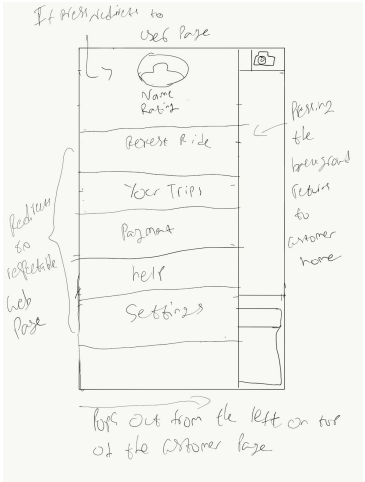


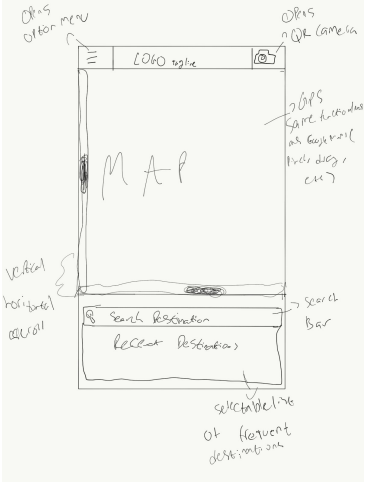


Wireframes Sketches: Detailed

The following are samples of further detailed versions of the preliminary sketches. To reiterate, only samples of sketch wireframes are included to compensate for the large size of high fidelity diagrams. The purpose of these sketches is to show how the user would interact with the display and what would happen when they do. For example, if the user scrolls around in the map, the vertical and horizontal scroll bars will move accordingly. This level of additional detail is needed as it better showcases the view the user will have when utilizing the app. It also makes it easier for other members on the team to branch off from as there is a copy that shows how the UI changes based off of user input.







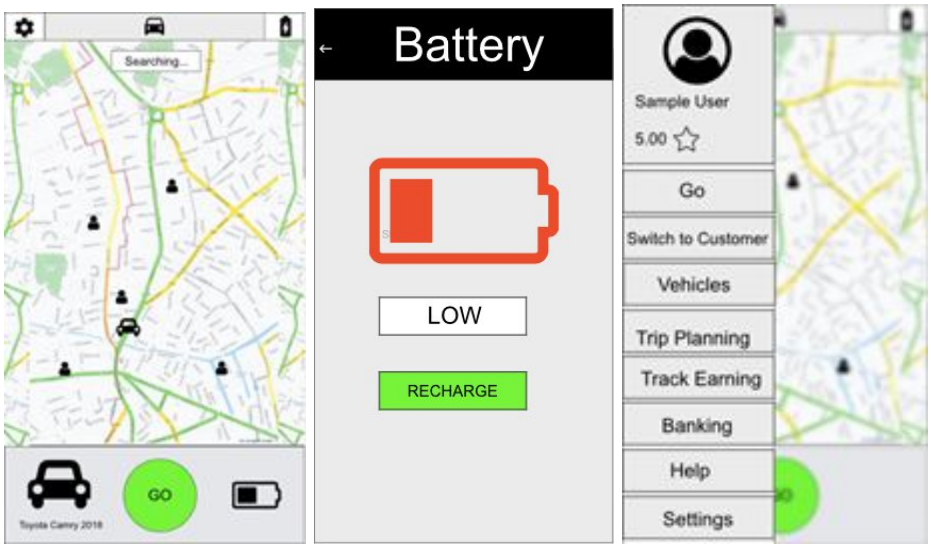
Wireframes Prototype: Low Fidelity

The following are samples of low fidelity wireframes from the customer's view of the app. The goal of creating low fidelity wireframes is to evolve detailed sketches a simple prototype defined by block diagrams. With this prototype, it can be used to test user flows and determine any violation towards trust-building elements.

Wireframes Prototypes: High Fidelity

The following is high fidelity of all navigations pages described by the information architecture. Unlike previous sections, all navigation pages are included to present the final content and behavior and blueprint for the final design.

Owner Perspective

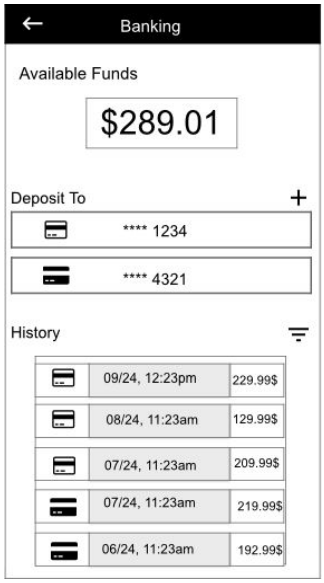


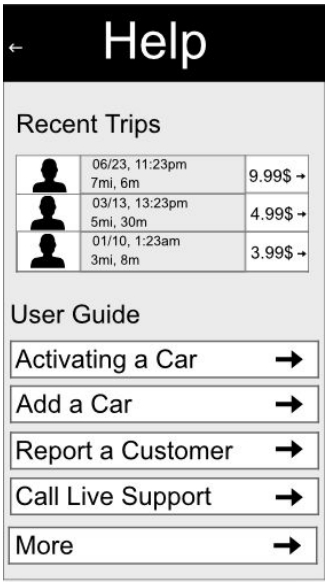

Your Profile
"I am a user and I used this app to support my finances"
JEFFY
JEFFJEFF
jeff@gmail.com
298-391-1930


Your Vehicles
ACTIVE CAR
Toyota Camry
2018 11,533 mi
YOUR CARS
Toyota Camry
2019 70,321 mi
Set as Active
Honda Civic
2010 187,321 mi
Set as Active

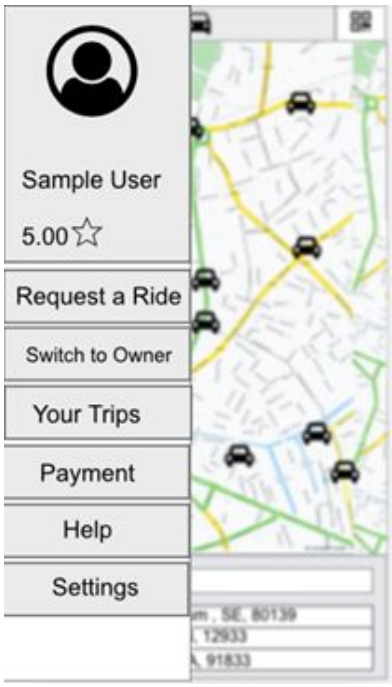

Trip Planner
Hourly Trends
University of Washington
Service Time
Activate By
Return by
Service Type
BasicPool
Premium
Basic
PremiumXL
Basic XL
WAV

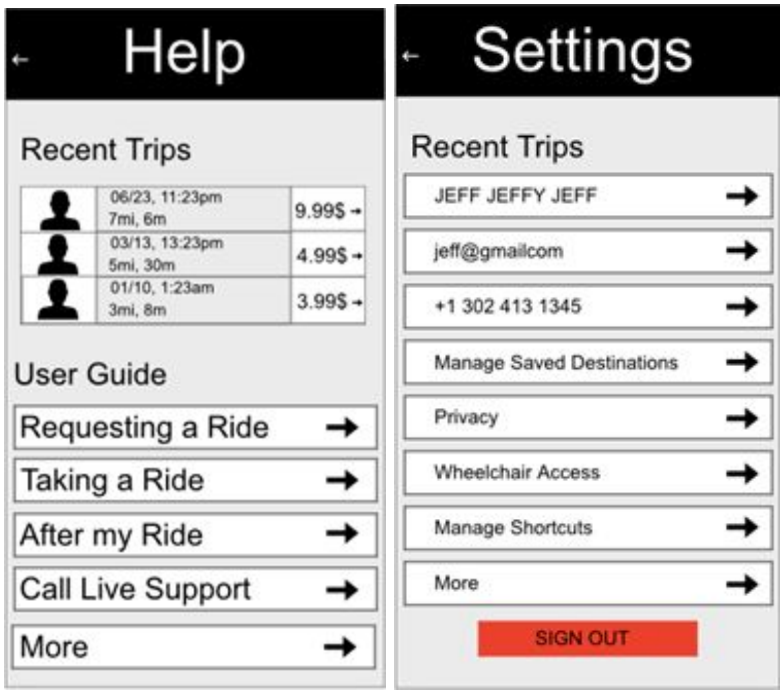

Track Earning
Weekly Earnings
$9.20
Mon
Tues
Wed
Thurs
Fri
Sat
Sun
Today's Earnings
$289.01
21 Trips Completed
09/24, 12:23pm
5mi, 5m
9.99$
09/24, 11:23am
5mi, 5m
9.99$


Banking
Available Funds
$289.01
Deposit To
**** 1234
**** 4321
History
09/24, 12:23pm
229.99$
08/24, 11:23am
129.99$
07/24, 11:23am
209.99$
07/24, 11:23am
219.99$
06/24, 11:23am
192.99$


Help
Recent Trips
06/23, 11:23pm
7mi, 6m
9.99$
03/13, 13:23pm
5mi, 30m
4.99$
01/10, 1:23am
3mi, 8m
3.99$
User Guide
Activating a Car
Add a Car
Report a Customer
Call Live Support
More

Customer Perspective

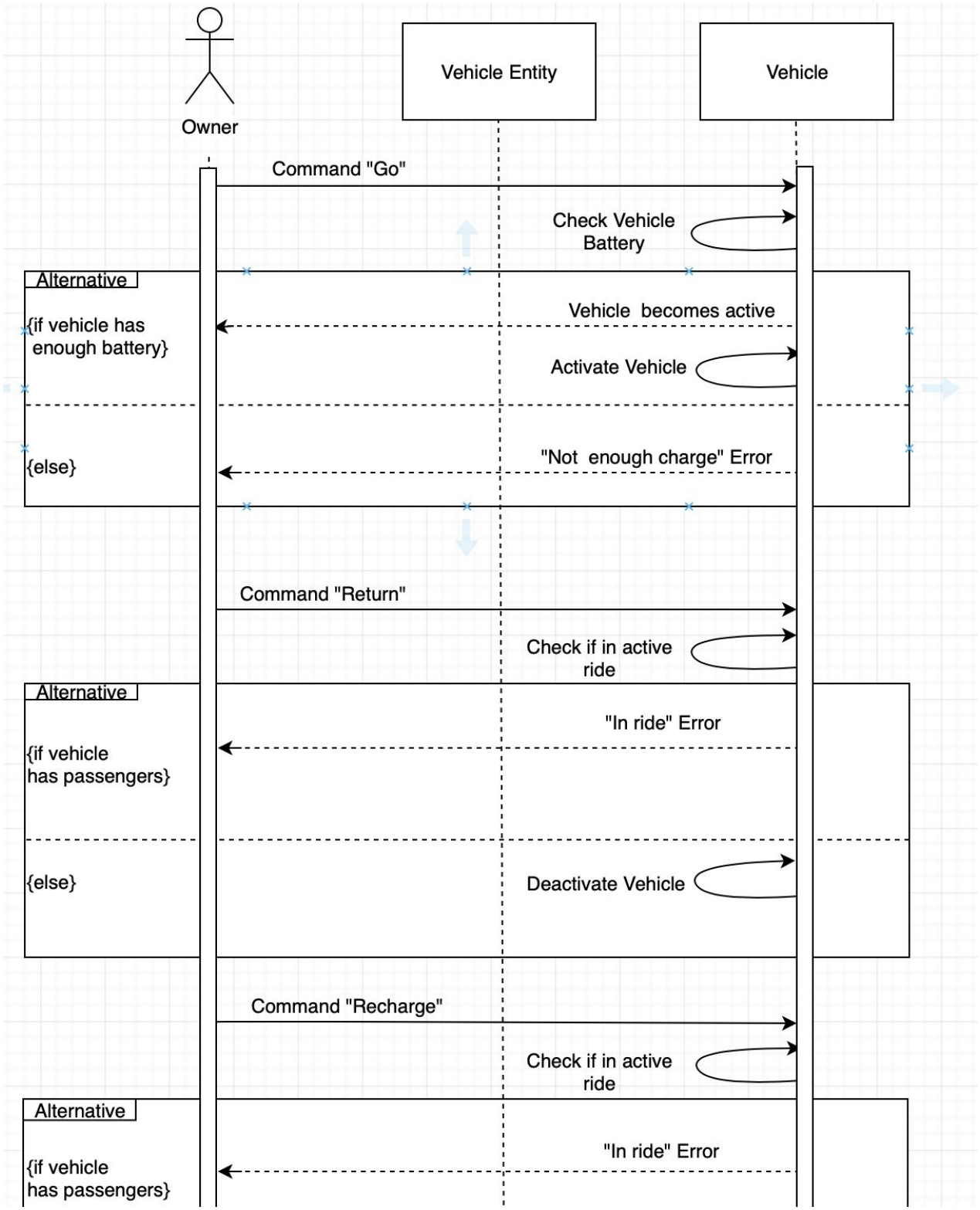


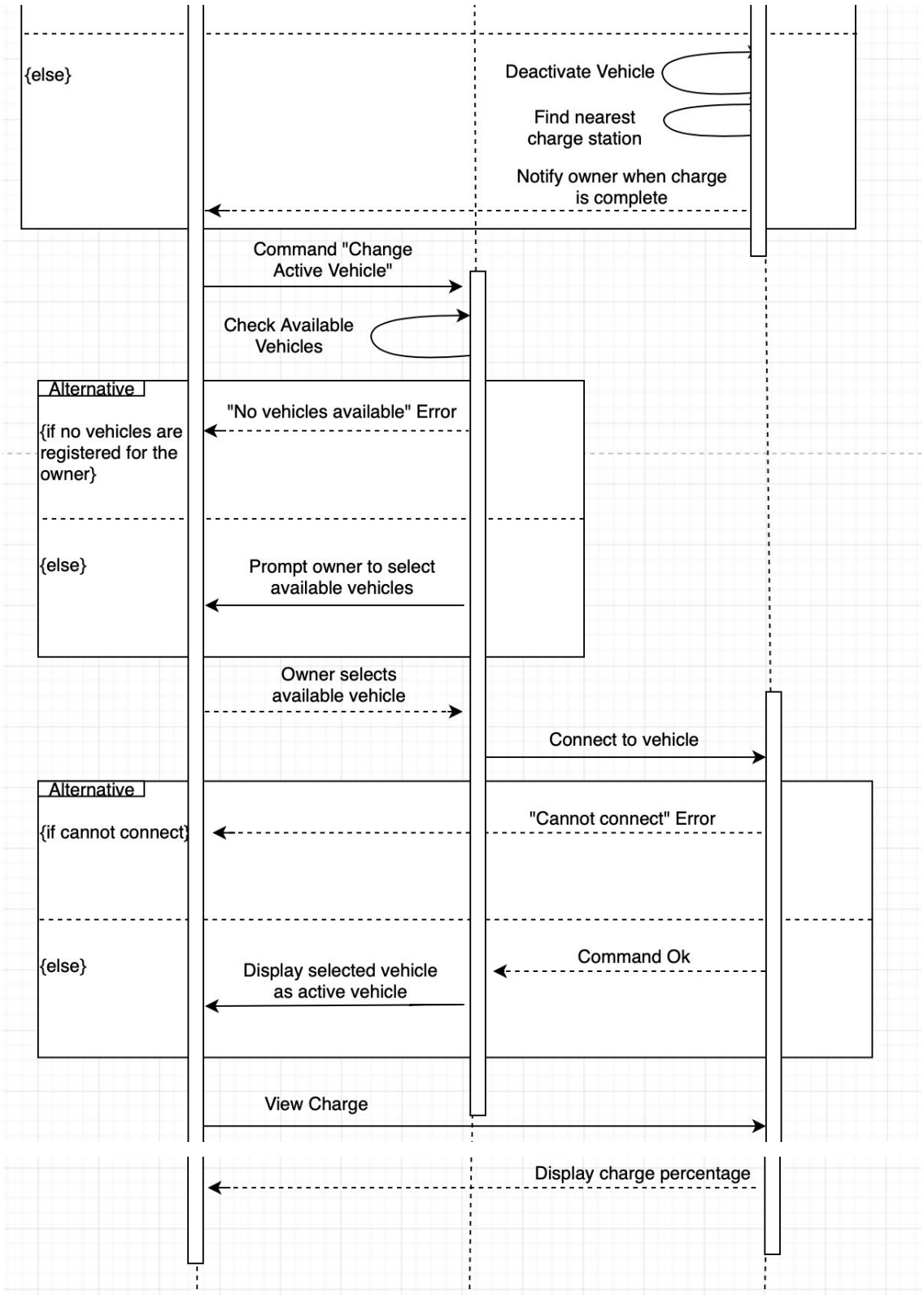


Owner Sequence Diagram

This is a sequence diagram for the Owner's main functionalities. These capabilities are the Owner to be able to command their active vehicle to go service rides, come back to

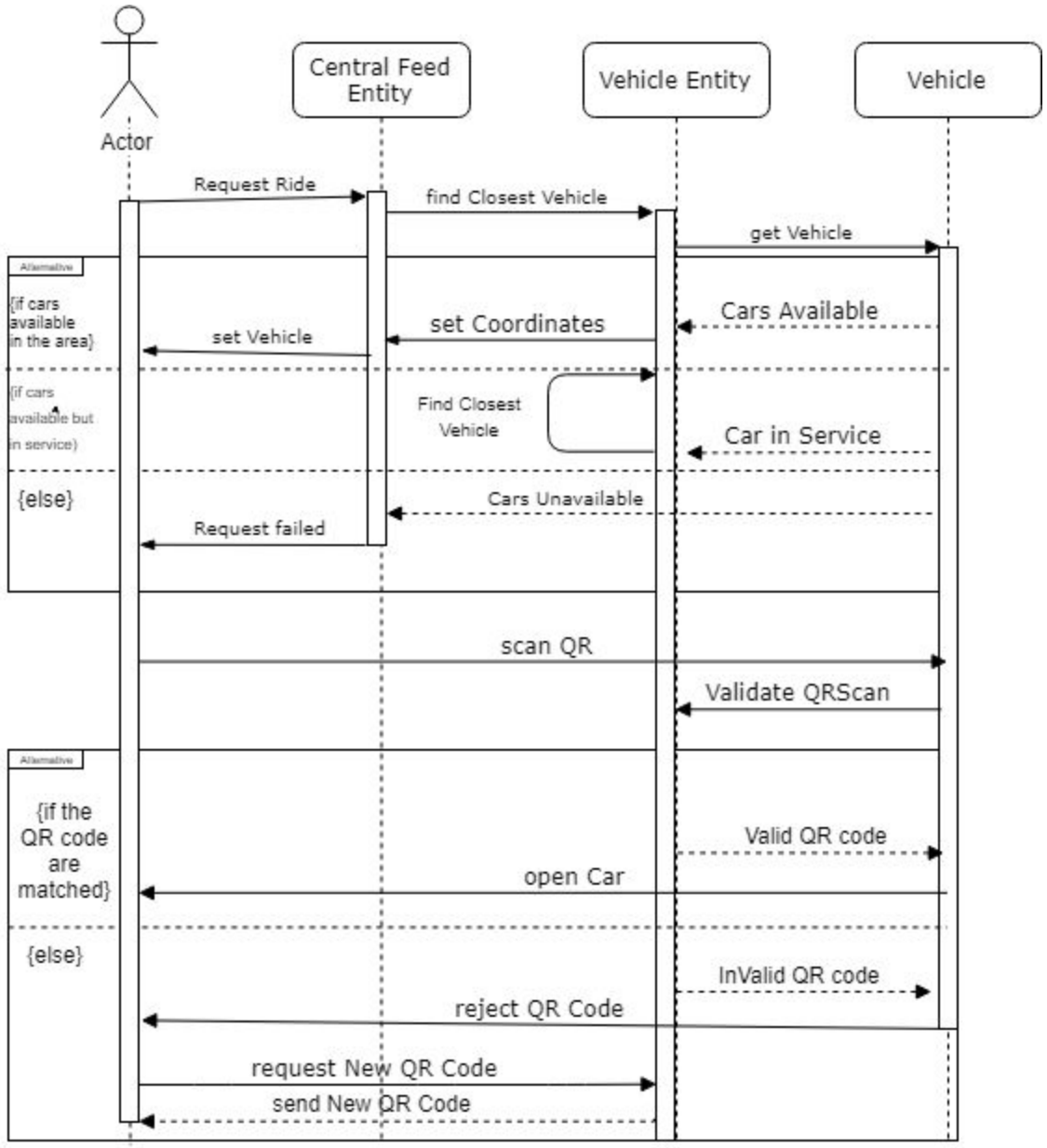
them, and recharge. On top of controlling their vehicles' independent movements, they can also decide which of their vehicles are active and see the battery percentage of each. This was made by Aaron Handjojo and Pratit Vithalani.




{else}
Deactivate Vehicle
Find nearest charge station
Notify owner when charge is complete
Command "Change Active Vehicle"
Check Available Vehicles
Alternative
{if no vehicles are registered for the owner}
"No vehicles available" Error
{else}
Prompt owner to select available vehicles
Owner selects available vehicle
Connect to vehicle
Alternative
{if cannot connect}
"Cannot connect" Error
{else}
Command Ok
Display selected vehicle as active vehicle
View Charge
Display charge percentage

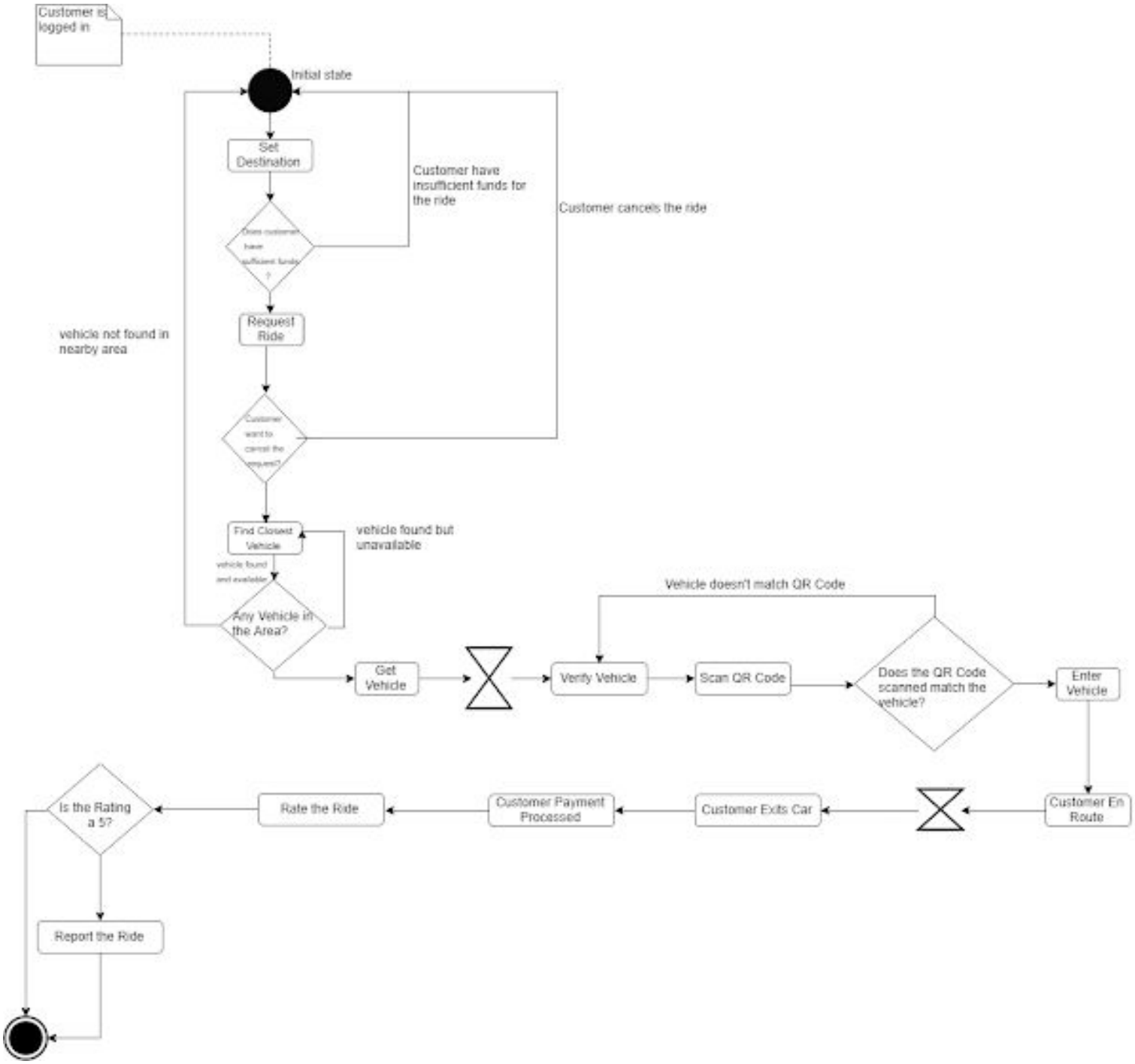
Customer Sequence Diagram

This is a sequence diagram of the Customer's main functionalities. The customer's interactions in the app revolve around the use of a vehicle to get to where they need to go. This means that the customer will need to be able to request a ride and verify to the vehicle that they are who they say they are. Both of these functionalities are shown in the following diagram, made by Sesario Imanputra and Phuc Huynh.



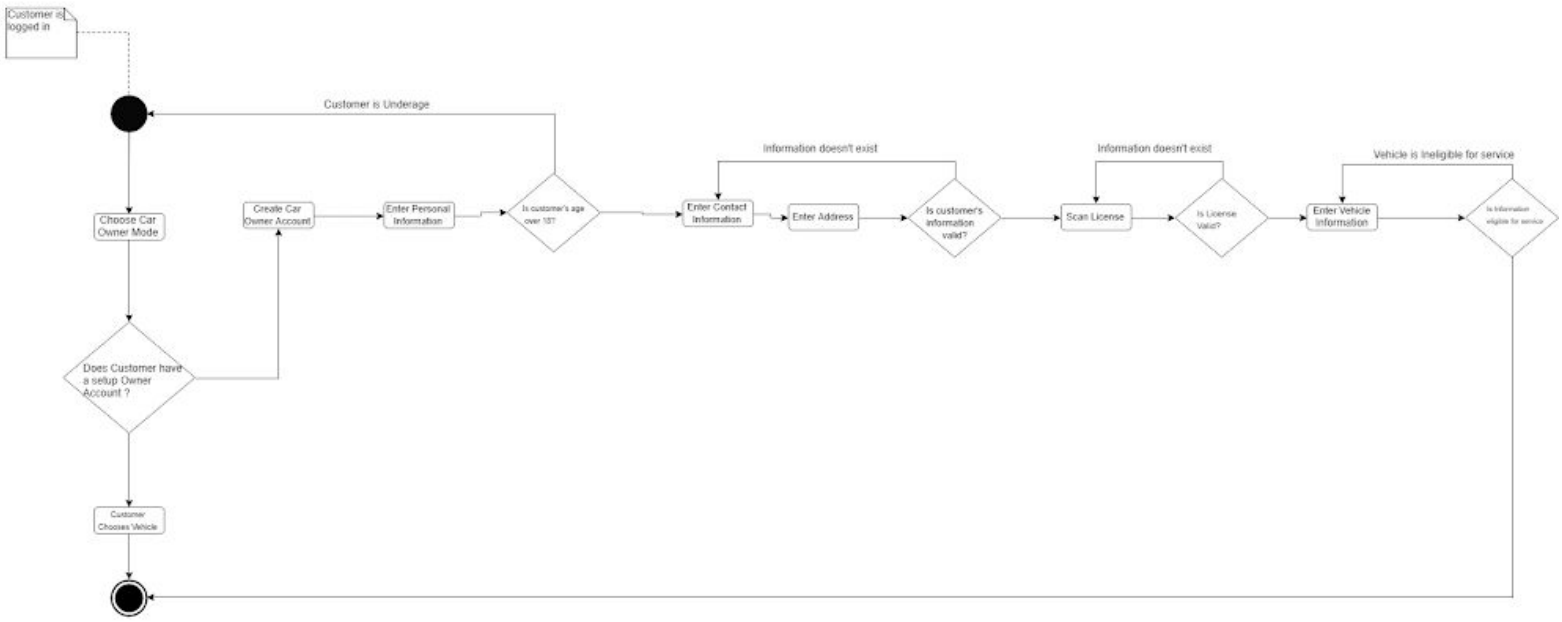
Customer Activity Diagrams

All the Customer Activity Diagrams made by Sesario Imanputra and Phuc Huynh. The first activity diagram describes the customer's workflow on the process of requesting a ride until the customer has reached the destination. Due to its lengthy



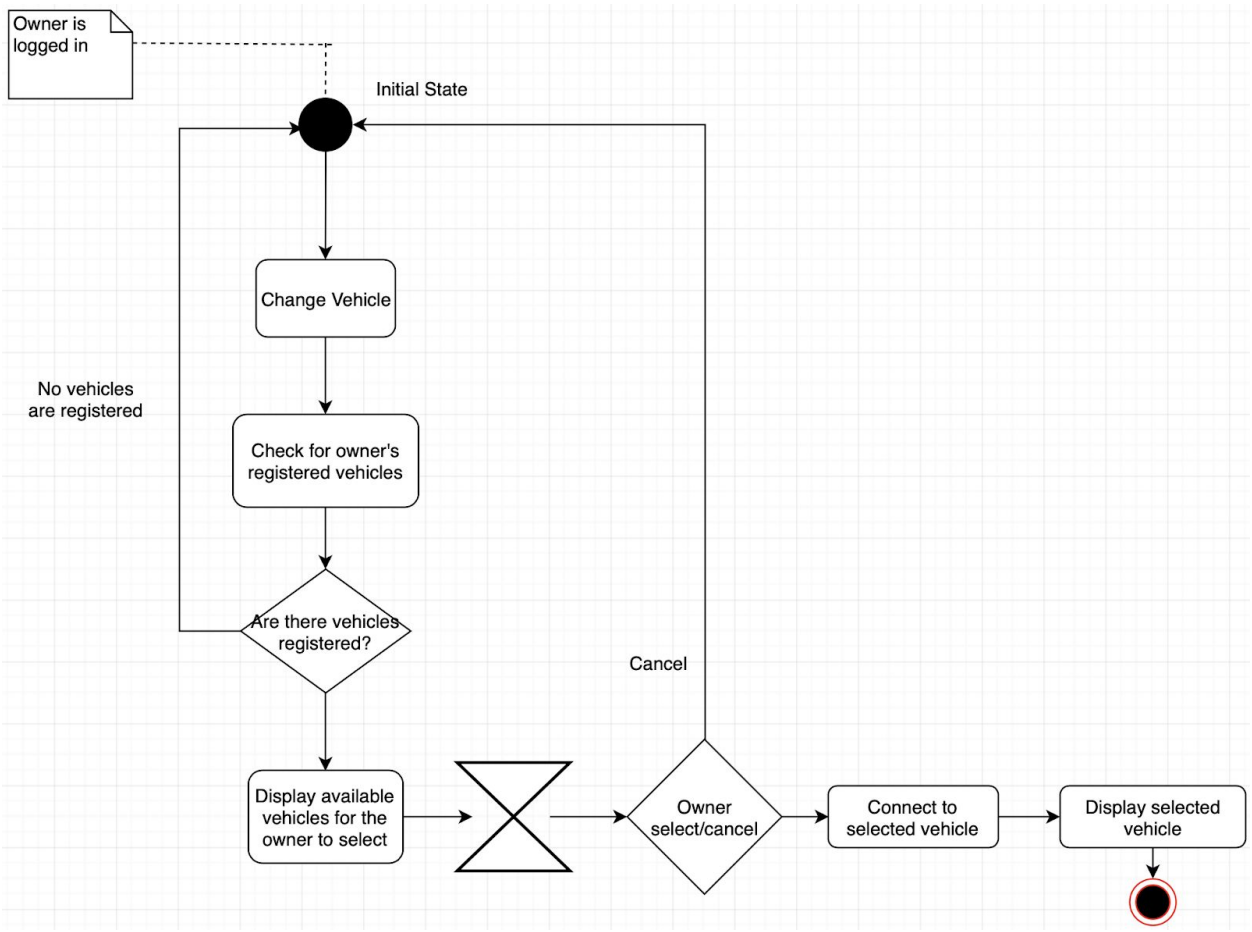
process, it also covers cancelling a request, finding the nearest vehicle, scanning the QR code, filing a report.

The second activity diagram describes the customer's workflow on switching to car owner mode, including registering as a car owner.

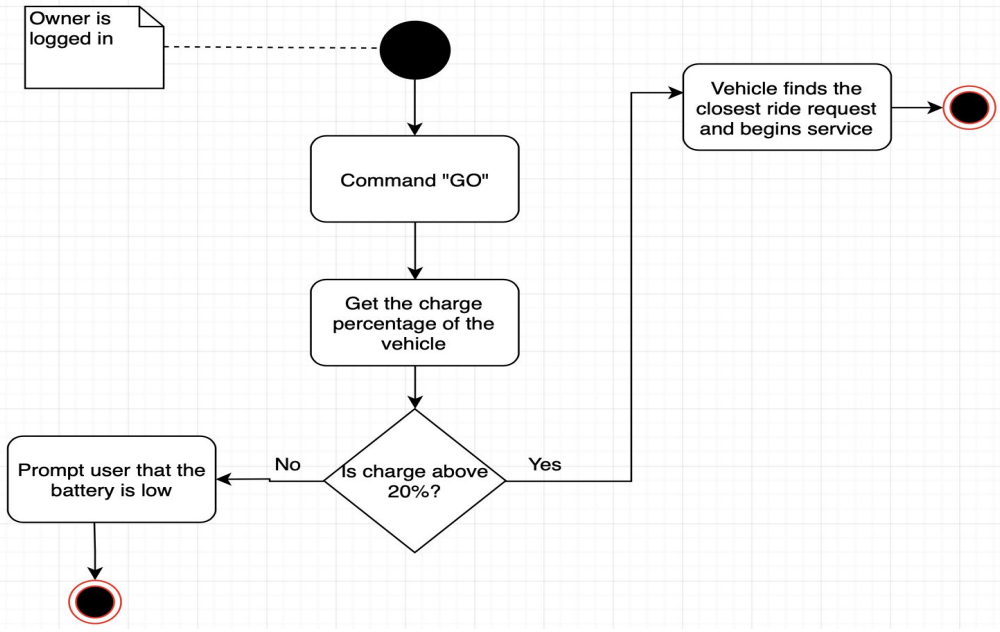


Owner Activity Diagrams

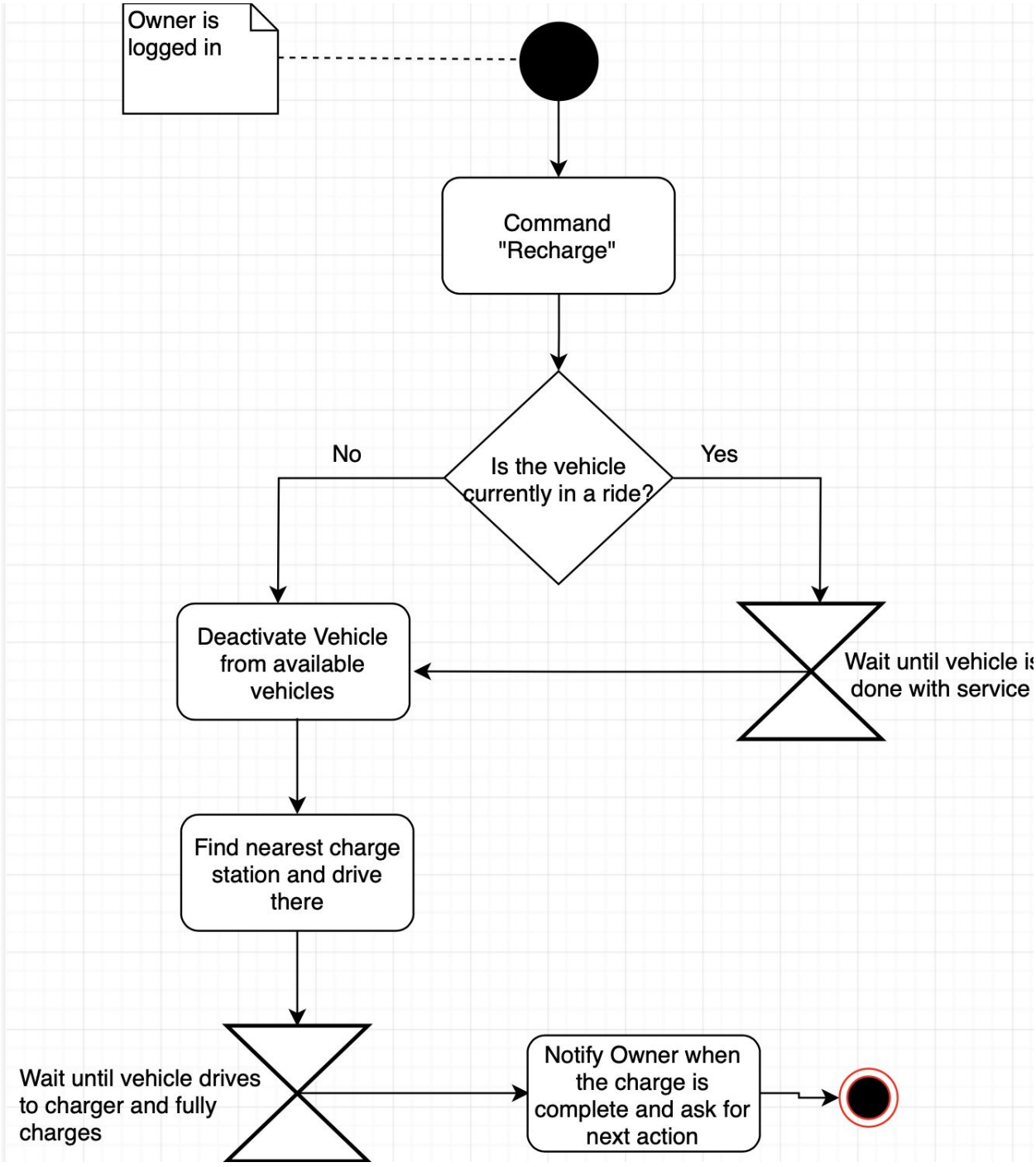
The Owner Activity Diagrams are made by Aaron Handjojo and Pratit Vithalani. This activity diagram shows the process behind the owner changing their active vehicle (the vehicle that is in currently in service).



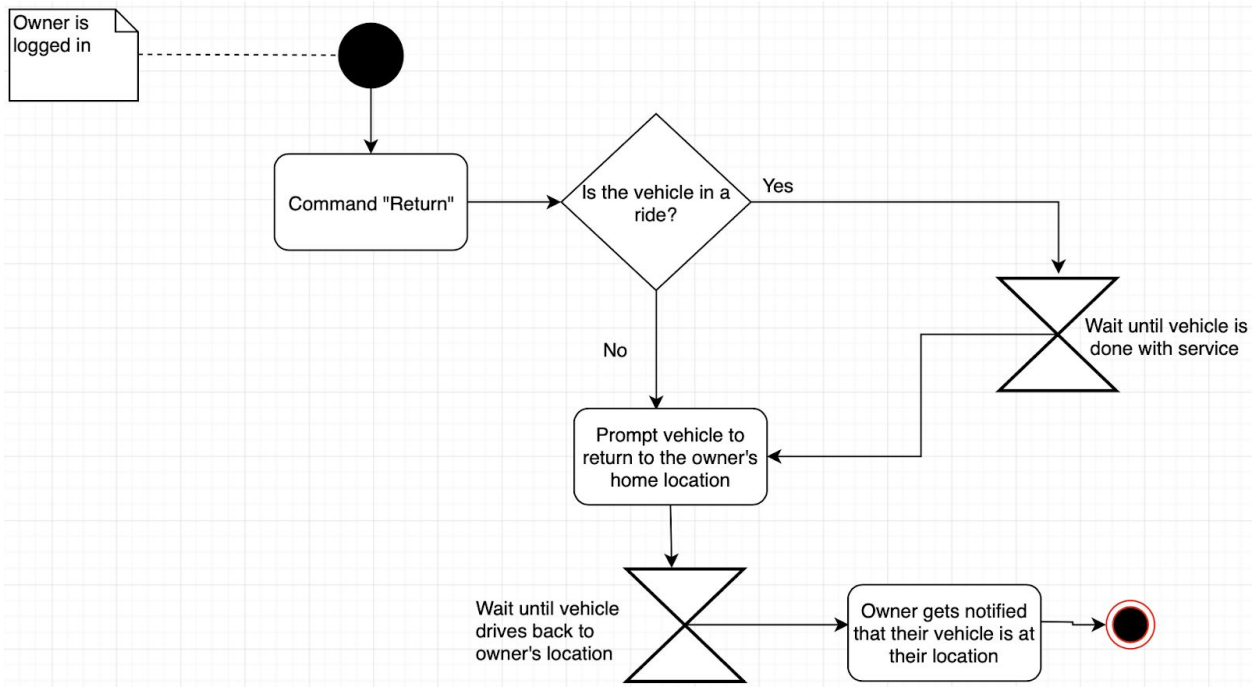
This next activity diagram shows the process in which an owner commands their vehicle to “Go” and get rides on the streets.



The following activity diagram demonstrates what happens when the owner sends their active vehicle to go and recharge the battery.



The final activity diagram illustrates the steps that go into having a vehicle return to its owner.



Data Flow Diagram

Organizing List of Activities

The following activities, which are based of use cases, are organized into their respective process:

1. Request

Request Details

Set Vehicle

2. Authentication

Scan QR Code

Open Vehicle

3. Report

Report Customer

Vehicle Feedback Report

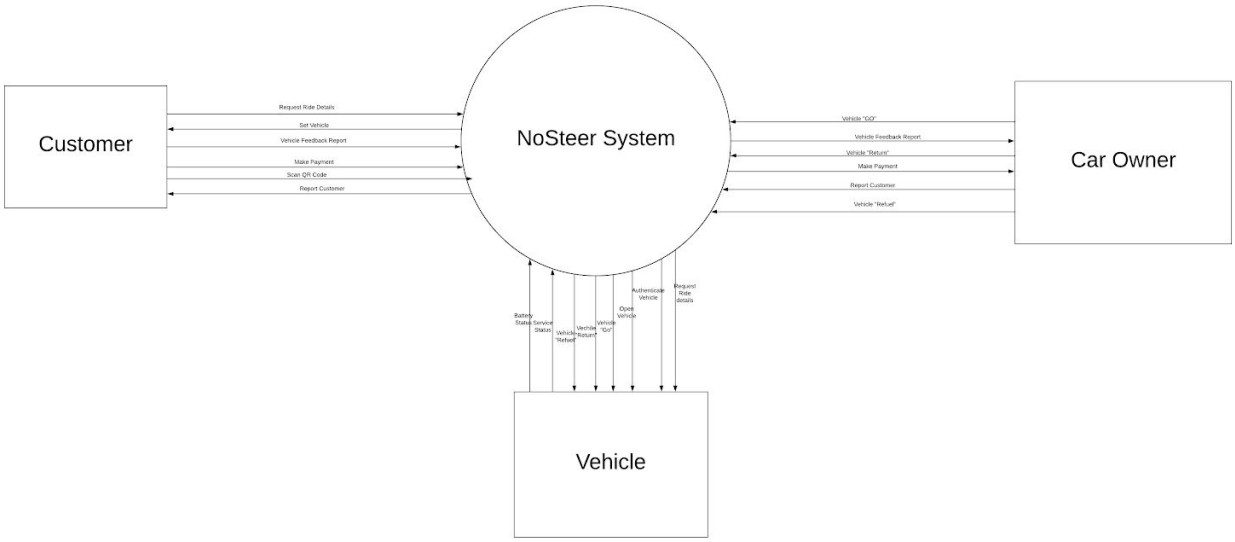
4. Payment

Payment Method

5. Maintenance
Vehicle "Refuel"
Vehicle "Return"
Service Status
Battery Status

Context Level Diagram

The following diagram uses the organized list of activities into dataflow. This reveals the NoSteer System process and three external entities, which includes customer, car owner, and vehicle.



Level 0 Diagram

The following diagram decomposes the NoSteer system into logical subsystems:

Request
Accepts request ride details
In order to pair the request with the nearest vehicle.

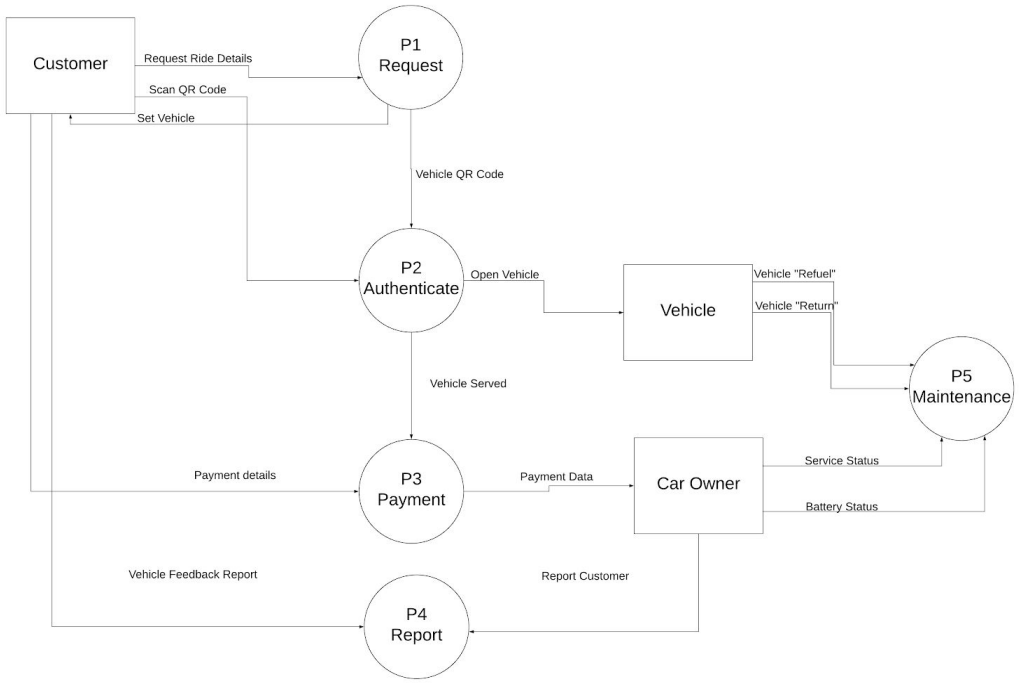
Authenticate
Accepts a QR Code sent from the user and compares it with their paired vehicle. In return, the vehicle door is opened.

Payment
Accepts payment details from the customer to be processed to car owner.

Report

Accepts report form from either customer or car owner.

Maintenance
Accepts car owner commands, allowing to car to vehicle as expected while recording history of vehicle related actions.



Process Decomposition

The following diagram describes how the logical subsystems presented in level 0 will be decomposed further in details:

Request
Coordinate Validation
Verifies if the coordinate isn't out of range of local area. Ex: local destination is in Seattle but destination is to Spain
Vehicle Request
Pairs the nearest vehicle to the given request details.

Authenticate
Verify QR Code
Verifies if the QR code sent by the user is the same as the paired vehicle

Vehicle Door
Opens the car if QR code is verified

Payment

Payment Gate
Verifies that payment method exist (ex: credit card)

Verify Payment
Verifies that system is able to obtain detailed amount.

Report
Verify Report
Verifies that the report sent is error free.

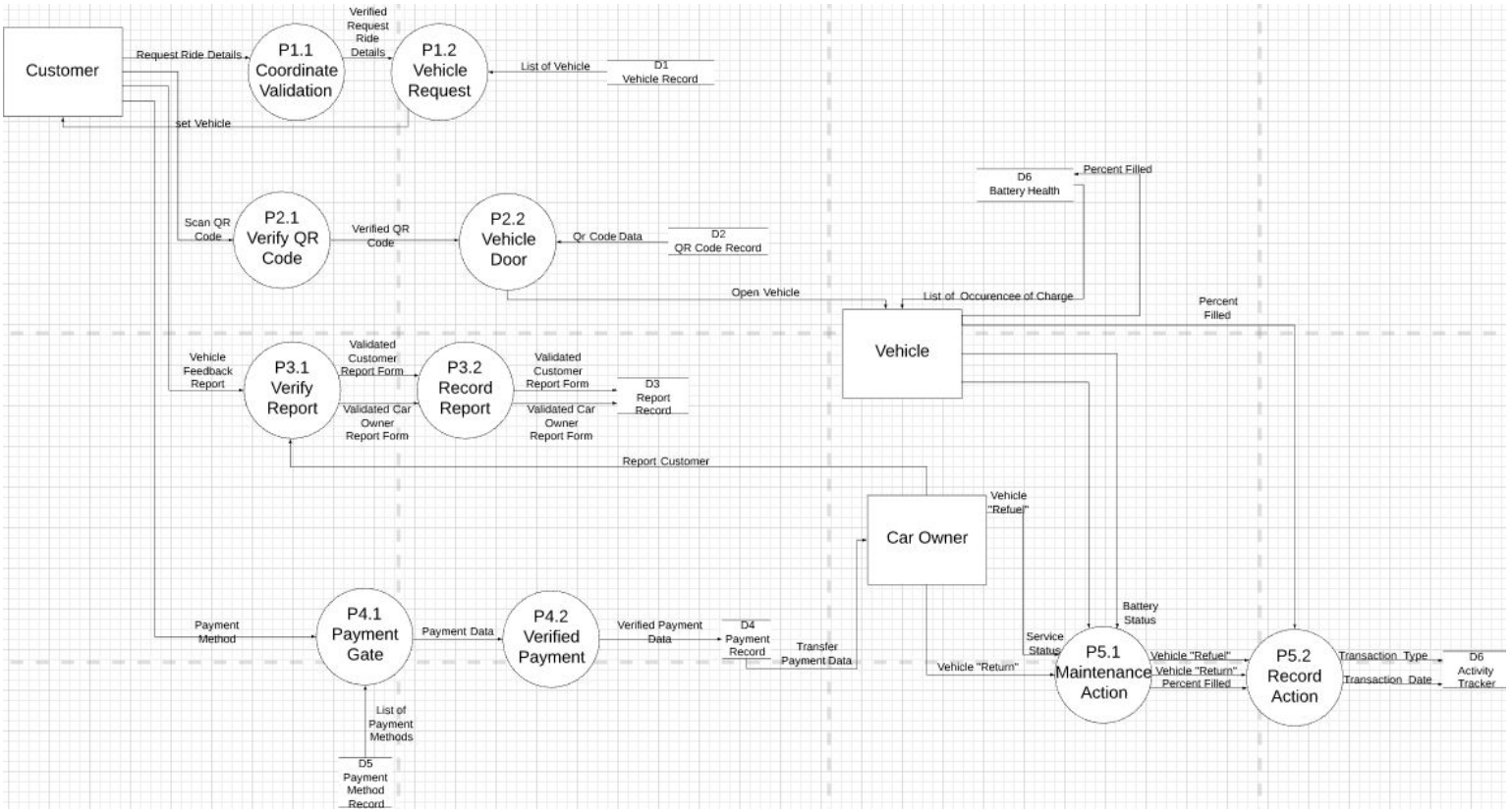
Record Report
Records report in the data store

Maintenance
Maintenance Action
Determines if action selected is possible

Record Action
Record report in the data store

Level 1-n Diagram

The following diagram uses the description and processes from the process description along with relevant data store:

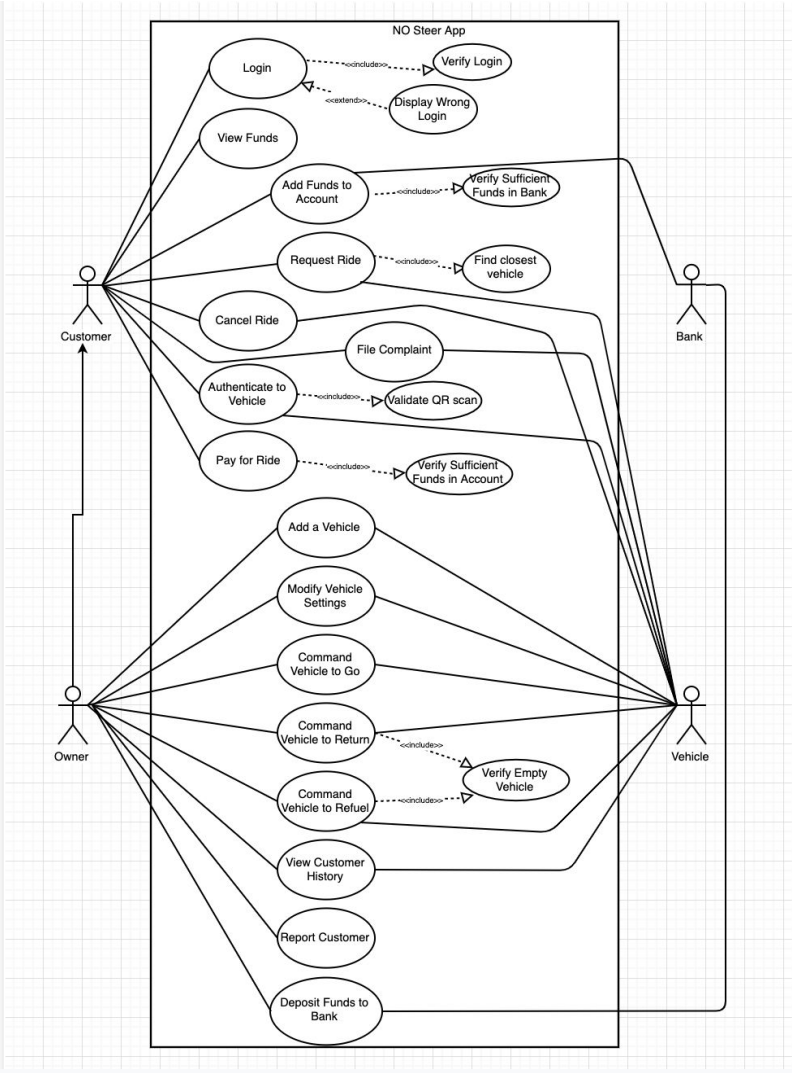


NOTE: Poor image quality since images must be smaller than 25 megapixels/

4+1 View Model

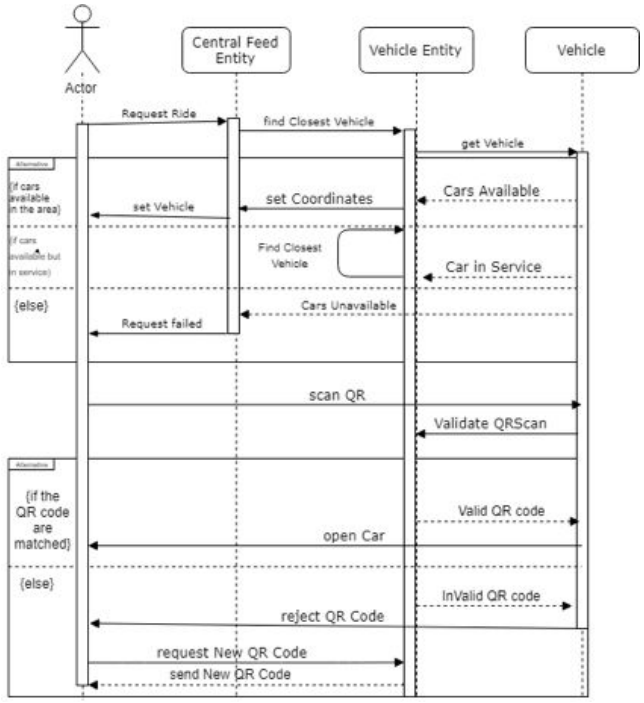
Scenarios

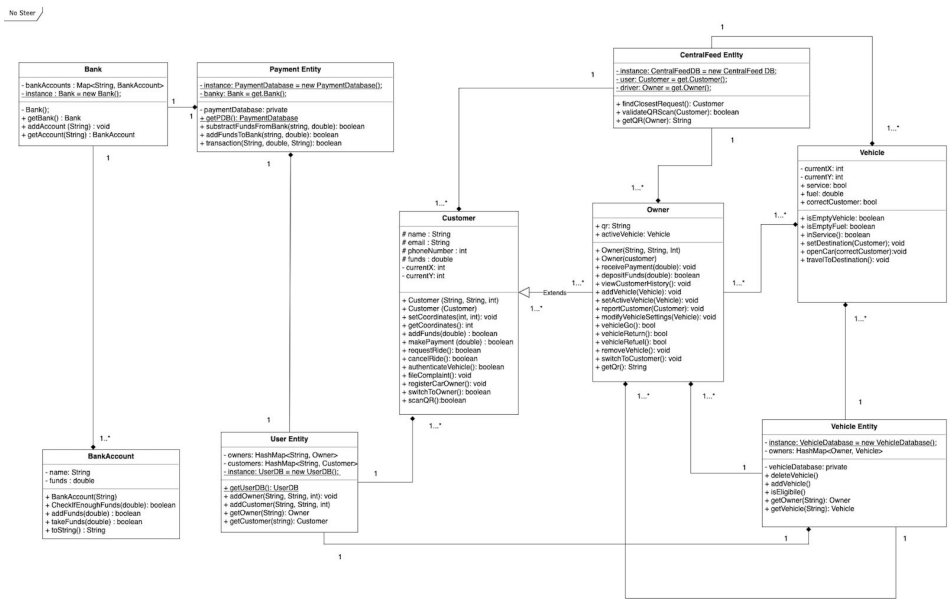
This shows the use case diagram. Since this was done before for workshop 9 & 10, its original images are shown in the Use Case diagram section.



Logical View

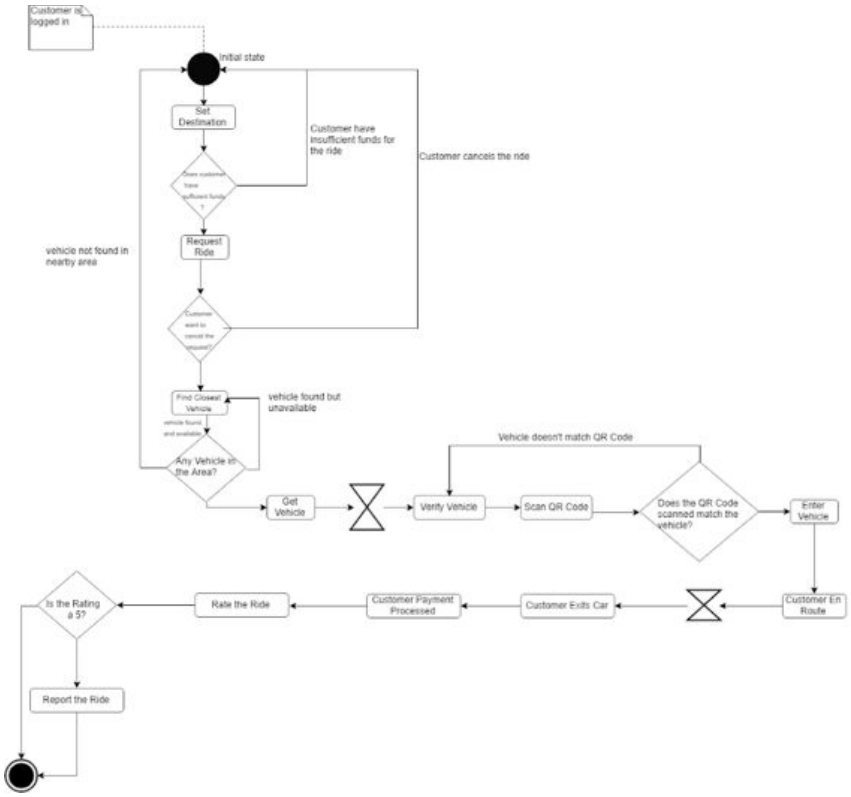
This shows the overall UML diagram and one of the customer sequence diagrams. Since this was done before for workshop 9 & 10, its original images are shown in the earlier sections.

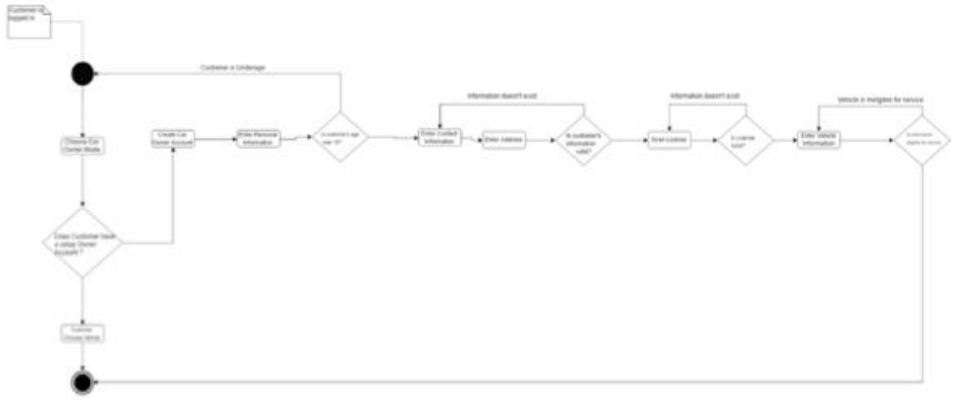




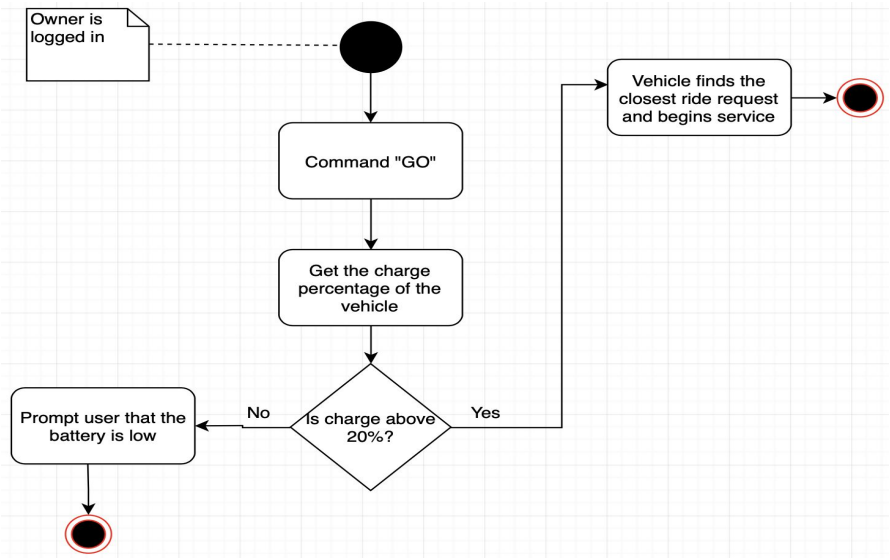
Process View

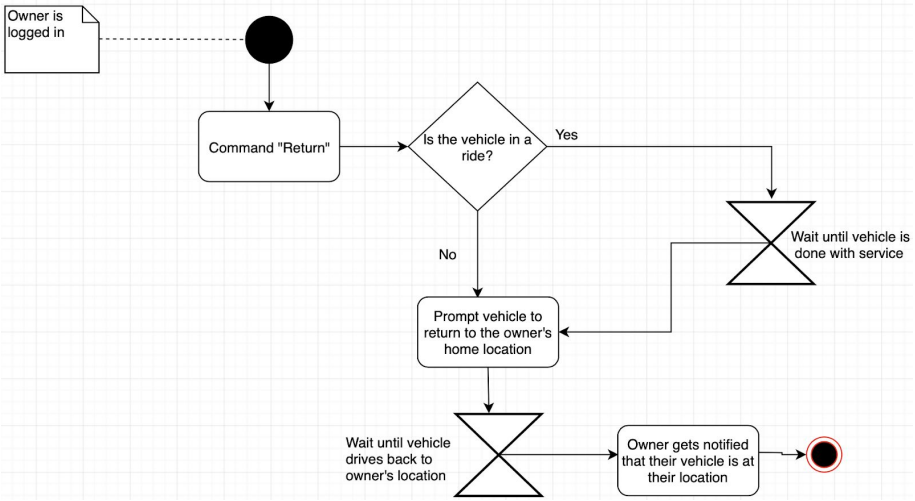
These are activity diagrams for the customer. Since this was done before for workshop 9 & 10, its original image is shown in the Customer Activity Diagram section.





These are activity diagrams for the owner. Since this was done before for workshop 9 & 10, its original image is shown in the Owner Activity Diagram section.

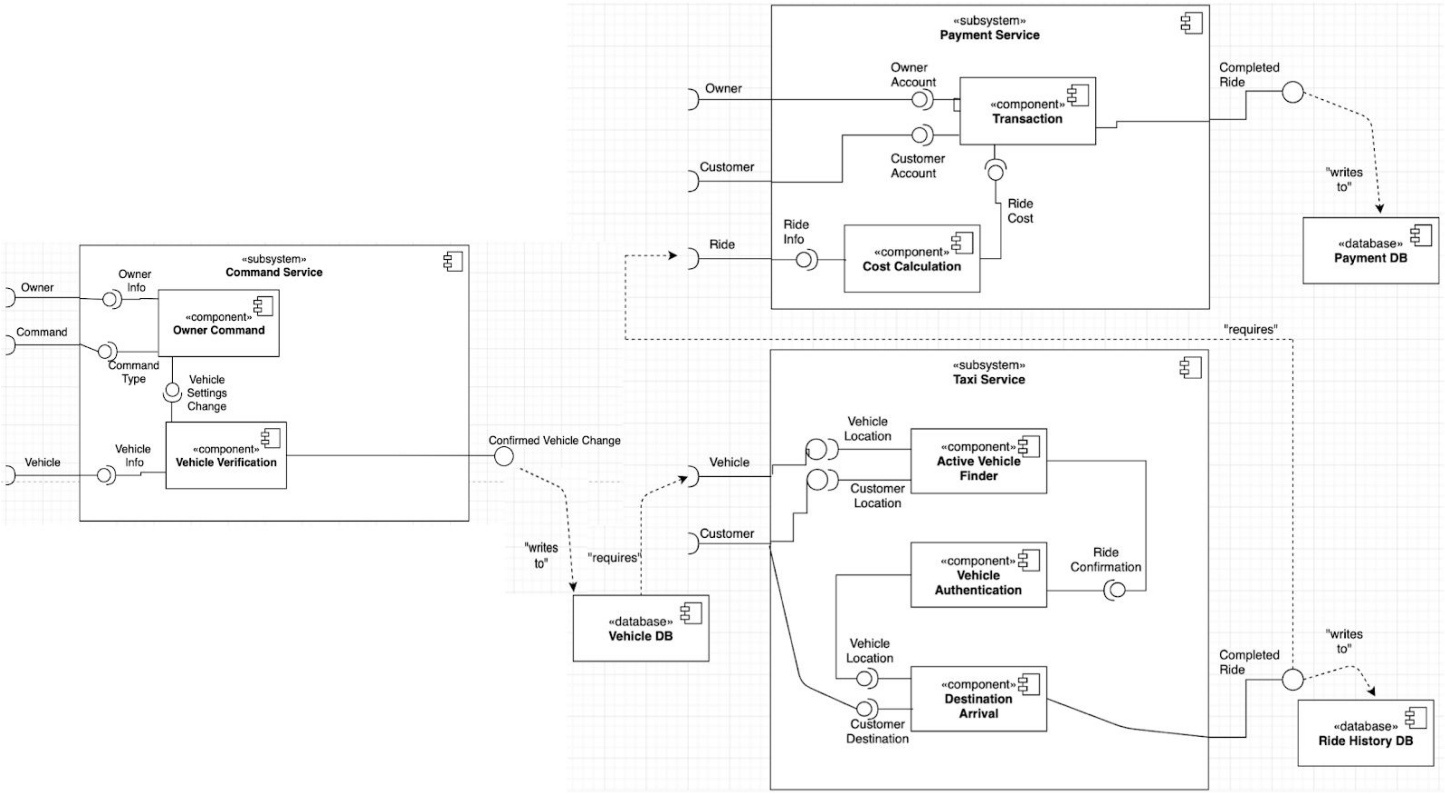




Development View

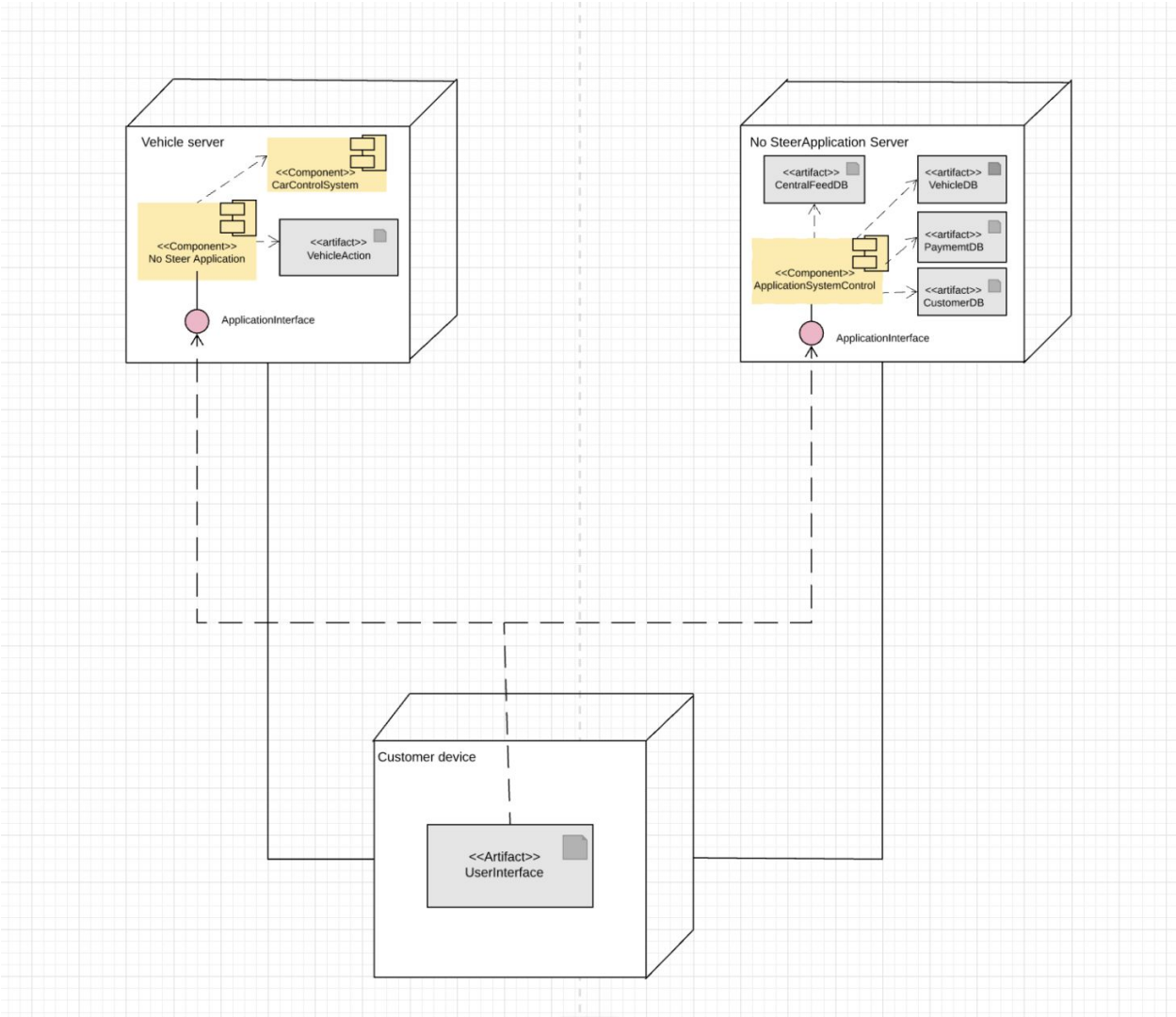
This is the component diagram for the entire system. There are three main subsystems: Payment, Taxi, and Command. The Payment shows the main interaction that happen

between the owner and the customer. The Taxi service shows the components involved in a customer interacting with a vehicle. The Command subsystem shows the components that are used when the owner changes the settings on any of their vehicles.



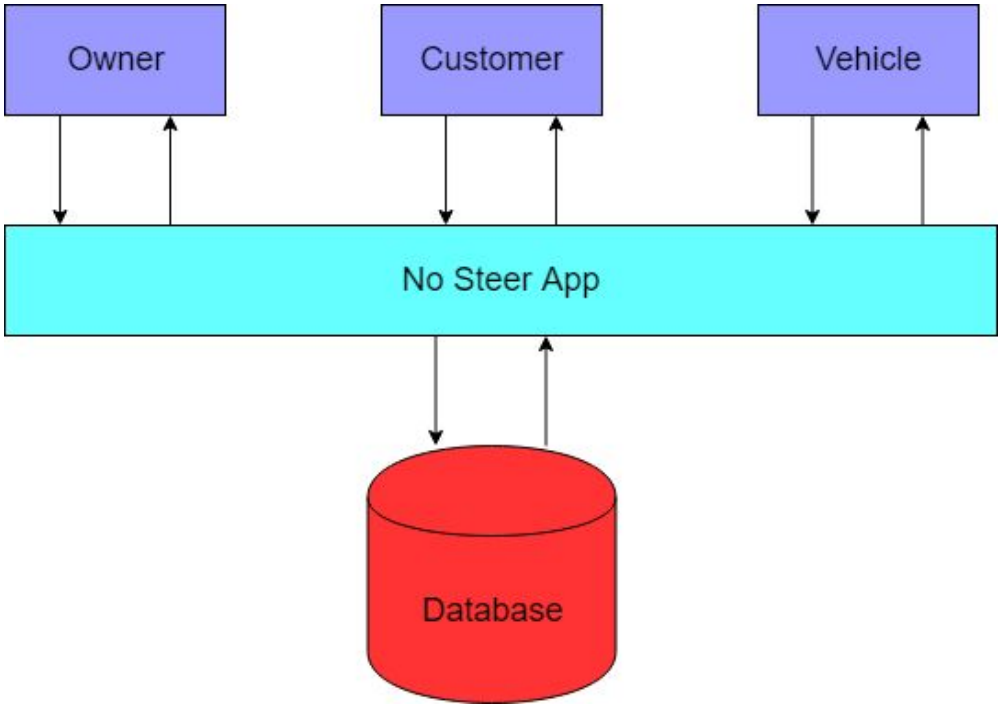
Physical View

This is the deployment diagram for the whole system. The three main pieces in this view are the application server, the vehicle, and the user's mobile device. The application server holds all of the data that is needed for the operation of our service, including the locations of the vehicles, amount of funds in the user's accounts, and the customer ride history. The vehicle is main vector of our product, but most of its software is made by its manufacturer. The only parts that we need is to control where it drives to and locking/unlocking the doors. The final part, the user's mobile device, does not require any hardware components from our end. The only thing that is needed is the No Steer app.



Prescriptive Architecture

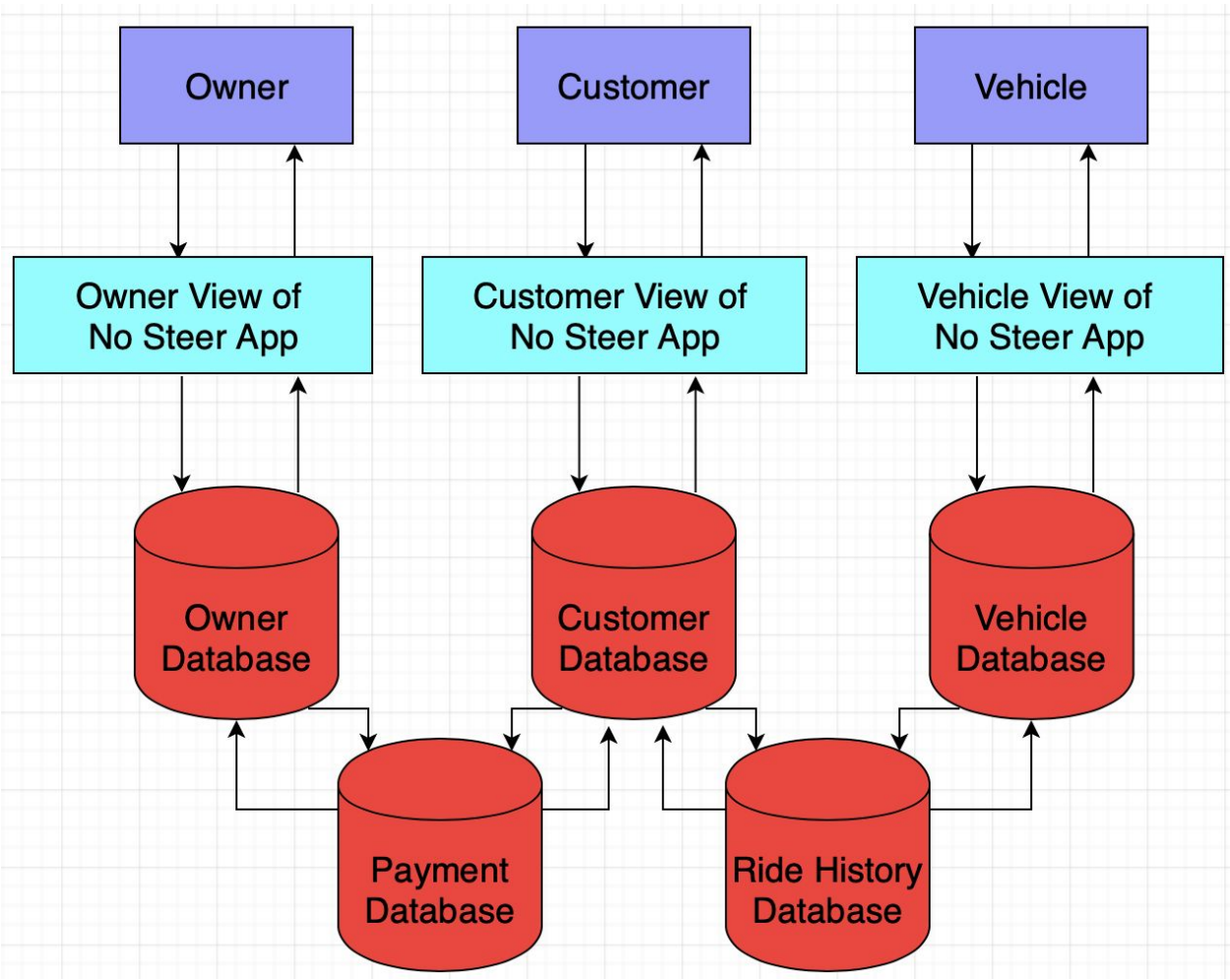
This is the prescriptive architecture that we envisioned our product to utilize. It is a simple client-server architecture with the users interacting with and interface which serves as a barrier to the database. We believed that we can hold all of our information one database and expand by replicating the database in multiple locations.



One problem that came up with this style is the fact that there would a bottleneck between the app and the database. That means that all the data flowing in and of the database would have to be from the same database. We see this as a problem because this would mean that we would have to specialize in database management, which is something that is product is not about. This product is about having a simple interface for the users to use to get places. That means that we have to focus on the app portion and spread out the database to keep it moving quickly in the back.

Descriptive Architecture

This is the descriptive architecture that we implemented. One of the key changes that was made from the prescriptive architecture is the fact that we split up the app portion into three parts. There will be a view for each user because that will allow for a easier separation of the databases, which has been split into five different databases (second key change). This was done to allow for scalability in the future. Since there would be less owners, customers, and vehicles than transactions and rides, we will be able to add additional payment and ride history databases. Giving each user their own database also allows us to see who has more traffic and adjust the number of databases according to that traffic.


Owner
Customer
Vehicle
Owner View of No Steer App
Customer View of No Steer App
Vehicle View of No Steer App
Owner Database
Customer Database
Vehicle Database
Payment Database
Ride History Database