Legfree: Team Project

Zhiheng Zhang, Junhan Yang, Shaohua Zhang, Hongbin Bao, Xinhao Yang, Hyun Suk Lee, Jaskaran Singh Kawatra

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

07/11/2022

Contents

1.Abstract:

In recent years, shared-use vehicle systems have gained a lot of popularity all over the world. These systems usually consist of a fleet of vehicles that are used by different individuals throughout the day. Shared-use vehicle systems have the potential to become an important link toward building smart and sustainable cities. These systems are attractive because they can potentially reduce the cost of commuting for an average person: people do not necessarily need to buy or lease a vehicle if their usage is limited. The focus of this thesis is to discuss a framework for developing a self-sustaining system capable of managing consumer requirements. Through an E-Vehicle Share System, a consumer can rent a vehicle at any location in the city if there is a working vehicle available at that location, return a vehicle to any location in the city after the consumer is done with its use, and report any defects the user may encounter during vehicle usage. Further, to realize the system in its full complexity, it is also capable of producing an operator interface that will allow the operator to track the location of all vehicles in the city, charge a vehicle when its battery has been depleted, repair vehicles that have been reported as defective by the user, and move vehicles around the city as and when required. This thesis will provide an overview of the existing software surveyed to implement the system, discuss suitable approaches to tackle this problem, and explain the design approaches implemented as well as the motivation behind choosing a particular design idea. We will also be discussing the technologies used to realize this system along with the testing procedures implemented.

1.1 Chapter Structure

This chapter summarized the motivation and aims behind the development of LegFree. The remaining chapters will go into broader detail and describe the working of the application and how the aims were successfully met. The structure of the remaining chapters is given below:

- **Chapter 2:** Background research of different products in the market and user response to the solutions presented in comparison to pre-existing ways of commute.
- **Chapter 3:** This chapter will focus on describing the MoSCoW prioritization technique and outline the functional and non-functional requirements for all three users (Consumer, Operator, and Manager).
- **Chapter 4:** In this chapter, we will describe the system architecture by going into the details of the deliverable and non-deliverable components of the project. This chapter will also cover the initial project planning stage by describing the ER diagram, activity diagram, paper prototypes, and wireframes.
- **Chapter 5:** The implementation chapter will focus heavily on demonstrating the product, going through different software engineering practices that were implemented, tools and technologies used, and key features and implementation details of the product.
- **Chapter 6:** The penultimate chapter will provide an evaluation of the system by going through different forms of testing methods and evaluating the product from the perspective of both the pilot user and the final user.
- **Chapter 7:** The final chapter will summarise the report, discussing any future work that can potentially make the product better while also providing an overall reflection of the project.

2.Background

2.1 Literature

2.1.1 Economics of Vehicle Sharing

Vehicle sharing systems also have the potential to generate a vast amount of economic benefit to the city in which they are deployed: an empirical analysis in Shanghai estimated that the annual saved travel time, cost, and economic benefits from these systems was 17.665 billion min, 6.463 billion CNY (£768 million), and 15.410 billion CNY-eq (£1.8 billion eq), respectively.[1] Despite the advantages these systems offer towards building a sustainable economy, they have a massive challenge of initially onboarding fleets of vehicles at a substantial investment. These systems typically begin generating profit years after they are first released to consumers.

To generate a viable business, market demand needs to be accurately predicted. It is also important to note that after these systems have been deployed, accurate prediction of peak hours helps estimate how many vehicles should be available at a station at any given time. Although vehicle sharing systems potentially offer a viable alternative for enhancing urban mobility, they suffer from the effects of fluctuating demand that leads to severe system inefficiencies. These inefficiencies are embedded into the fabric of bike sharing because one way trips are allowed and the operator has little control over the behaviour of the user. This impedes potential users to pick up or drop off their bikes at a desired station which may lead to user dissatisfaction and eventually may result in the decline of the user base.[2]

Further, the system developed needs to offer a competitive advantage to the consumer by introducing lower prices so as to not let competition gain a higher market share. To thrive in this competitive market, it is vital for bike sharing companies and app developers to understand their competitors, the services they offer, as well as the incentives they provide to their users. These systems can be implemented in three different ways: station-based, dock less, and hybrid. Kou et al. conclude that hybrid systems when compared to station-based systems offer a potential benefit of $734 per day in an urban city like Chicago.[3]

[1] Kun Gao et al., "Quantifying Economic Benefits from Free-Floating Bike-Sharing Systems: A Trip-Level Inference Approach and City-Scale Analysis," *Transportation Research Part A: Policy and Practice* 144 (February 1, 2021): 89–103, https://doi.org/10.1016/j.tra.2020.12.009.

[2] Robert Regue and Will Recker, "Proactive Vehicle Routing with Inferred Demand to Solve the Bikesharing Rebalancing Problem," *Transportation Research Part E: Logistics and Transportation Review* 72 (December 1, 2014): 192–209, https://doi.org/10.1016/j.tre.2014.10.005.

[3] Zhaoyu Kou and Hua Cai, "Comparing the Performance of Different Types of Bike Share Systems," *Transportation Research Part D: Transport and Environment* 94 (May 1, 2021): 102823, https://doi.org/10.1016/j.trd.2021.102823.

2.1.2 Carbon Emission Impact Due to Vehicle Sharing

It has been estimated that global warming caused due to carbon dioxide emissions will raise the overall temperature of the planet by around 2°C if the current emission trends continue. This will lead to a lot of increasingly unstable weather patterns and rise in sea levels. To alleviate the impending challenge we face, bike share systems promise to reduce the emissions by the transport sector significantly if major cities shift to this alternative.

In a case study of Just Eat Cycles in Edinburgh, D'Almeida et al conclude that the environmental impact from a single such bike sharing system can amount to 200 tonnes of reduced emissions. [4] If these systems were to be incorporated into every major city across the world, it would on average amount to 102.4 kilo tonnes of reduced CO2 emissions. These savings don't take into consideration the impact these systems would have on user behaviour and traffic patterns. Accurate metrics taking these factors into account will produce an even bigger estimate of the potential benefits of implementing these systems.

In another study by Zhang et al, a big data based analysis reveals that bike sharing in Shanghai saved 8,358 tonnes of petrol and decreased CO2 and NOx emissions by 25,240 and 64 tonnes respectively.[5] The benefits observed are directly correlated to the population density of a region. Although the estimates were made using the data from one bike sharing company, they give an idea for how these systems in general have an impact on the emission dynamics of a city.

2.1.3 User Perspective on Vehicle Sharing Systems

After World War-II, increased incomes and lower production costs resulted in deep penetration of personal cars in middle class households of developed countries. Wu et al conclude that household vehicle ownership behaviour is governed not only by economic considerations but also by psychological and sociological factors.[6] In previous studies, Koppelman et al explored interrelationships among perceptions, feelings, preferences, and choice using a factor analysis method. They determined that attitude variables other than perceptions of mode performance influence a travellers' choice for travel modes.[7]

However, the attitude shift is evident in light of different surveys conducted in different cities around the world. In a survey conducted by Shaheen et al in the United States, it was found that members who participate in vehicle sharing systems usually have slightly higher incomes, are younger, and tend to be more educated than the general population in the demographic region.[8] However, this strongly depends on the sample of demographic data being used as is evident in **Table 2 (Table 3 Distribution of Respondent and City Demographics)**

[4] Léa D'Almeida, Tom Rye, and Francesco Pomponi, "Emissions Assessment of Bike Sharing Schemes: The Case of Just Eat Cycles in Edinburgh, UK," *Sustainable Cities and Society* 71 (August 1, 2021): 103012, https://doi.org/10.1016/j.scs.2021.103012.

[5] Yongping Zhang and Zhifu Mi, "Environmental Benefits of Bike Sharing: A Big Data-Based Analysis," *Applied Energy* 220 (June 15, 2018): 296–301, https://doi.org/10.1016/j.apenergy.2018.03.101.

[6] Ge Wu, Toshiyuki Yamamoto, and Ryuichi Kitamura, "Vehicle Ownership Model That Incorporates the Causal Structure Underlying Attitudes Toward Vehicle Ownership," *Transportation Research Record* 1676, no. 1 (January 1, 1999): 61–67, https://doi.org/10.3141/1676-08.

[7] Franks Koppelman and Eric I Pas, "TRAVEL-CHOICE BEHAVIOR: MODELS OF PERCEPTIONS, FEELINGS, PREFERENCE, AND CHOICE," *Transportation Research Record*, n.d., 8.

[8] Susan Shaheen, Elliot Martin, and Adam Cohen, "Public Bikesharing and Modal Shift Behavior: A Comparative Study of Early Bikesharing Systems in North America," December 1, 2013, https://doi.org/10.14257/ijt.2013.1.1.03.

A research report on the adoption of e-scooters published by Populus[9] shows that 70% of the citizens of major cities in the United States expressed a positive attitude towards micro mobility solutions. People generally consider these to be a great alternative to owning a personal vehicle. These studies have demonstrated that people in general are interested in pursuing unusual modes of mobility in favour of moving towards a more sustainable future.

As the response to such systems is positive in general, it is important for governments to be onboard for integrating them into wider city development projects. Cities that develop sustainable mobility solutions will not only have a more interconnected transportation system, but will also be subject to higher economic benefit.

2.1.4 Traffic Congestion Alleviation

Cities which tend to have higher population densities are more prone to traffic congestion. Reducing the overall number of vehicles on the road is therefore, the most viable solution to alleviating traffic congestion. Some of the reasons which lead to traffic congestion are: improper planning of city development, improper lane management, accidents, and illegal parking. Traffic contributes to air pollution and leads to increased frustration levels among individuals.[10]

By simulating a large-scale mobility on demand system for the city of Prague, Fiedlar et al conclude that these systems can dramatically reduce the number of vehicles needed to satisfy existing transportation demand.[11] Further, for Capital Bike Share based in Washington DC, Hamilton and Wichman point out that the existence of bike share stations in a block reduces traffic congestion and the average treatment effect contributes to a 4% reduction.[12] Incentivizing users to also participate in MoD logistics makes the system even more efficient. This helps in managing the amount of bikes in the system to minimize the expected operating cost. Jin et al conclude that the use of dynamic pickup and return rewards can provide very substantial reductions in the operating cost, especially in operating environments with a high traffic intensity of bike returns outside the central location and relative to bike pickups in the system. [13]

---

[9] Populus, "The Micro-Mobility Revolution: The Introduction and Adoption of Electric Scooters in the United States; Populus: San Francisco, CA, USA, 2018.," accessed October 30, 2022, https://fs.hubspotusercontent00.net/hubfs/3933558/Active%20Whitepapers/Populus_MicroMobility_2018-July.pdf?__hstc=52079798.b1dfbb9c3a9b64c95d2fa6ee4d552974.1667161735409.1667161735409.1667161735409.1&__hssc=52079798.1.1667161735409&__hsfp=4219251101.

[10] Uddesh Wandhare et al., "Case Study of Traffic Congestion," accessed October 30, 2022, https://d1wqtxts1xzle7.cloudfront.net/64414685/IRJET-V7I2621-with-cover-page-v2.pdf?Expires=1667168042&Signature=E33nfCt8-kXFg6jrEZ71R3Xmd6mdTVII7sWwSVEQnmydCph34XWxlDdM7vP~KYDSL7UdvY2nhTMs12DRXJQQi0Tfu7IGjwfpX0ocaEkBBgEJYoLvuJBCX1SZJr81g7nxhWCPoNwiztziovw6Jm2YYpFBcwWaRA937Fx5j68MdeM8oE0bY5tUQ8sKj-15B5hvKOmSyaPj9BgIZp6cGPSrokc02-H3-C9j4GbuV-TuuGwTiruvTdHcl-K--huCF59Jrh47O2GmY7RG8G2Kddbmnk9Pqdyoxq7WSmyZMa5fdkZwzKSdIsjYFn1NhsLfDlwFysYQghuEDj2Q2mKPTKSD-Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

[11] David Fiedler, Michal Čáp, and Michal Čertický, "Impact of Mobility-on-Demand on Traffic Congestion: Simulation-Based Study," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, 1–6, https://doi.org/10.1109/ITSC.2017.8317830.

[12] Timothy L. Hamilton and Casey J. Wichman, "Bicycle Infrastructure and Traffic Congestion: Evidence from DC's Capital Bikeshare," *Journal of Environmental Economics and Management* 87 (January 1, 2018): 72–93, https://doi.org/10.1016/j.jeem.2017.03.007.

[13] Huan Jin et al., "Dynamic Incentive Schemes for Managing Dockless Bike-Sharing Systems," *Transportation Research Part C: Emerging Technologies* 136 (March 1, 2022): 103527, https://doi.org/10.1016/j.trc.2021.103527.

2.2 Pre-Existing Related Applications

This chapter will compare and review six of the pre-existing vehicle sharing systems. Background, strengths, and weaknesses of all the applications will also be discussed.

2.2.1 Lime

Neutron Holdings, Inc., doing business under the name Lime is a transportation company based in San Francisco, California. It runs electric scooters, electric bikes, and electric mopeds in various cities around the world. The system offers dockless vehicles that users find and unlock via a smartphone app that knows the location of available vehicles via GPS. The valuation of the company is over $1.1 billion and has a user base of over 150,000 users.[14]

Strengths:

- Lime makes use of the QR code scanning feature in their mobile application which is a user friendly initiative.
- The incentivizing scheme allows a decent compensation of £4 credit for both parties for referring the application.
- Tapping on statistics in the dashboard allows the user to gain an intuitive understanding of weekly/lifetime biking trends.
- Lime offers schemes to riders to earn compensation while riding a bike and fulfilling certain tasks.
- The history section of the application is intuitive and provides a detailed description of all the journeys.

Weaknesses:

- Reporting an accident is not a user friendly experience as you need to find the section first in the help section which does not seem natural.
- Lime has very limited coverage within the UK with stations mostly available only in Manchester, London, and Bletchley.
- It is more expensive than other alternatives costing £1 to start the service and 17p/min thereon for e scooters and 19p/min for e bikes.

2.2.2 Dott

Dott is a very new addition to the micro mobility market. It is a Dutch-French company based in Amsterdam and was founded in 2019. It operates over 30,000 shared electric scooters and electric bikes in 17 cities in Europe. Dott seems to have strong investor relations as the startup raised $29.7 million in its

[14] "Lime (Transportation Company), Wikipedia, 2022, Https://En.Wikipedia.Org/w/Index.Php?Title=Lime_(Transportation_company)&oldid=1111674878," in *Wikipedia*, September 22, 2022, https://en.wikipedia.org/w/index.php?title=Lime_(transportation_company)&oldid=1111674878.

Series A round of funding and $85 million in its Series B round of funding.[15] This is a higher funding average than most other companies which compete in Series A and Series B rounds of funding.

Strengths:

- It is a cheaper alternative than Lime in the UK costing £1 to start the service and 17p/min thereon for both e scooters and e bikes.
- Has good geolocation integration which allows for smooth real time tracking of vehicles.
- The system not only generates ride information, but also generates invoices for every ride separately. This can enable users to budget their micro mobility usage more efficiently.
- Provides a detailed section on how to ride all their vehicles with smooth and easy to understand animations.

Weaknesses:

- The incentive plan offered for referrals is not at par with Lime, offering only a 20 min free ride for both parties.
- Reporting accidents is not user friendly as the user first needs to pick a country and then call insurance or contact Dott by email.
- Empty Promo and Passes tabs which do not contribute much to efficient UI design.
- Users can only use credit cards or wallets to register with the service. An option to use debit cards has not been provided.
- As of now, it only operates in London within the United Kingdom.

2.2.3 TIER

TIER Mobility based in Berlin is an e-scooter and bicycle rental company founded in 2018. The company develops, offers and operates a digital platform for renting e-scooters, mopeds and bicycles. The company was founded in 2018 and since 2021 TIER Mobility has acquired nextbike, a company founded in 2004. With all its subsidiaries, TIER is represented in over 520 cities and 21 countries.[16]

Advantages:

- The how to ride section provides detailed insights and introduces intuitive animations for users to get familiar with the ride dynamics of different vehicles.
- TIER also implements a beginner mode in its application which allows the user to ride the vehicle with a smoother acceleration and with reduced top speeds.
- The help centre is responsive with an active team supporting the users with a usual reply time of less than a minute.
- It is the most widespread service for bike sharing in the entirety of Europe.

Disadvantages:

- There are bugs in the Bluetooth interface used to connect to bikes. It fails to load at times.
- There is no insights section to look at previous ride dynamics.

[15] "Dott (Transportation Company), Wikipedia, 2022, Https://En.Wikipedia.Org/w/Index.Php?Title=Dott_(Transportation_company)&oldid=1117391045," in *Wikipedia*, October 21, 2022, https://en.wikipedia.org/w/index.php?title=Dott_(transportation_company)&oldid=1117391045.

[16] "Tier Mobility, Wikipedia, 2022, Https://De.Wikipedia.Org/w/Index.Php?Title=Tier_Mobility&oldid=226696539," in *Wikipedia*, October 2, 2022, https://de.wikipedia.org/w/index.php?title=Tier_Mobility&oldid=226696539.

- Although a necessity, license verification is the very first step. It makes the entire experience less user-friendly.
- Incentive for referrals only offers a 5 minute free ride for both parties which is a considerably lower incentive than the ones offered by the competition.

2.2.4 Bounce

Bounce is an Indian smart mobility company, and the operator of the only dockless self drive scooter service in India. The company's fleet uses a patented keyless technology which lets a user pick up the nearest Bounce scooter and drop it off at any legal parking zone near the destination. The company offers its dockless scooters in Bengaluru and Hyderabad. Bounce claims to have started doing more than 130,000 rides a day and has clocked more than 10 million rides since its inception in 2018. Bounce is currently valued at over $520 million.[17]

Advantages:
- The system allows the user to ride specific bikes without uploading their driving license.
- Issue reporting is convenient and intuitive.
- User stories are accessible making the entire experience more engaging for the user.

Disadvantages:

- Implemented a limited map design to save resources does not allow the user to easily look for bikes in areas further away from the user's current location.
- A how to ride section has not been provided in the application.
- Booking a bike is not the most intuitive experience.
- The help section is very limited in nature and doesn't introduce real time user support.

3. REQUIREMENT ANALYSIS

3.1 MoSCoW Prioritisation Technique

Moscow technique is one of the easiest methods for requirements prioritization. It is used by stakeholders to prioritize requirements in a collaborative manner.[18] According to MoSCoW mechanism, the list of requirements can be classified into four priority categories as follows:
- **M – Must have:** In this group, requirements must be contained in the project. These requirements are critical to the success of the project.
- **S – Should have:** High priority features that are not critical to launch but are supposed to be important and of high value to users. Such requirements fill the second place on the priority list.

[17] "Bounce Scooter Share, Wikipedia, 2022, Https://En.Wikipedia.Org/w/Index.Php?Title=Bounce_Scooter_Share&oldid=1114249896," in *Wikipedia*, October 5, 2022, https://en.wikipedia.org/w/index.php?title=Bounce_Scooter_Share&oldid=1114249896.

[18] Amjad Hudaib et al., "Requirements Prioritization Techniques Comparison," *Modern Applied Science* 12 (January 14, 2018), https://doi.org/10.5539/mas.v12n2p62.

- **C – Could have:** This group contains the desirable requirements but not the necessary ones. They may or may not be of high value to the users but may be nice to have.
- **W – Won't have:** Requirements which will not be implemented in the current development process but may be implemented in the future.

3.2 Functional Requirements

Functional requirements capture the features required in a software system. This section prioritises the functional requirements of the LegFree using the MoSCoW prioritisation technique.

There are three types of users of the LegFree system, and their requirements are prioritised as below:

1.1 Customer

Must have:

- Rent a vehicle
- Return a vehicle
- Amount deduction based on travel time and type of vehicle used
- Report a vehicle as defective
- Pay any charges currently outstanding on their account

Should have:

- Setting up an account allows users to deposit a certain amount of money
- Choose the severity of the defect being reported
- Choose between different modes of payment
- Choose vehicle based on number of defects detected
- More renting options

Could have:

- Mobile application for the customer to allow iOS and Android integration
- Change UI theme based on time of day
- Add family members under the same user profile

Won't have:

- Display real time location of user
- Bike sharing incentives based on the user's activity
- Edit personal information after creating profile
- An encrypted payment gateway system

1.2 Operator

Must have:

- Track the location of all vehicles in the city
- Charge a vehicle when the battery is depleted
- Repair a vehicle when it is reported as defective
- Move vehicles to different locations around

Should have:

- Repair multiple vehicles at once
- Charge multiple vehicles at the same time if they're in the same location
- Move vehicles to different locations factoring in the capacity of a station
- Flag a station as having defective/battery depleted vehicles

Could have:

- Specify charging time for different types of vehicles
- Different charging voltage options for enabling fast charging
- Communicate with different operators for system efficiency

Won't have:

- Optimizing the moving vehicle algorithm based on frequency of usage
- Allowing operators to communicate with the customer
- Track real-time locations of vehicles using GPS

1.3 Manager

Must have:

- Ability to generate reports of vehicle activity over a defined period
- Generate visualizations for a more intuitive understanding of the vehicle activity

Should have:

- Ability to facilitate user management based on generated reports
- Select users that fulfil a certain requirement and perform analysis
- Predict revenue based on past user behaviour

Could have:

- Dashboards to look at data through different angles
- Real time data access through the database to have up-to-date information
- Modifiable querying environment to look at subsets of data

Won't have:

- User removal from database
- Addition of new type of vehicle into the system
- Adjusting the rates based on how busy the system is at a particular time

3.3 Non-Functional Requirements

These requirements represent qualities a software system must possess and can be critical to the product's success. For example, if the system requires heavy space requirements, it may fail to load on some user's devices.

Performance – The system must be responsive and must return results in a timely manner.

Security – The system must be secured with authentication and use appropriate access control mechanisms to ensure data confidentiality and integrity. For example, the system should not allow the user to unlock a bike at another station.

Availability and Scalability – The system must be accessible to users and not experience any downtime during peak usage hours. Therefore, it must be possible for the system to handle a high volume of users using cloud based solutions.

Usability – Users must be able to use and navigate through the system with ease effectively.

Reliability – The system must be reliable: if an action fails, it must handle the error responsibly and display an appropriate error message.

4. DESIGN

The design of LegFree is one of its most defining features because a good design ensures that the standards of the product are up to the mark. In the following section, a detailed description of all the design elements of LegFree is presented. A system architecture diagram conveys all the different components. Further, the behaviour of the system and its users is discussed in a peer code review exercise.

4.1 System Architecture

System architecture gives an overview of all the processes that realise the LegFree system when applied in conjunction. A system architecture serves as a key milestone in the system's lifecycle. A software system architecture comprises a collection of software and system components, connections, and constraints.[19] A system architecture is essential to development of a project as it increases the quality of the platform, helps manage its complexity, allows for better code maintainability, and makes the product scalable.[20]

4.1.1 Deliverable Components

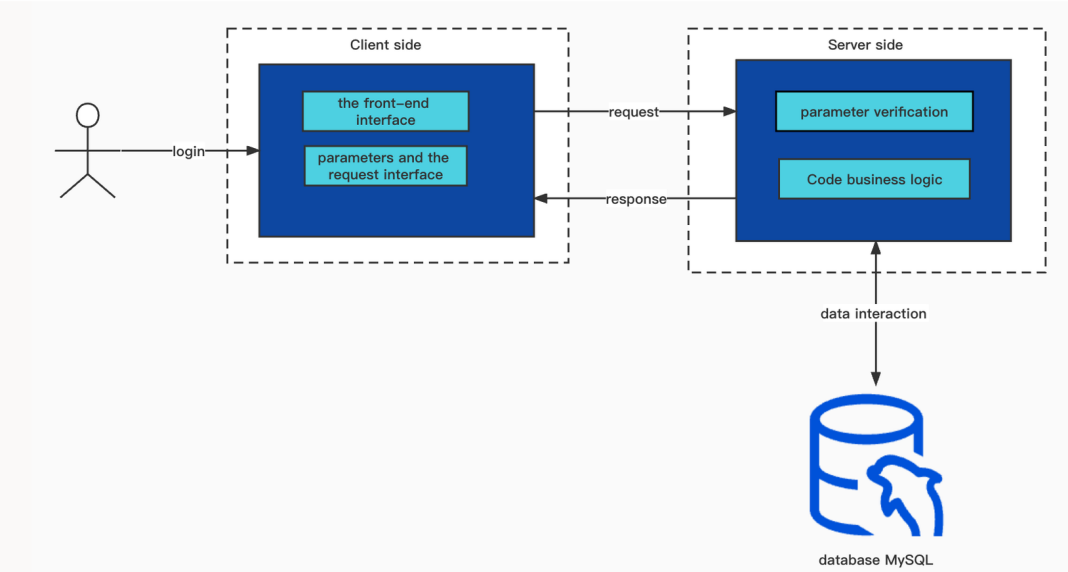


Figure 4.a: *System Architecture of LegFree*

The Frontend is used to create a channel for the users of the systems to communicate with the backend in an intuitive manner.

[19] C Gacek et al., "On the Definition of Software System Architecture," n.d., 11.

[20] apiumhub, "15 Benefits of Software Architecture You Should Know," January 2, 2021, https://apiumhub.com/tech-blog-barcelona/benefits-of-software-architecture/.

The architecture of LegFree was designed to ensure that a high quality software product is developed while meeting all its functional and non-functional requirements. A web-based architecture has been employed to allow cross-platform compatibility.

The user logs in to the client, which sends a parameter verification and request interface to the back-end through the front-end interface, and the back-end receives the request for parameter verification and returns the parameters and the front-end request interface to the back-end through a series of business logic on the back-end.

The server side can interact with the database to achieve a series of operations such as adding car records, modifying vehicle information, querying user balance and account information and order records.

4.1.2 Non Deliverable Components

The database is essential for the application and acts as a cornerstone for the entire project. A database based on MySQL has been used as it ensures that the product can be thoroughly tested. Although, not very good for implementing very large databases, it offers sufficient flexibility as it has a unique storage engine architecture which makes it faster, cheaper, and more reliable than other database systems.[21] All the information relating to the user, defects pertaining to the vehicle, information relating to the vehicle and its battery levels as well as all the orders that the user makes on the system are realised through the database.
The client-side script communicates with the server using a request-response framework. For the operator to move the vehicles around the city, it requires multiple calls to the server which can affect the response cycle if there are a number of vehicles being shifted around. To ensure that the project meets all the necessary requirements, we can utilise the integration between the VUE framework used in the development of the frontend and the Spring framework used in the development of backend. The efficiency brought about by using these two frameworks in conjunction ensures that the user doesn't receive any delays when the client-side script needs to communicate to the server.

Activity Diagram
The activity diagram prepared for LegFree illustrates

4.2 Initial Prototype

During the initial project planning stage for LegFree, an initial prototype was used to influence the design choices that would come later during the project implementation stage. The prototype shows how the system should look to the end user and is represented using wireframes.

All the wireframes and their features are as follows:

- **(Figure E.1)** shows what the login page should ideally look like to a user (customer, operator or manager).
  Customer
- **(Fig E.2)** shows what the user interface should look like after the user logs in.
- (**Fig E.3)** shows details of previously rented vehicles for the customer.
- **(Fig E.4)** shows order invoice and time rented for a particular ride.
- **(Fig E.5)** shows the details of all vehicles available at a particular station.
- **(Fig E.6)** shows the information of a particular vehicle after the user chooses a vehicle.

[21] admin, "MySQL advantages and disadvantages," W3schools, July 15, 2019, https://www.w3schools.blog/mysql-advantages-disadvantages.

- **(Fig E.7)** shows the payment screen for the user to pay for a particular trip.
- **(Fig E.8)** shows the screen where the customer can report a defect with a particular vehicle.
- **(Fig E.9)**

Operator

- **(Fig E.10)** shows the view when an operator chooses a particular station
- **(Fig E.11)** shows the list of available options when an operator picks a vehicle.
- **(Fig E.12)** change the location of the vehicle to a particular station.
- **(Fig E.13)** shows what the charging screen when an operator chooses to charge a vehicle.
- **(Fig E.14)** shows the resolution screen when the operator decides to repair a vehicle.
- **(Fig E.14)** shows the tracking screen when the operator decides to track a vehicle.

Manager

- **Fig E.15)** shows a view with all the relevant information about the LegFree system to the manager.
- **(Fig E.16)** shows the visualisations that the manager can see to get insights.

4.3 Data Model and Database Design

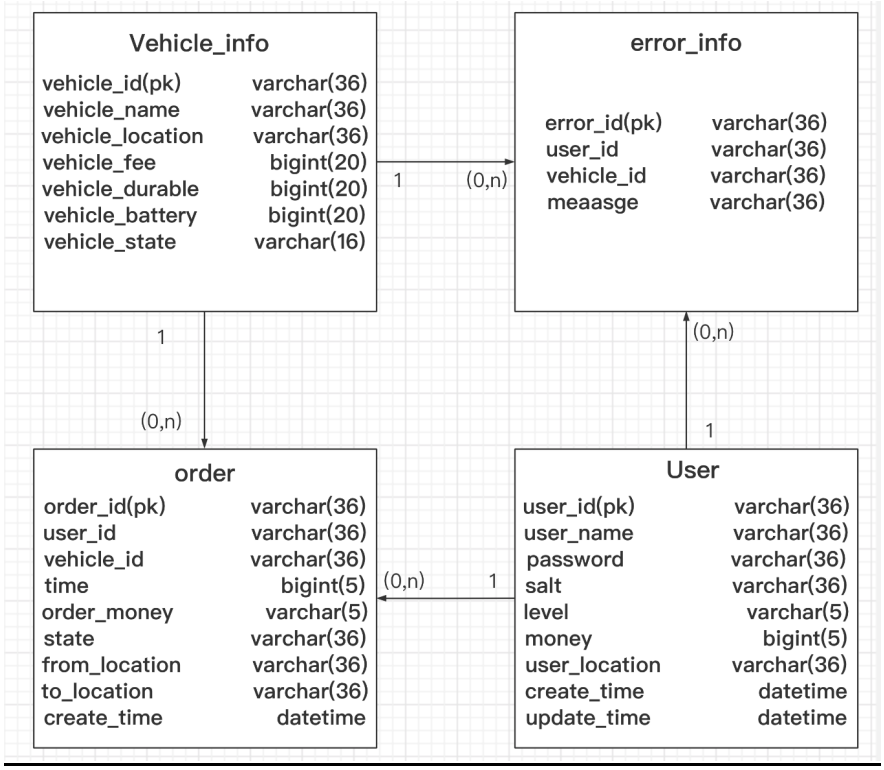


Figure 4.3: *ER Diagram for LegFree*

The ER diagram implemented for LegFree contains 4 tables as represented in Fig 4.3. The tables are connected to each other using one to many relationships to imply that different one entity in a table can interact with multiple entries from a different table.

All the tables and their attributes are described as below:

1. Vehicle_info
   Keeps a track of all the vehicles in the database.
   Attributes:
   1. vehicle_id: Primary key for the Vehicle_info table
   2. vehicle_name: Name of a particular vehicle
   3. vehicle_location: Current location of a particular vehicle
   4. vehicle_fee: Fee per hour to ride a vehicle

5. vehicle_durable: Identifier to check the durability of a vehicle
6. vehicle_battery: Identifier to check battery of a vehicle
7. vehicle_state: Identifier to check whether vehicle is normal, in use, broken or low battery

2. error_info
   Keeps track of all the error requests pertaining to a particular vehicle.
   Attributes:
   1. error_id: Primary key for error_message_info table
   2. user_id: Foreign key from User table
   3. vehicle_id: Foreign key from Vehicle_info table
   4. message: Shows defective vehicle description input by the user

3. order
   Keeps a track of all the user requests to perform actions on vehicles
   Attributes:
   1. order_id: Primary key for order table
   2. user_id: Foreign key from User table
   3. vehicle_id: Foreign key from Vehicle_info table
   4. time: Duration for which the user rent the vehicle
   5. order_money: Fee for a particular order based on rent duration of the user
   6. state: Indicates whether the amount for the order has been paid or not paid
   7. from_location: The location from where the vehicle was picked
   8. to_location: The location to where the vehicle was returned
   9. create_time: Timestamp for when the order was created

4. User
   Stores all the information pertaining to the user
   Attributes:
   1. user_id: Primary key for the User table
   2. user_name: Username for logging in to the system
   3. password: Password for logging in to the system
   4. salt: Used to encrypt the password **(Additional Functionality)**
   5. level: To specify user type between customer, operator, and manager
   6. money: Current balance of user in the wallet
   7. create_time: Timestamp for when the user was created
   8. update_time: Timestamp for when the user information changes

5. IMPLEMENTATION

5.1 Software Engineering Practices

5.1.1 Development Methodology

Waterfall development methodology has been considered for the implementation of LegFree. Waterfall follows several fixed phases: problem definition, requirement analysis, design, implementation, testing, deployment, and maintenance.
For LegFree, a waterfall approach is a better fit for the following reasons:

- The system has a set of well-defined requirements.
- The project has a fixed timeline.
- Development does not require customer involvement.

The waterfall model defines implementation and testing as two separate stages. This approach was modified to make sure that parts of the system were able to work independently when faced with edge cases. Developing the project in this fashion allows the system to work well independently as well as collectively. If some part of the planning stage is not realised or fails a test case, it can be solved without compromising the integrity of the entire project.

In summary, LegFree follows a waterfall development methodology with room for continuous adjustments and quality control in the testing and implementation stages.

5.1.2 Version Control and Repository Management

In addition to a development methodology, repository management is key in Software Engineering. The implementation of LegFree involves utilising all the resources provided by GitHub. The repository created contains all the Frontend and Backend code along with scripts, related documentation, and report drafts. It maintains a history of every file, allows for rollbacks, and provides branching and merging capabilities. Such an approach enables continuous integration and delivery.

5.1.3 Clean Code

Building long living software involves making many decisions that influence maintainability and extensibility of a system throughout its life cycle. In early stages of building a software system, the focus is often on a certain set of platform technologies and patterns. During its following lifetime, the software-system will be challenged by inevitable change of requirements, increased workload triggering performance issues, and modernization requests due to emerging technologies.[22]

To complement this approach, the implementation of LegFree aims to produce software using clean and expressive code that is easy to read and write, with few comments. Poorly written code can result in significant wastage of project resources. Clean code is a development style that leads to modular and maintainable code.

[22] Bjorn Latte, Soren Henning, and Maik Wojcieszak, "Clean Code: On the Use of Practices and Tools to Produce Maintainable Code for Long-Living Software," *Living Systems*, n.d., 4.

5.2 Tools and Technologies Used

5.2.1 Frontend

This project uses a combination of frameworks to achieve the front-end build The main technologies used to realise the frontend are: Vue, Bootstrap and Easy UI. Most of these technologies are based on CSS, HTML and JavaScript.

1. VUE framework
   VUE.js, as a lightweight framework, focuses only on the view layer and takes up very little memory. Vue.js splits various modules in a single-page application into separate components through components, and the programmer just needs to write various component tags in the parent application first and determine the properties of the components in the component tags that are passed in parameters, and then write the implementation of each component separately to complete the entire application. When using Vue to change data, there is no need to make logical code changes, only to manipulate the data.
2. Bootstrap
   Bootstrap is based on CSS, JavaScript and HTML front-end frameworks , its package comes with global CSS settings and defines the basic HTML element styles and reusable components , etc. , to facilitate the design of page layout. Using a given basic framework, there is no need to spend a lot of time to adjust the relevant accessories, This can speed up the project and give users a better visual experience.
3. EasyUI
   EasyUI is a collection of jQuery-based UI plugins, although its interface is not as effective as Bootstrap, it has powerful data interaction features that can provide the necessary functionality for interactive js applications. EasyUI library, as a relatively mature js front-end framework, provides a rich set of UI controls, including form controls, layout controls, and grid controls. Programmers don't need to write a lot of js code, and in most cases only need to write html tags to define the user interface.

5.2.2 Backend

The entire backend uses Spring Boot as the main framework for developing functionality, Shiro as the security framework for encrypting user information, and Java jdk 19 as the chosen development language.

1. Spring Boot
   Spring Boot is a new framework provided by the Pivotal team and is designed to simplify the initial build and development process of new Spring applications. The framework uses a specific approach to configuration so that developers no longer need to define sample configurations.
   Developing with Spring Boot, only a small amount of configuration is needed in the Maven and Gradle configuration files to use the required framework in the code, simplifying the original configuration to almost zero code and zero XML configuration, while the versioning of dependency packages is easily solved by Spring Boot.
2. Shiro
   Shiro can perform authentication, authorization, encryption, session management, web integration, caching, etc., and supports multiple data sources for authentication. In this project, the main use of Shiro framework for user passwords MD5 encryption . Compared to Spring Security, it is more lightweight and simpler.

5.2.3 Database

This project uses MySQL database version 8.0.30 and uses Navicat as the database management tool, which has a complete graphical user interface (GUI) to create, organize, access and share information in a safe, secure and simple way and provides an easy way to manage, design and operate the database.

5.3 Additional Functionality Implemented

There is a problem with using MD5 using Shiro, the same password generates the same hash value, if two users set the same password, then two identical values are stored in the database, which is extremely insecure, adding Salt can solve this problem to some extent. When a user provides a password for the first time (usually when registering) the system automatically sprinkles some 'spice' into this password and then in the hash, and when the user logs in, the system sprinkles the same 'spice' on the code provided by the user and then hashes it, and then compares the the hash value to determine if the password is correct. This is an additional functionality of the LegFree system.

5.4 Key Features and Implementation Details

5.4.1 Login Page

After users log in with their own information and go to the vehicle rental interface, when the user decides to rent this type of vehicle, click Rent, the order will be automatically generated, and then jump to the settlement interface. Among them, the front-end interface submits data through a post request and VehicleController will pass the parameters of UserId, vehicleId, vehicleFee, vehicleLocation and date into VehicleService to further complete the business logic operation of the "Car rental" step. The current time will also generate a unique order ID. In the Service layer, the following code accesses the DAO layer code to insert into order_info... To create a new order, set the default status of the order to "Not Paid", and store the order information in the database (refer to figure E.2 in Appendix E)

5.4.2 Payment

After the user enters the rental history, select an unpaid order and click the 'Pay' button. At this time, a pop-up window will appear on the interface. Click 'OK' to complete the payment. VehicleController after the orderId and the userId parameter collection, through the res = vehicleService. UpdateOrderInfoPaid (order_id); to call vehicleService and complete the relevant operation. First of all call the DAO layer code through VehicleService, update the order status to 'Paid' and return a boolean data to mark whether the payment was successful or not. If the payment is successful, the pop-up 'Thanks for using the vehicle! 'Popup, or say' Paying fall! ' to indicate a payment failure. Finally complete the update of the order "whether to complete payment". Next, through VehicleController vehicleService. UpdateUserAccount (user_id,

order_id); updates the account balance. In VehicleService, you would call the DAO code to operate on the database, subtract the cost of the account from the cost calculated based on the length of the car rental, and update the final cost into the user account (refer to figure E.3 in Appendix E).

5.5 Finished System

See video Demonstration, readme document and user guide in the folder

video Demonstration link:
https://youtu.be/k5rNFSg-9lA

6 EVALUATION

6.1 Testing

Project testing and evaluation is one of the most important steps in the development phase. Among them, the tester of this project focuses on two aspects: whether the project finally achieves the expected goal and whether each function can be smooth and stable. In the project test stage, the "V model water" design idea and integration testing, a specific black box test method, was used.

6.1.1 Black box testing

Another name for functional testing is black box testing. It focuses on functional requirements to determine whether the requirements are met. To put it simply, the tester needs to see the test part as a black box, regardless of the internal logic of the program, only based on the requirements of the system, to check whether the function of the program conforms to its function description. It can be flexibly applied to all levels of software testing. (https://en.wikipedia.org/wiki/Black-box_testing)

6.1.2 Integration testing

Integration testing is a type of software testing that has the highest input-output ratio compared to unit testing and stress testing because it basically covers the most common use cases. Integration testing is the combination of several software modules to test a specific function. Unlike unit testing, which tests software modules in isolation, integration testing focuses more on the interactions between different software modules. (https://en.wikipedia.org/wiki/Integration_testing) In that case, almost all systems need to integration testing to varying degrees.

6.2 Testing Process

The back-end test

1. Create a use case and initialize data
2. Run the back-end code and enter the interface you want to test in postMan
3. Use postMan to test the interface

Integration testing

1. Access the frontend interface

2. Obtain the test result according to the test case requirements and test steps (for example, login user account successfully)

3. Compare the expected result with the test result to check whether the test is successful (for example, the user can log in successfully).

Category	Action description	Feature	Pass/Fail
Login Page	User login	Login User account successfully	Pass
Vehicle List	First page for User	Show the list of vehicles and details	Pass
	The user select the vehicle and rent	Select a vehicle and click 'Rent' button. Return 'Return' pag	Pass
		The vehicle's status turn into 'In Use'	Pass
		If vehicle status was not 'In use', the popup showed up and unable to rent	Pass
	The user report vehicle which has problem	Select a vehicle and Report button. Return 'Report' page	Pass
	The user can check his own renting history	Click Renting History and return 'Return History' page	Pass
	The user logout	Logout and return login page	Pass
Rent Page	The user can check vehicle status.	Show the detail of vehicle and fee correctly	Pass
	The user return the vehicle	User type Destination and save and return.	Pass
		Return vehicle successfully and update Database	Pass
		Popup 'Thanks for using the vehicle!!'	Pass
	Return popup	Click cancel and return renting page	Pass
		Click Okay and return 'Return History' page	Pass
		Vehicle's status turn into 'Normal'	Pass
	Report at the rent page	Click 'Report Faculty' button and return report page	Pass
Return History	The user pay for the last trip	Rent history status is 'Not Paid'	Pass
		Select the renting history and click 'Pay' button	Pass
	The user select pay popup	popup 'Thanks for using the vehicle!!'	Pass
		cancel popup and return 'Return History' page	Pass
		ok popup and return 'Return History' page	Pass
		click ok button then update status to 'paid' and update Dest	Pass
Manage Money	User manage the money in own account	User can check own money	Pass
	User charge the money in own account	User can update the money	Pass
Reporting	The user can report vehicle's problem	Show user name and vehicle no correctly	Pass
		Click cancel button	Pass
		Select problem and click submit button	Pass
		Return vehicle list.	Pass
		Status of vehicle turn into 'Broken'	Pass
Login Page	Operator login	Login Operator account Successfully	Pass
Vehicle List	First page for Operator	Show the list of vehicles and details	Pass
	Operator Charging the vehicle's battery	Select vehicle and click 'Charge'	Pass
		Show charge successfully	Pass
		Battery charged to 100%	Pass
	Repair the Vehicle	Select vehicle and click 'Repair'	Pass
		Show Repair successfully	Pass
		Vehicle's status turn into 'Normal'	Pass
	Move the Vehicle	Select vehicle and click 'Move'	Pass
		Return Moving page	Pass
	Logout	Click logout button and return login page	Pass
Manager Page	Manger login	Login Manager account Successfully	Pass
	Renting data grid	Select from date and to date and search	Pass
		Show the data on the grid	Pass
	Bar chart	Chart show the correct data	Pass

Fig 5.a: *Test items and results*

Provide test prerequisites, operating procedures and expected results, and then conduct the test and compare the test results with the expected results to determine whether the test passes

6.3 Developer Evaluation:

The system has realized the required functions. As a simple car rental program, users can complete the car rental, return and fault report through simple operation. The operator can timely adjust and deal with the faulty

vehicle and the location of the vehicle. Managers can perform certain data analysis. In the functional design of the system, the system designer made a comparative analysis of the technological advantages and innovative thinking of large car rental companies such as Uber. For example, big data analysis can be used to determine the car price in a certain period of time, and discounts can be provided for people who use vehicles more often. However, in the process of coding implementation, after building the basic code framework, the technology of using advanced algorithms is often difficult to achieve, and the workload becomes a power function, which eventually leads to the system function is not realized in accordance with the high standard envisaged at the beginning. In the subsequent maintenance and upgrade process, developers can update the system functions at any time.

In the testing process, because of the design of comprehensive test cases, the system reflects the high stability and operability, so that many test projects can be completed in a short time. The final test results have basically achieved the purpose of the test.

6.4User Evaluation:

The evaluation is generally based on whether the functional requirements and non-functional requirements of the project, such as aesthetics, practicability, usability and security, meet user expectations and whether the user experience in the process of using the software is good.

Aesthetics and usability: The system uses the VUE front-end framework, although it lacks the use of a certain dynamic graphics, but through the use of simple static graphics, the operation can be more clearly displayed in front of the user, so as to facilitate the user's operation of the software. In conclusion, the interface is elegant for a small project, even if it needs further refinement.

Practicability: car rental software is now becoming an essential tool for people to travel in the region. Based on this social background, the software has realized the basic functions of vehicle rental, which basically meet the requirements of practicality, however in terms of function, it still needs to be further expanded and improved.

Security: The software uses MD5 to encrypt the user password in the database, and reasonably sets the get and set methods in the encoding process to ensure the security of user data as far as possible. In the following process, developers can further protect data by setting up firewalls and other ways.

7 Conclusion

This chapter briefly summarises the dissertation, discusses future work on LegFree, by talking about new features and further research, and reaches a conclusion after reflecting on the entire project through a final reflection.

7.1 Summary

E Vehicle Sharing Systems are proving to be a cornerstone for the advent of smart cities all over the world. These systems can provide users with a ride as and when required which in turn positively impacts the economy, the climate, as well as a city's traffic dynamics. LegFree achieves traffic improvement by allowing a customer to return vehicles at any location in the city. Moreover, integrating these systems into a city has a direct correlation with the general health of the city's population. With LegFree, one of the aims is to also improve people's physical movement using bicycles. Governments around the world are beginning to realise the importance of adding these systems to a city's infrastructure and many of them have passed legislations for mobility on demand systems to be implemented. The LegFree E Vehicle Sharing System provides a unique way for all three kinds of users to interact with the system which translates to a better user experience and allows the operations for each of the users to flow in a structured way.

The report was structured to present all the information relating to the system in an intuitive manner. First, background research and key features implemented in related products were discussed to understand the requirements. Second, the functional and non-functional requirements of the system were gathered and presented using a prioritisation technique. Next, the design of the system throughout different stages of its evolution was presented and then the implementation of the entire system was discussed and realised. Lastly, the evaluation of the system concluded that it can be used in a real world setting with minimal production errors.

7.2 Future Work

In the implementation phase of LegFree, all non-negotiable requirements of the system were implemented. Future improvements of the system will be based on feedback received during final user evaluation and on the desire to include additional features, including:

1. Adding more secure modes of payment which ensure that the user's financial data is kept secure. This would ensure that more users are comfortable adding their financial details to the LegFree system.
2. Incentivizing users to drop vehicles at a requested location will ensure that stations never run out of vehicles in an area where demand is greater than the population.
3. Allowing the users to track vehicle locations in real time using GPS will ensure that all the actions pertaining to individual users can be performed more accurately. The customer will be able to precisely estimate the nearest return station based on current location, the operator will be able to decide where to move a vehicle based on real time location of other vehicles and managers would be able to see a more accurate view of the entire fleet in operation.
4. Adding more charging options to the stations will ensure that the operator can fast-charge in demand vehicles while keeping other vehicles in a basic charging state.
5. Developing a more efficient algorithm to move vehicles based on frequency of usage will ensure that vehicles can be dynamically moved to stations based on revenue being generated through that particular station.
6. As of now, the user can only pay for one order at a time. In the future, we can let the user select all the pending orders and pay for all of them at once.

In addition to implementation of new features, this system can also be used to conduct future research. The patterns of behaviour when introduced with mobility on demand systems are different for people of different age groups, regions, ethnicities, and financial capability as discussed in background research. Understanding what causes these differences can ensure that cities all over the world with different demographics are best able to utilise the benefits of mobility on demand systems.

7.3 Reflection

This section serves to present the concluding thoughts for the project. The project constituted varied aspects of software development and allowed us to learn and implement key practices that are in accordance with the standards set within the industry. The practices taught at the University of Glasgow are reflected in the project's structure and will enable us to deliver high quality projects throughout our professional lives. There were a lot of challenges throughout the project but the faculty provided us with immense support whenever we faced a difficult challenge. This has allowed all of us to grow a lot as practitioners in our field.

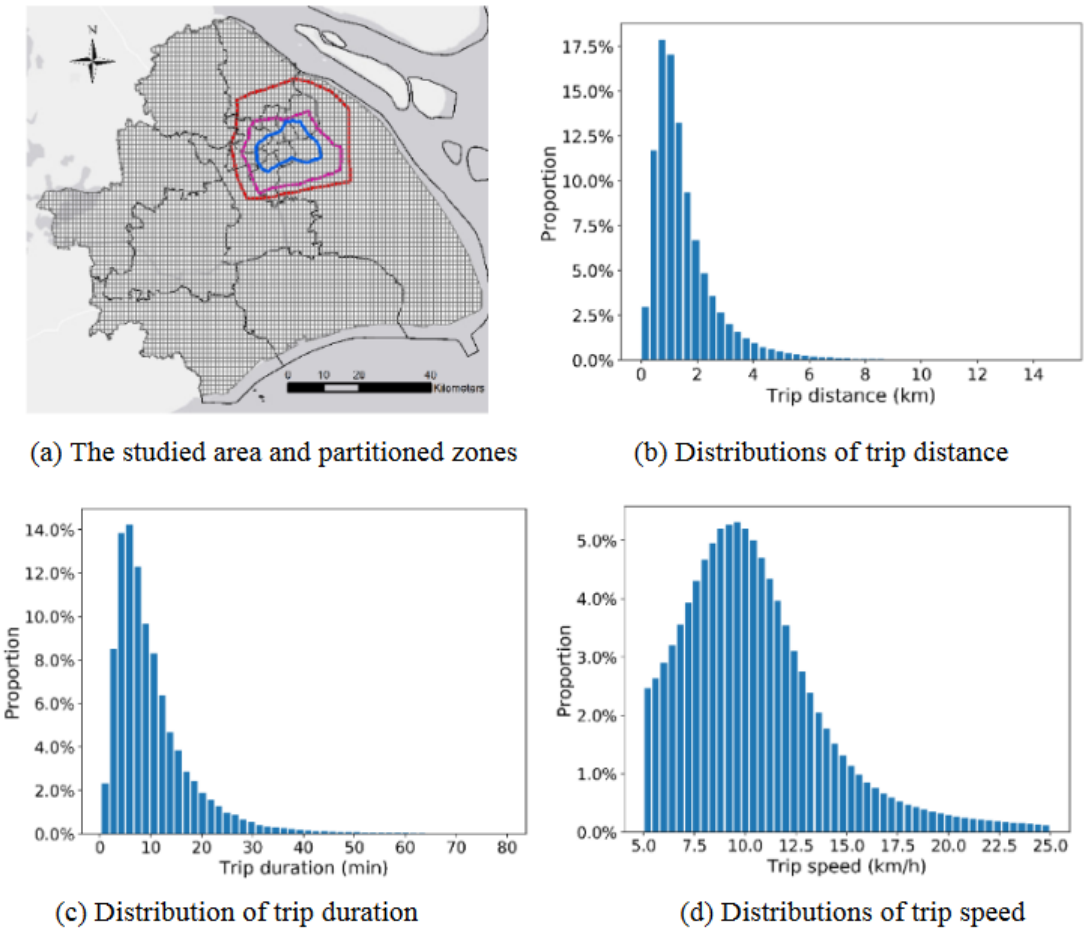
Firstly, we realised that it is of utmost importance to choose a database system which is not only easy to understand, but also has a lot of flexibility when it comes to integrating with different technologies. This allowed us to integrate all the technologies seamlessly.

Secondly, we realised the importance of communication for a project of this scale. When done effectively, communication allows for transparency and ensures that if someone is stuck on a particular problem, other team members can help. This ensures that better relations are developed among team members and the project keeps evolving at a good pace.

Lastly, we learnt the importance of delegation and accountability. A group project comprises of several requirements which cannot be implemented by one individual alone. Delegation and accountability ensure that team members take with varying levels of expertise in different areas can work on a particular sub-problem. If team members take responsibility for their work, the project can be rapidly developed while minimizing issues.

Appendix

A | Literature



(a) The studied area and partitioned zones

(b) Distributions of trip distance

(c) Distribution of trip duration

(d) Distributions of trip speed

Fig A.1: *The studied area and the distributions surveyed for the bike sharing trips from* Kun Gao et al., *"Quantifying Economic Benefits from Free-Floating Bike-Sharing Systems: A Trip-Level Inference Approach and City-Scale Analysis," Transportation Research Part A: Policy and Practice 144 (February 1, 2021)*

City	System Type	Percent of unserved trips	Excess time (minute) per trip	Average rebalance mileages in one day	Number of on-street bikes in one grid (worst case)
Los Angeles	Station-based	1.27%	2.07	51	0
	Hybrid (50%)	1.14%	0.91	51	43
	Dockless	1.28%	0.61	53	64
Philadelphia	Station-based	5.81%	1.83	38	0
	Hybrid (50%)	3.16%	0.96	37	28
	Dockless	3.76%	0.61	43	59
Chicago	Station-based	10.93%	2.38	148	0
	Hybrid (50%)	9.55%	0.92	138	155
	Dockless	10.87%	0.60	164	195

Table A.1: *Performance summary of different types of stations considered in the study from Zhaoyu Kou and Hua Cai, "Comparing the Performance of Different Types of Bike Share Systems," Transportation Research Part D: Transport and Environment 94 (May 1, 2021)*

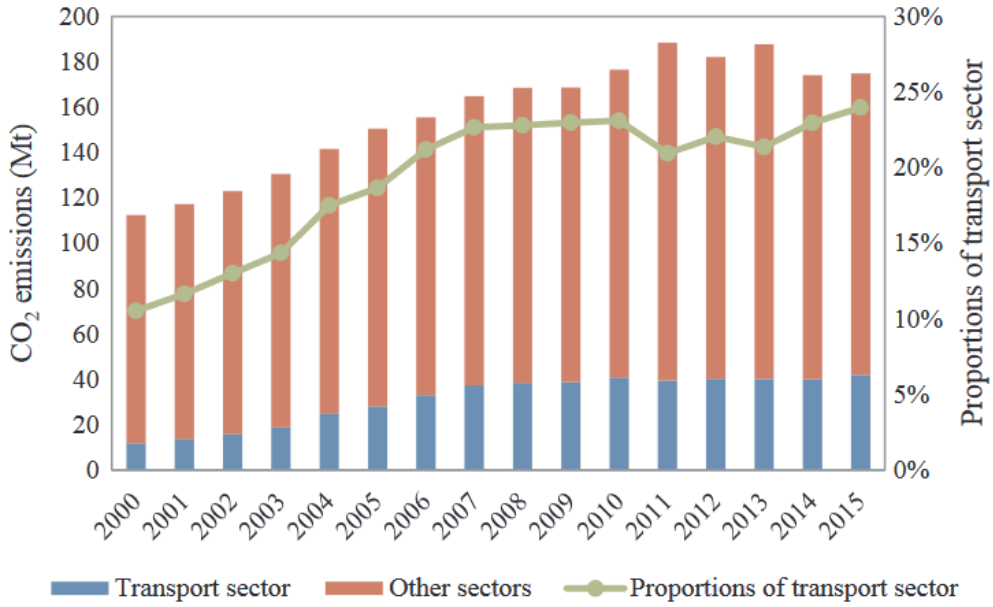


Fig A.2: *Transport sector emissions in proportion to other sectors in Shanghai between 2000 and 2015 from Yongping Zhang and Zhifu Mi, "Environmental Benefits of Bike Sharing: A Big Data-Based Analysis," Applied Energy 220 (June 15, 2018)*

HOUSEHOLD INCOME	Washington DC			Minneapolis			Montreal			Toronto		
	2011 ACS	Survey	(+/-)	2011 ACS	Survey	(+/-)	2006 Census	Survey	(+/-)	2006 Census	Survey	(+/-)
Less than $10,000	11%	4%	-	12%	5%	-	2%	5%	+	2%	3%	+
$10,000 to $14,999	4%	1%	-	6%	3%	-	2%	4%	+	2%	1%	-
$15,000 to $24,999	8%	2%	-	11%	6%	-	7%	7%	+	5%	4%	-
$25,000 to $34,999	7%	5%	-	10%	8%	-	10%	7%	-	7%	5%	-
$35,000 to $49,999	11%	14%	+	13%	14%	+	16%	15%	-	12%	9%	-
$50,000 to $74,999	15%	22%	+	17%	16%	-	23%	21%	-	20%	17%	-
$75,000 to $99,999	11%	18%	+	11%	13%	+	17%	15%	-	17%	15%	-
$100,000 to $149,999	15%	14%	-	11%	18%	+	15%	16%	+	20%	23%	+
$150,000 or more	18%	19%	+	9%	16%	+	8%	10%	+	15%	24%	+
EDUCATION	Washington DC			Minneapolis			Montreal			Toronto		
	2011 ACS	Survey	(+/-)	2011 ACS	Survey	(+/-)	2011 NHS	Survey	(+/-)	2011 NHS	Survey	(+/-)
Less than high school	13%	0%	-	12%	0%	-	12%	0%	-	11%	0%	-
High school	32%	0%	-	35%	13%	-	18%	3%	-	21%	5%	-
Technical school/Cegep	3%	5%	+	6%	4%	-	34%	14%	-	29%	5%	-
Bachelor's degree	23%	42%	+	29%	45%	+	21%	42%	+	24%	49%	+
Advanced degree	29%	53%	+	17%	39%	+	15%	41%	+	16%	40%	+
AGE	Washington DC			Minneapolis			Montreal			Toronto		
	2011 ACS	Survey	(+/-)	2011 ACS	Survey	(+/-)	2011 Census	Survey	(+/-)	2011 Census	Survey	(+/-)
18 - 24	12%	11%	-	14%	11%	-	12%	10%	-	12%	11%	-
25 - 34	27%	55%	+	27%	40%	+	21%	43%	+	19%	44%	+
35 - 44	17%	20%	+	17%	20%	+	18%	23%	+	18%	27%	+
45 - 54	15%	4%	-	17%	17%	+	17%	16%	-	19%	18%	-
55 - 64	14%	9%	-	13%	11%	-	14%	7%	-	14%	7%	-
65 years or older	14%	1%	-	13%	2%	-	19%	1%	-	18%	2%	-
RACE/ETHNICITY	Washington DC			Minneapolis			Montreal			Toronto		
	2011 ACS	Survey	(+/-)	2011 ACS	Survey	(+/-)	2011 NHS	Survey	(+/-)	2011 NHS	Survey	(+/-)
Caucasian	35%	81%	+	61%	90%	+	74%	90%	+	51%	80%	+
African-American	49%	3%	-	18%	1%	-	8%	1%	-	8%	0%	-
Hispanic/Latino	10%	5%	-	11%	1%	-	4%	3%	-	3%	1%	-
Asian/Pacific Islander	4%	7%	+	5%	5%	+	11%	3%	-	34%	16%	-
Other/Multi-Racial	2%	4%	+	5%	3%	-	3%	3%	-	4%	3%	-
GENDER	Washington DC			Minneapolis			Montreal			Toronto		
	2011 ACS	Survey	(+/-)	2011 ACS	Survey	(+/-)	2011 NHS	Survey	(+/-)	2011 NHS	Survey	(+/-)
Male	47.3%	55%	+	50.3%	55%	+	49%	57%	+	48%	70%	+
Female	52.7%	45%	-	49.7%	45%	-	51%	43%	-	52%	30%	-

Table A.2: *Distribution of Respondents and City Demographics from Shaheen et al., "Public Bikesharing and Modal Shift Behavior: A Comparative Study of Early Bikesharing Systems in North America," December 1, 2013*

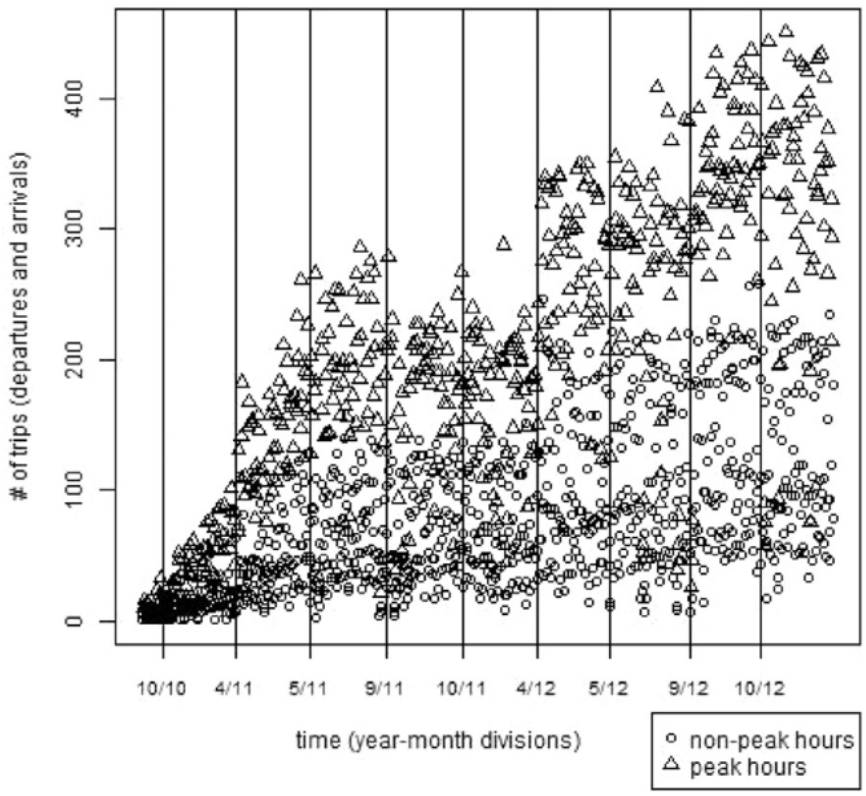


Fig A.3: *Total count of bikeshare trips made throughout sampled time of day from Timothy L et al., "Bicycle Infrastructure and Traffic Congestion: Evidence from DC's Capital Bikeshare," Journal of Environmental Economics and Management 87 (January 1, 2018)*

B | Existing Applications

Lime

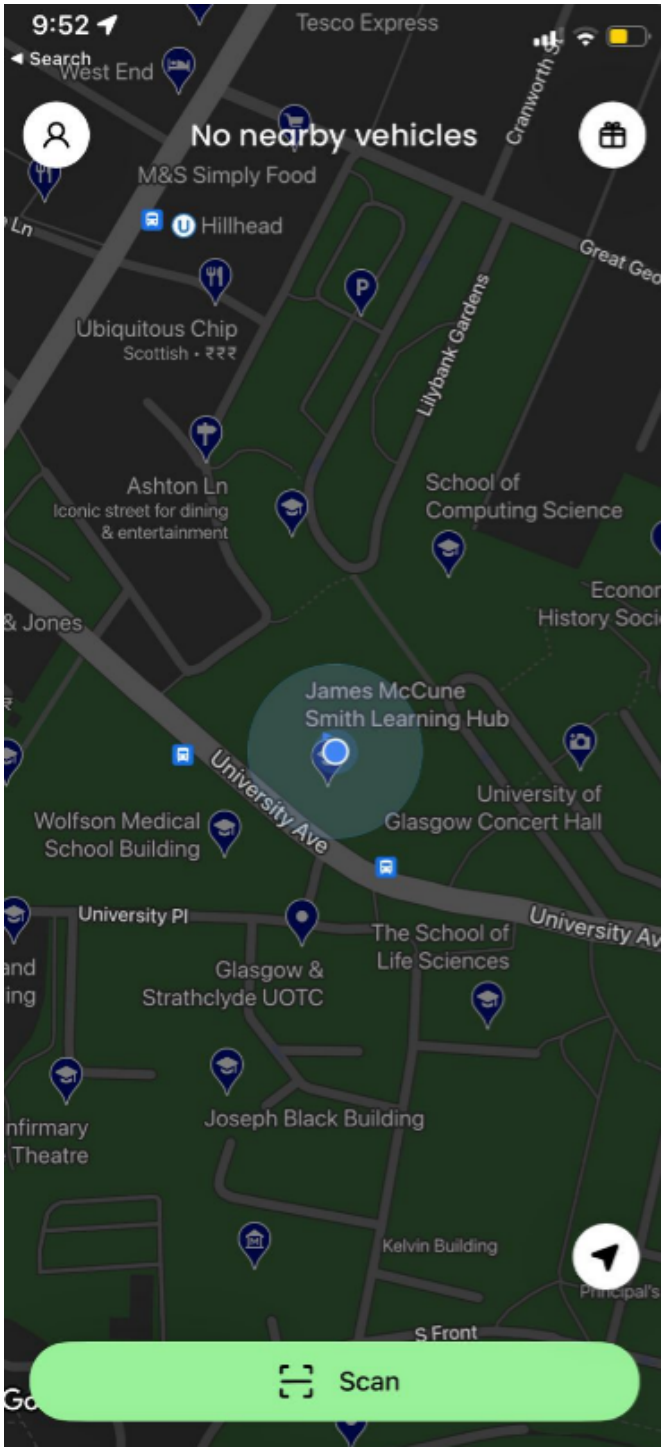


Fig B.1: *Lime Homescreen allows you to scan QR code of bike as soon as the app is opened*

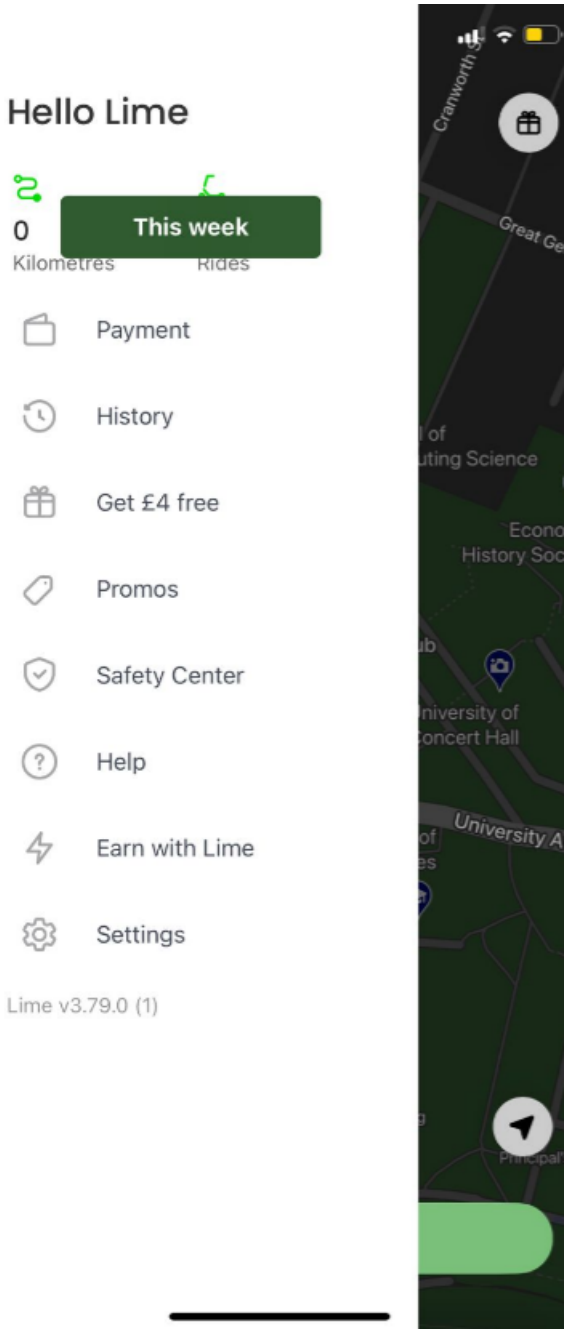


Figure B.2: *Lime Sidebar showing user statistics per week and throughout lifetime along with other options*

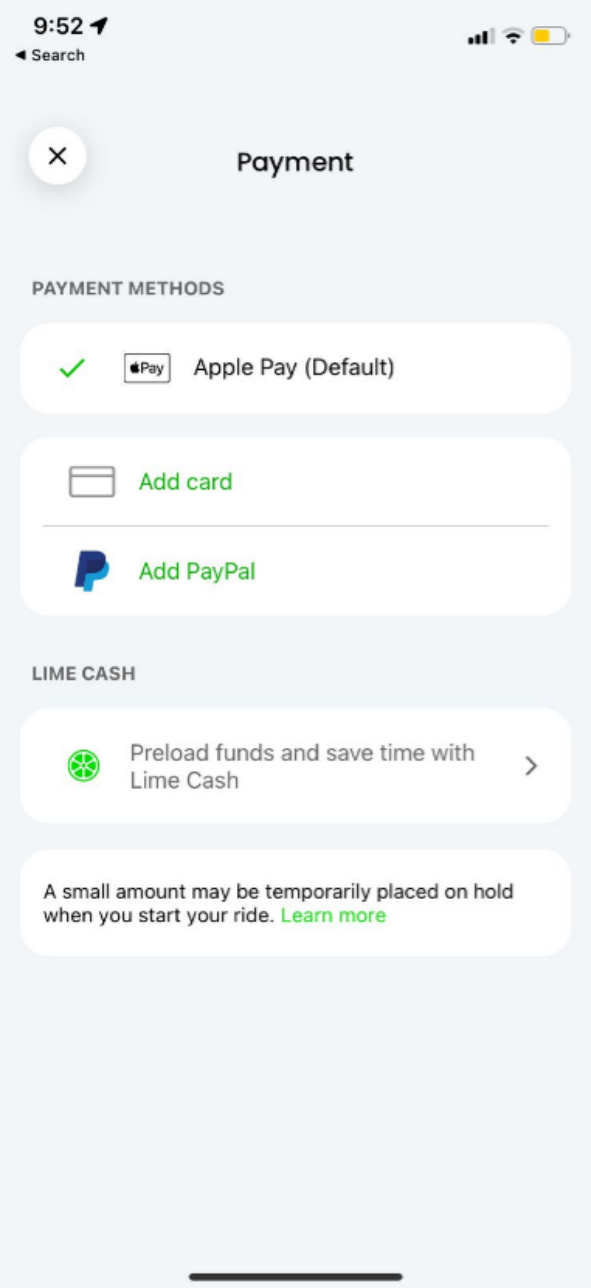


Figure B.3: *Payment options provided by Lime*

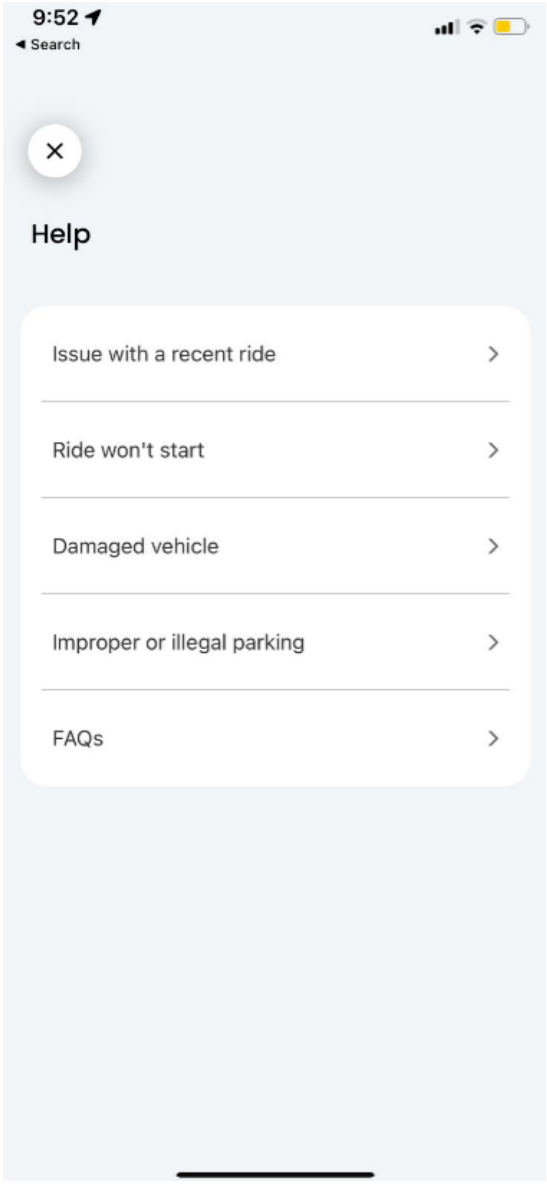


Figure B.4: *Help options provided by Lime*

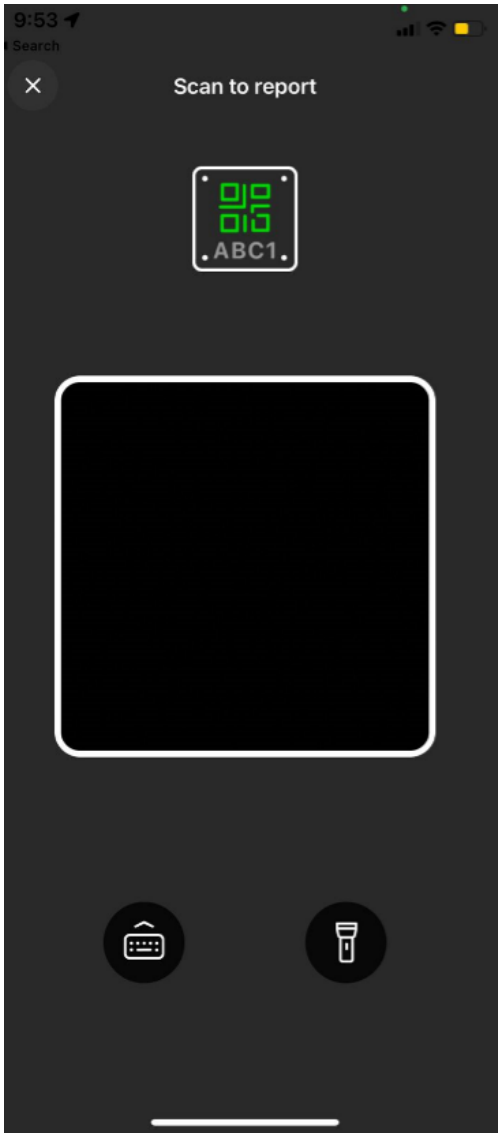


Fig B.5: *Reporting a vehicle defect by scanning the QR code*

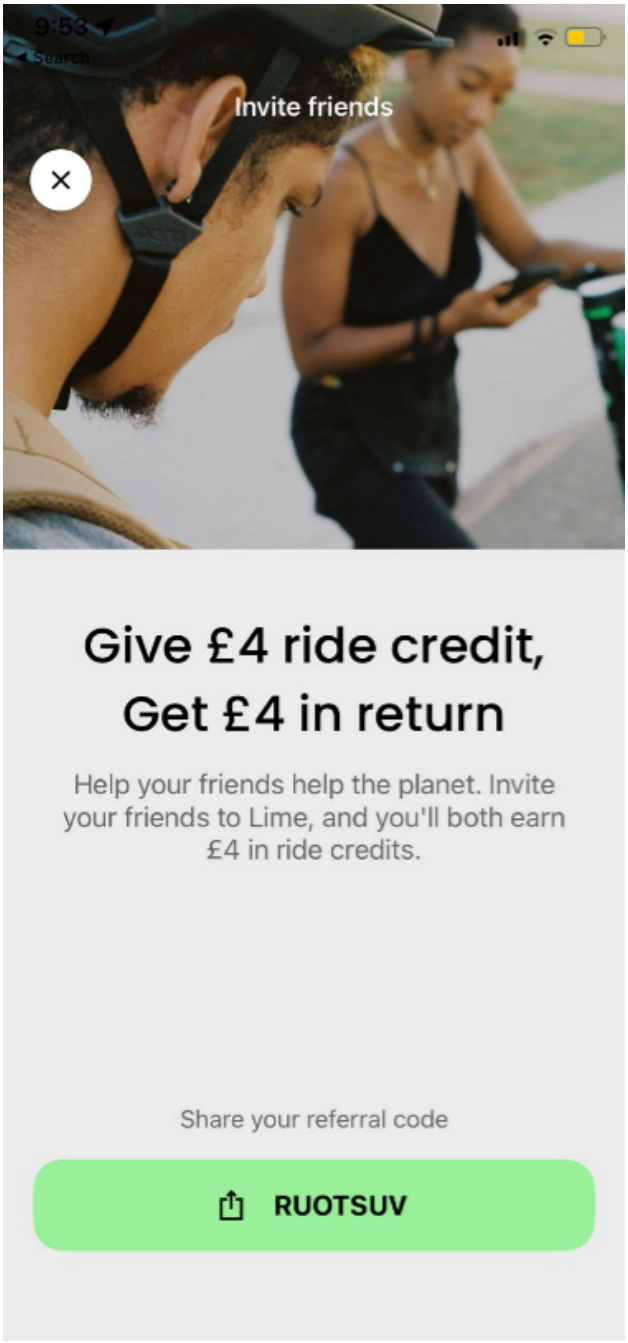


Fig B.6: *Referral incentives provided by Lime*

Dott

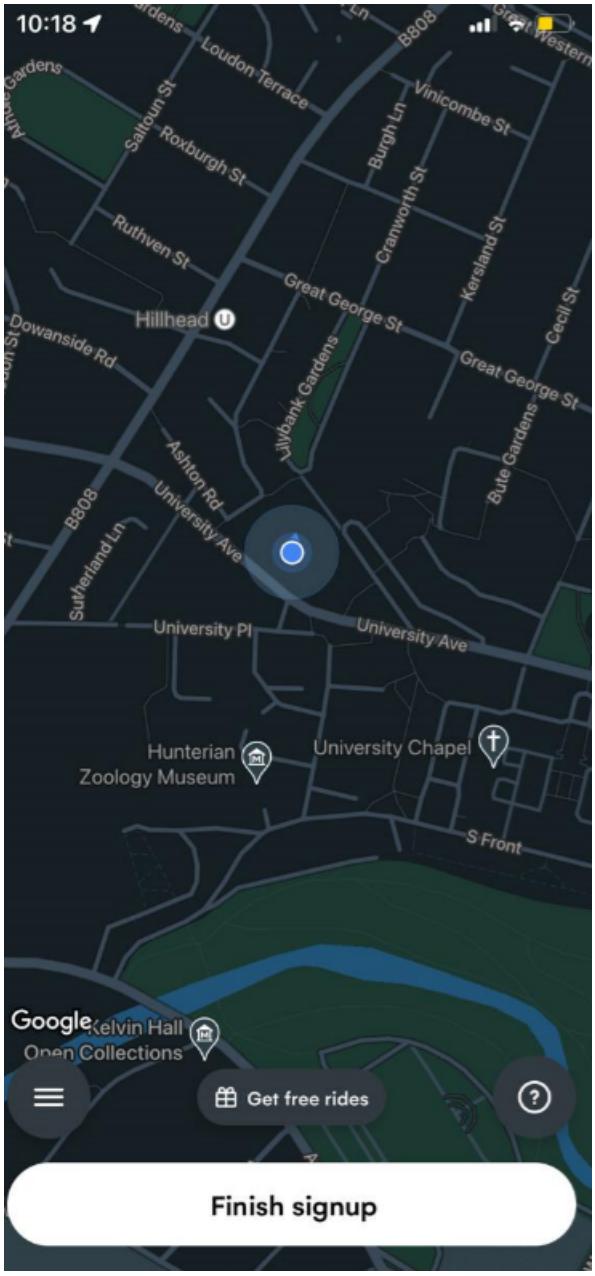


Figure B.5: *Dott homescreen design*

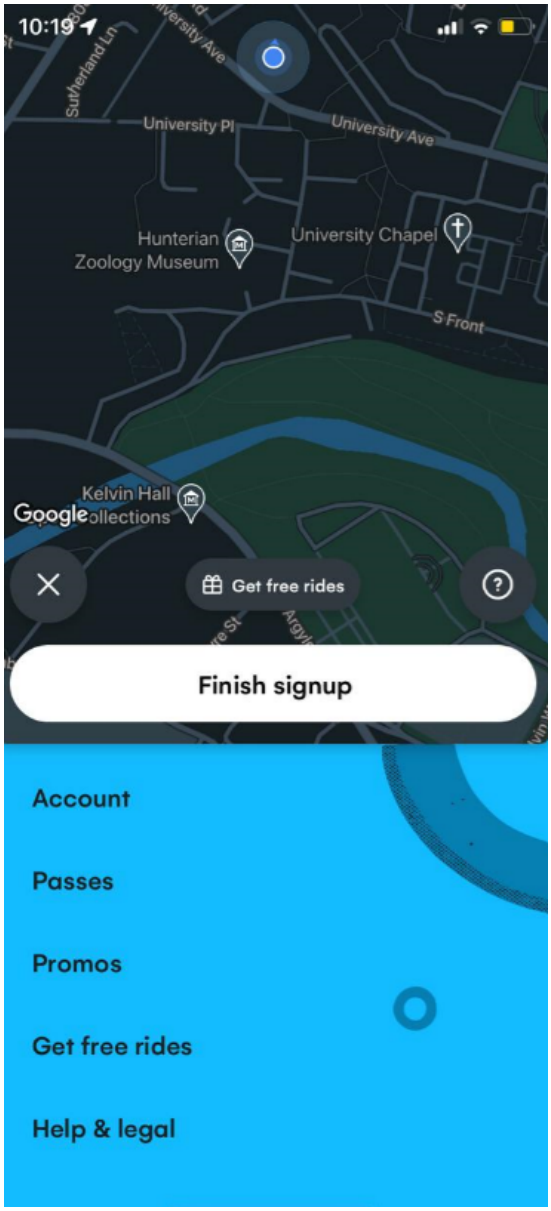


Figure B.6: *Different sidebar options in Dott*

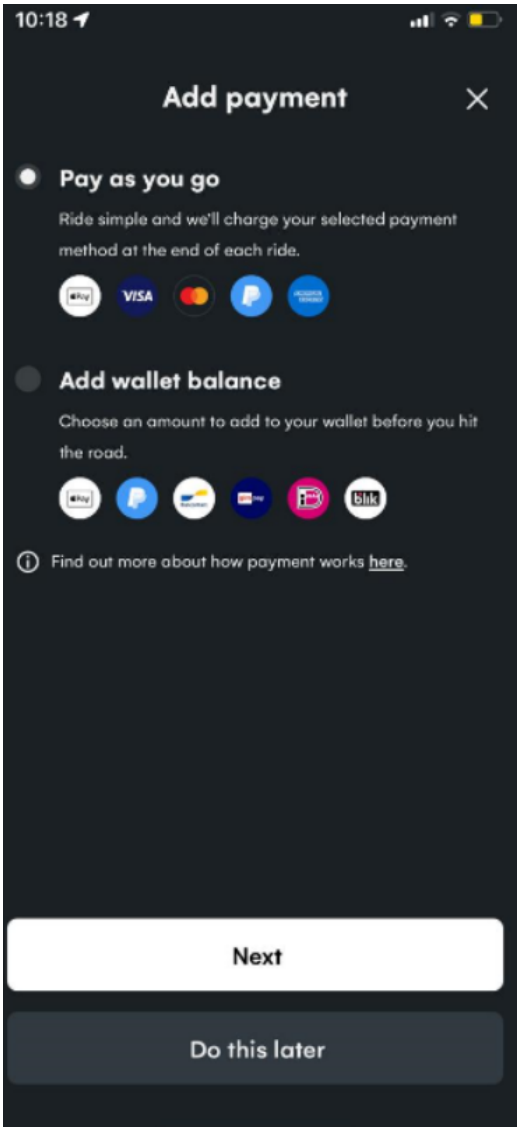


Figure B.7: *Dott payment setup interface*

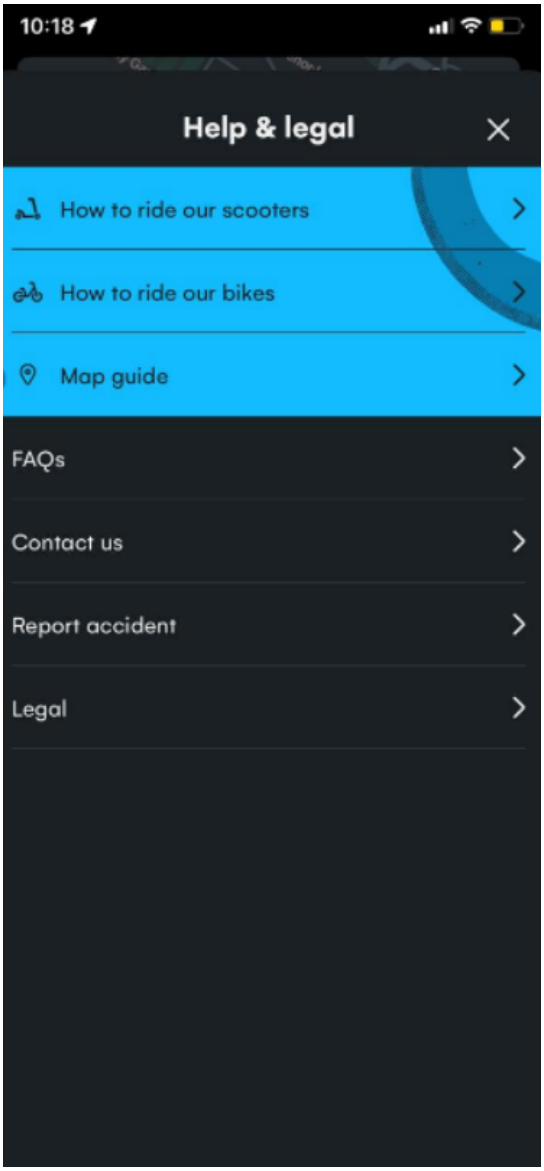


Figure B.8: *Dott help section*

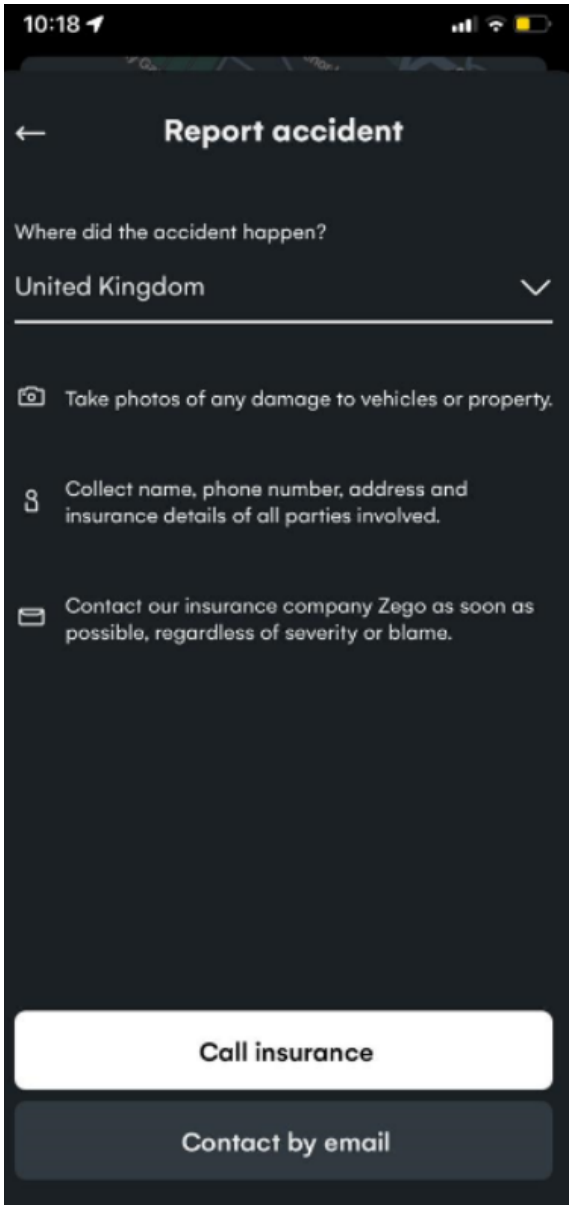


Figure B.9: *Reporting an accident with Dott*

TIER

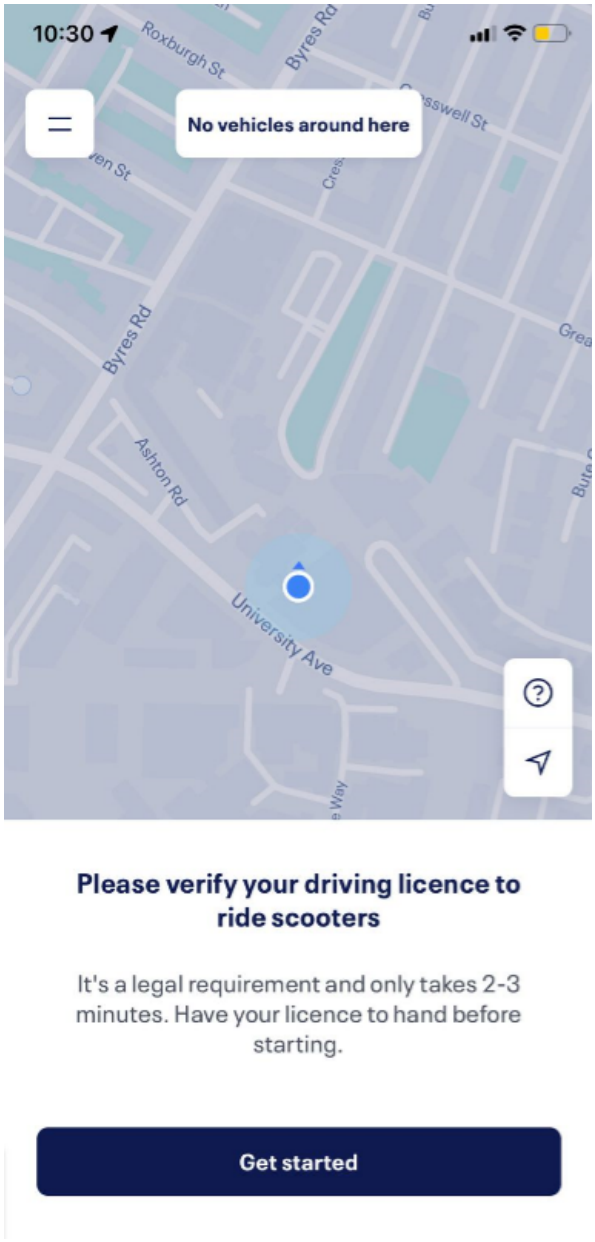


Figure B.10: *Main TIER homescreen doesn't allow an individual to start riding until their driving licence has been verified.*

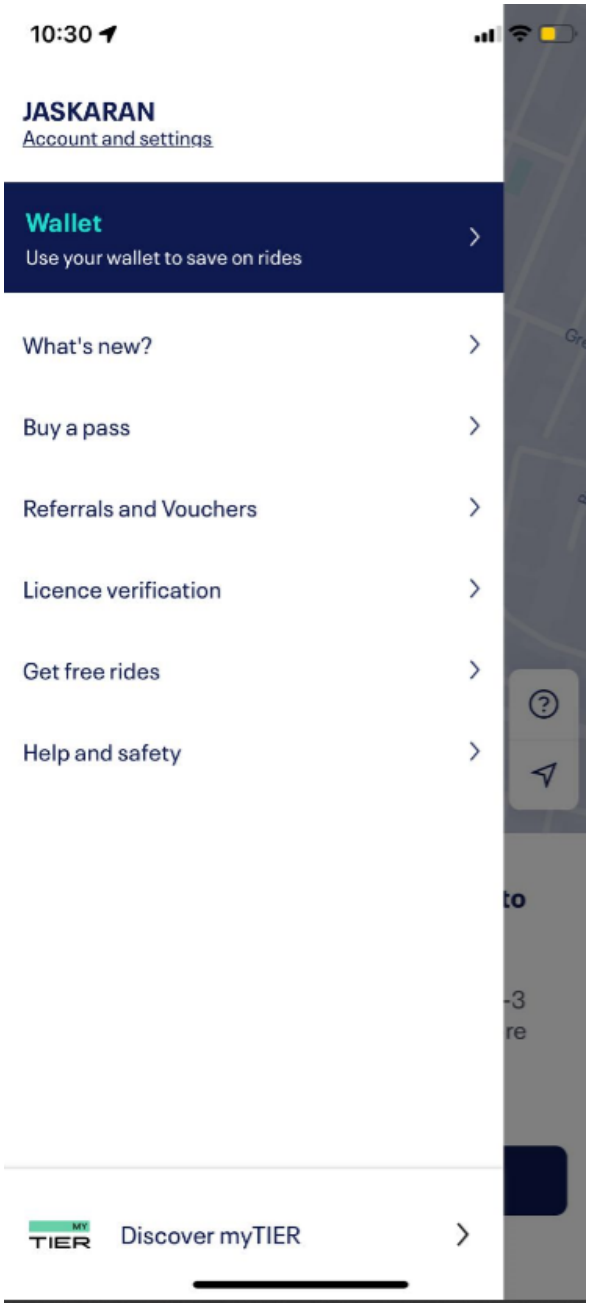


Figure B.11: *Sidebar options in TIER*

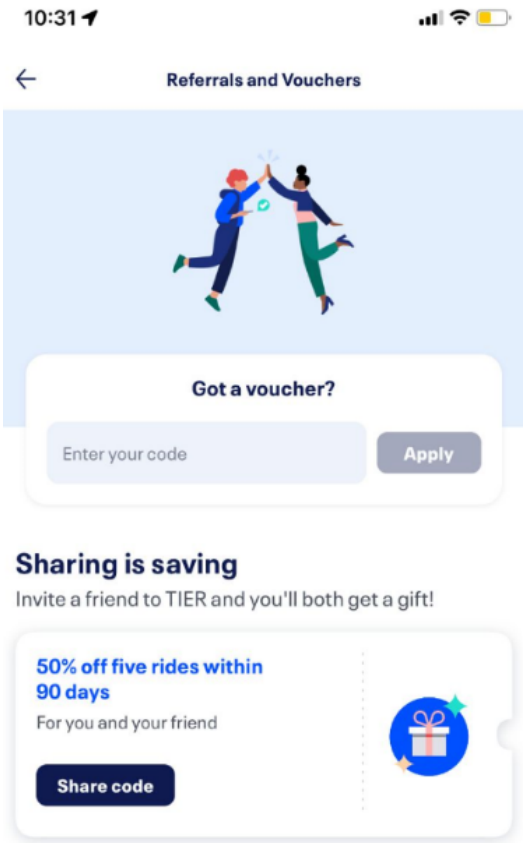


Fig B.12: *Incentive options provided by TIER*

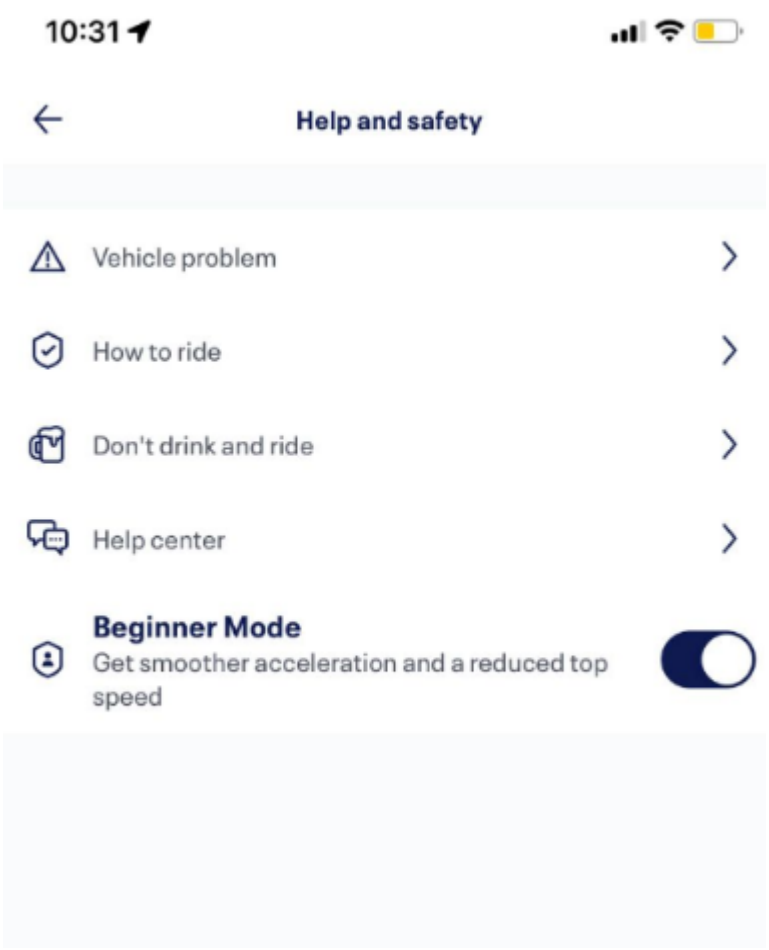


Figure B.13: *Help options provide by TIER*

Bounce

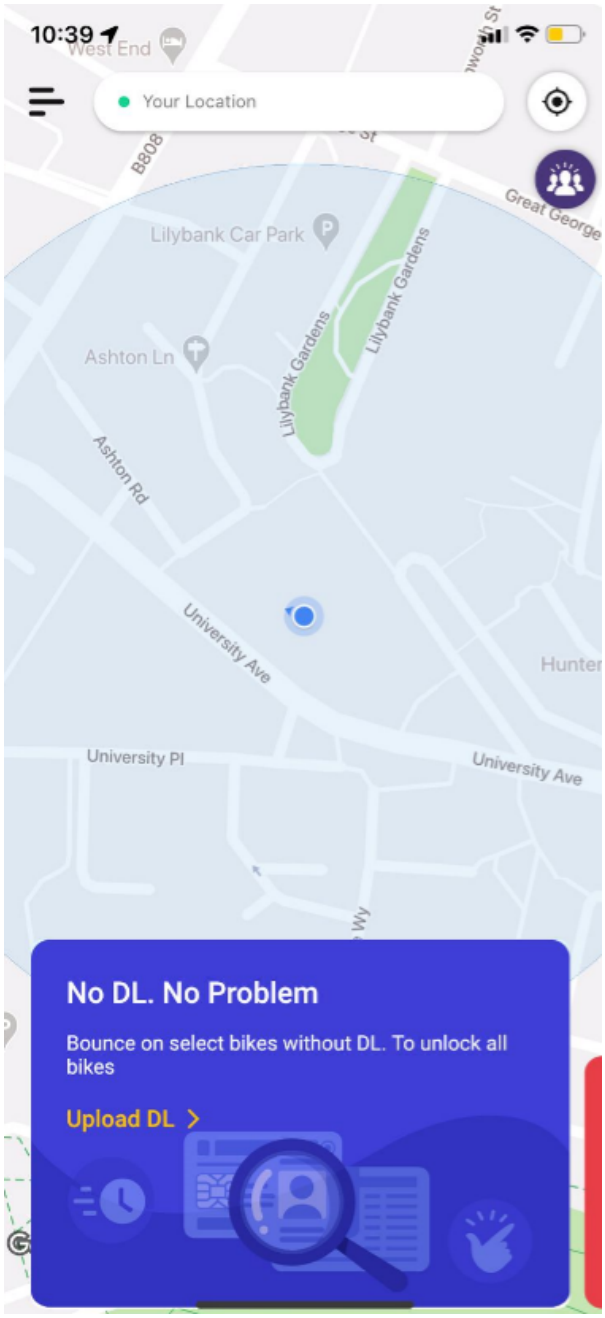


Figure B.14: Bounce homescreen

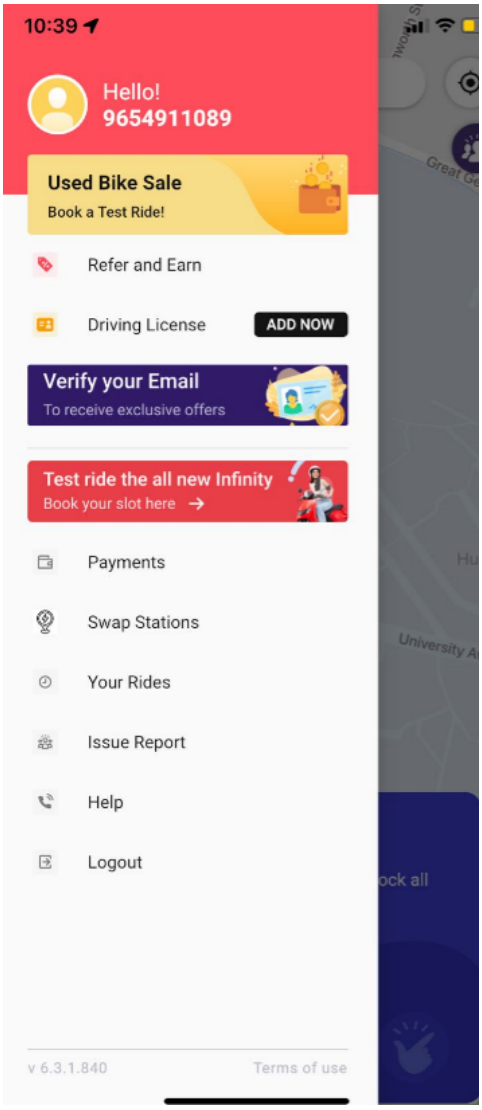


Figure B.15: *Sidebar provided by Bounce*

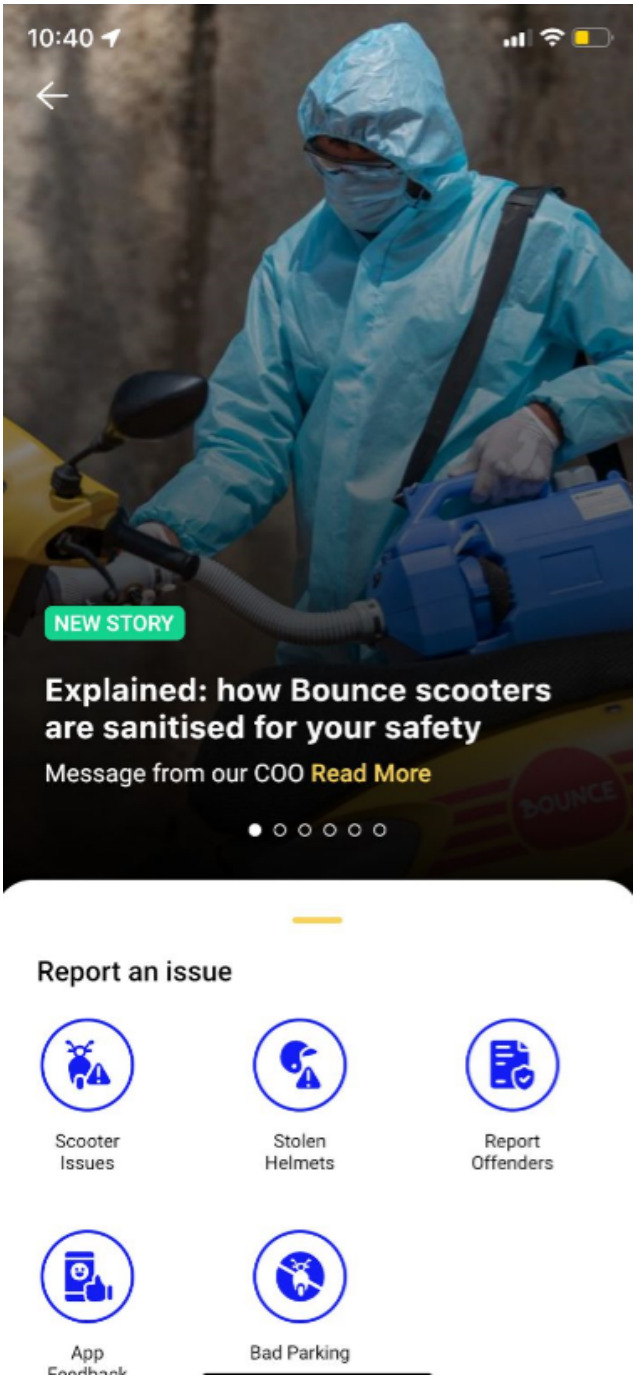


Figure B.16: *Reporting issues with Bounce*

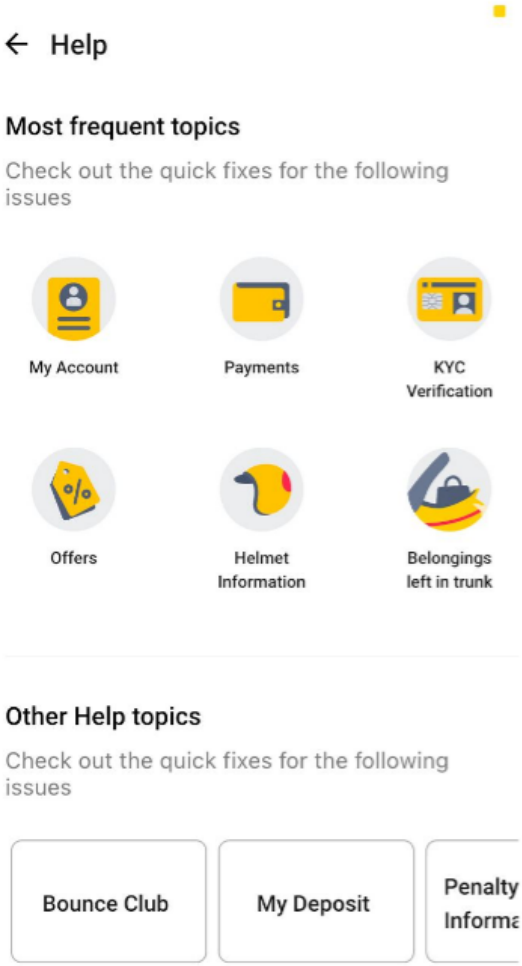


Figure B.17: *Help section provided by Bounce*

C | Activity Diagram & Use Case

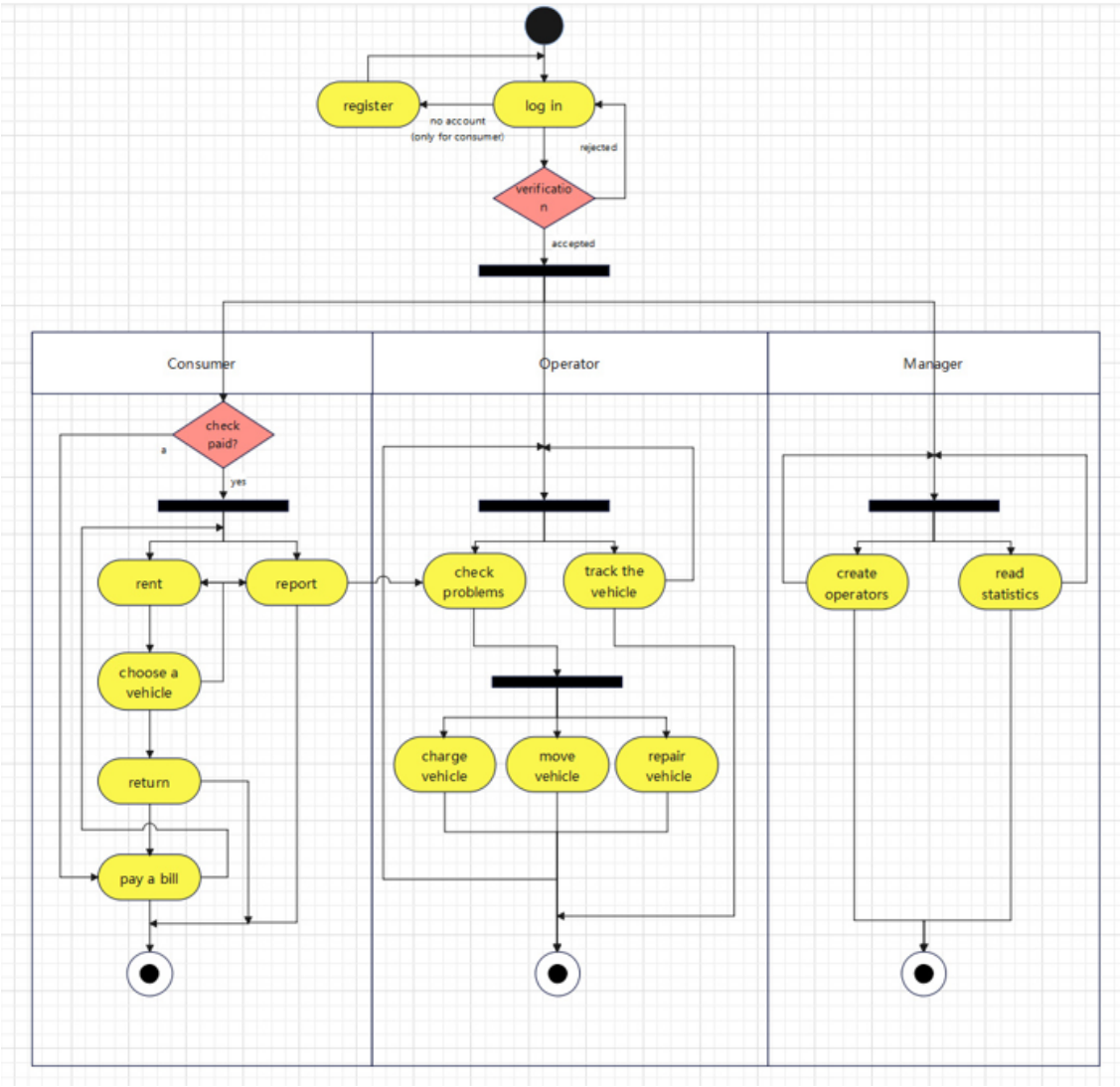


Figure C.1: Activity diagram for LegFree representing the step-wise workflow of the entire system

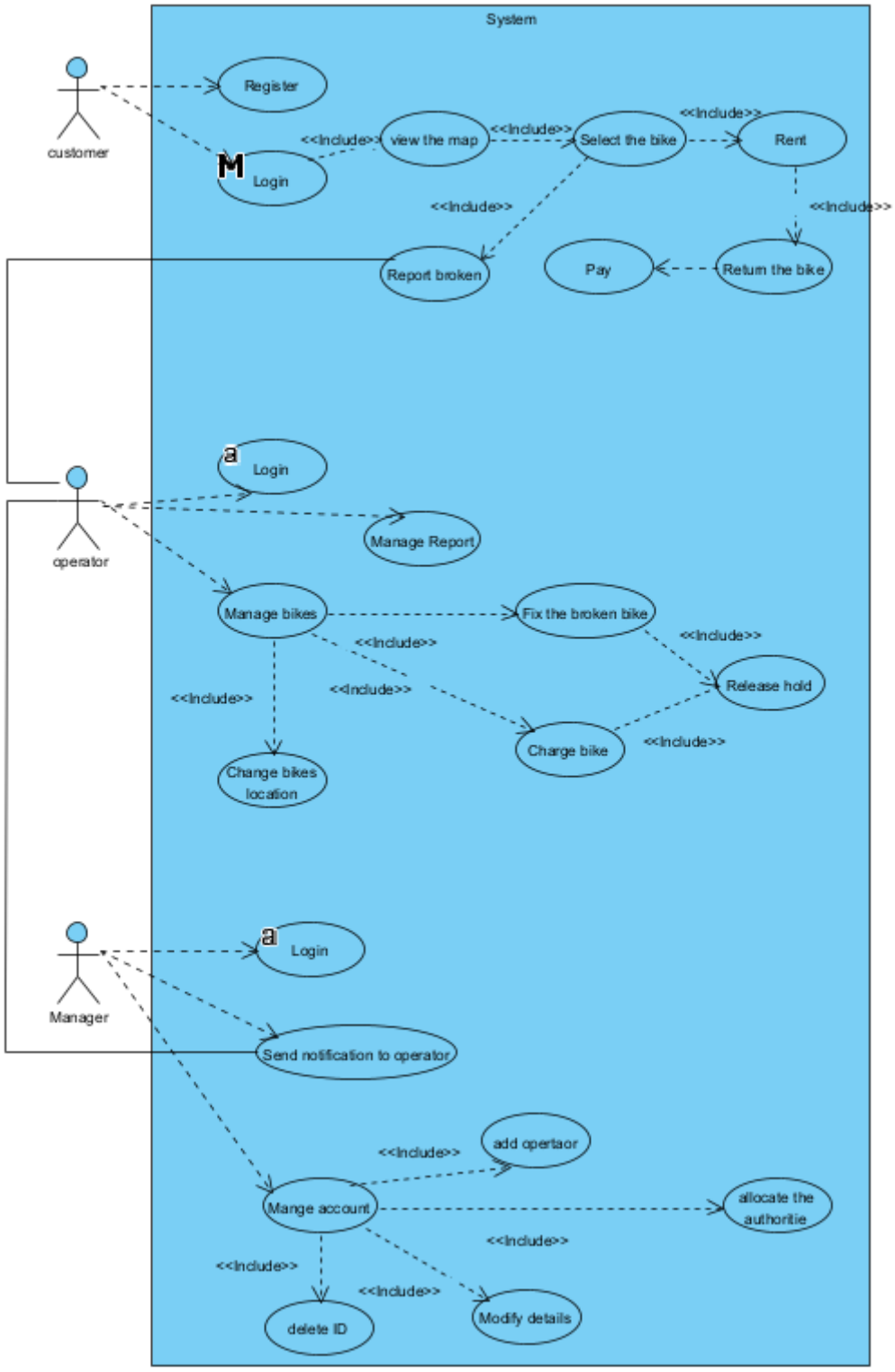


Figure C.2: Use case diagram for LegFree represent the features available to different users

D | Flow Diagrams

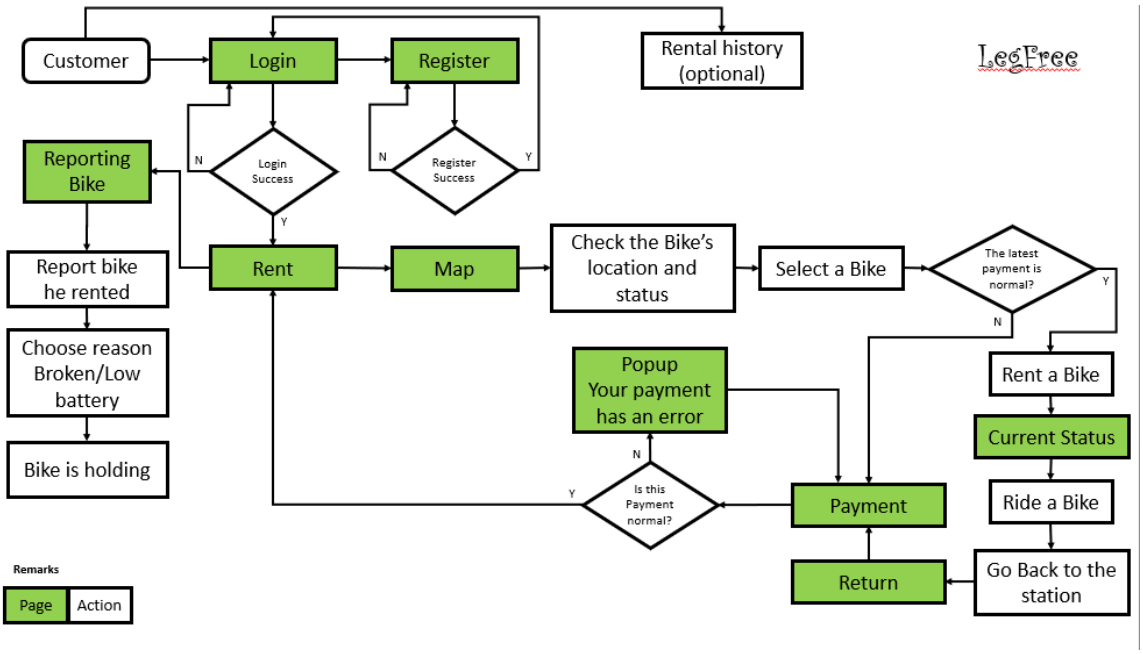


Figure D.1: *Flow diagram for Customer*

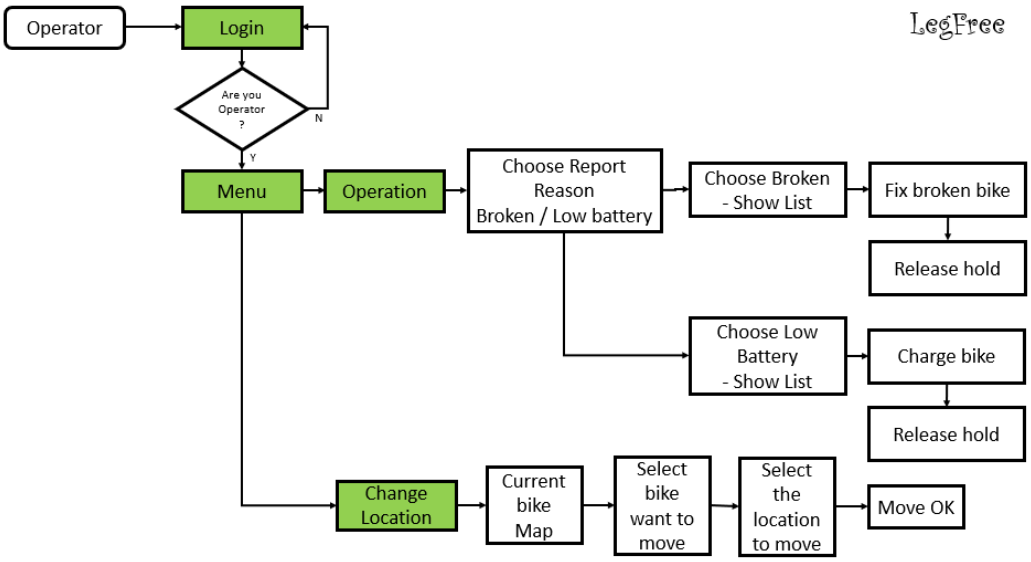


Figure D.2: *Flow diagram for Operator*

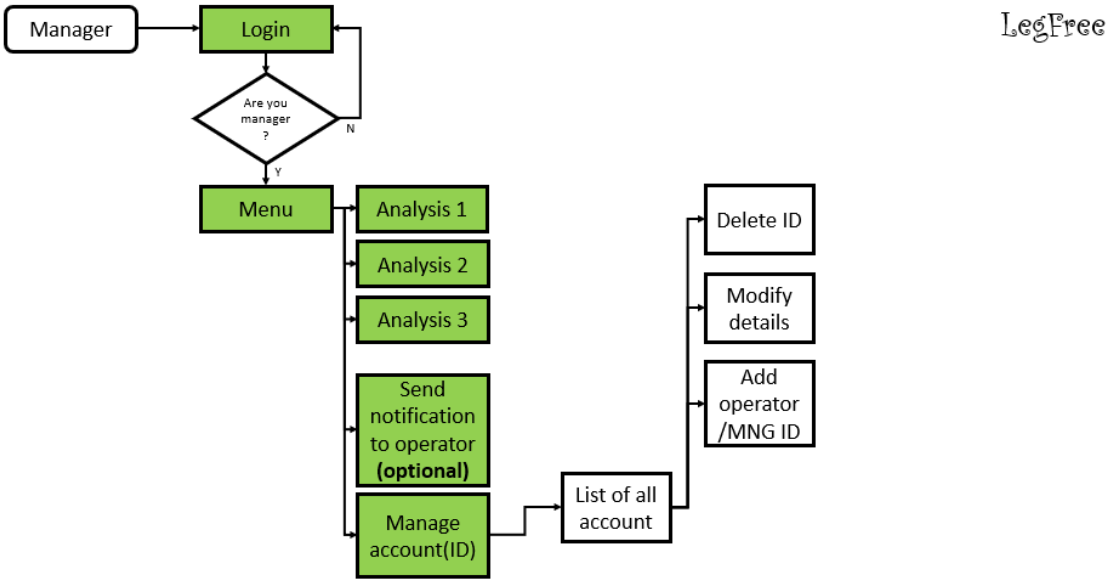


Figure D.3: *Flow diagram for Manager*

E | Wireframes

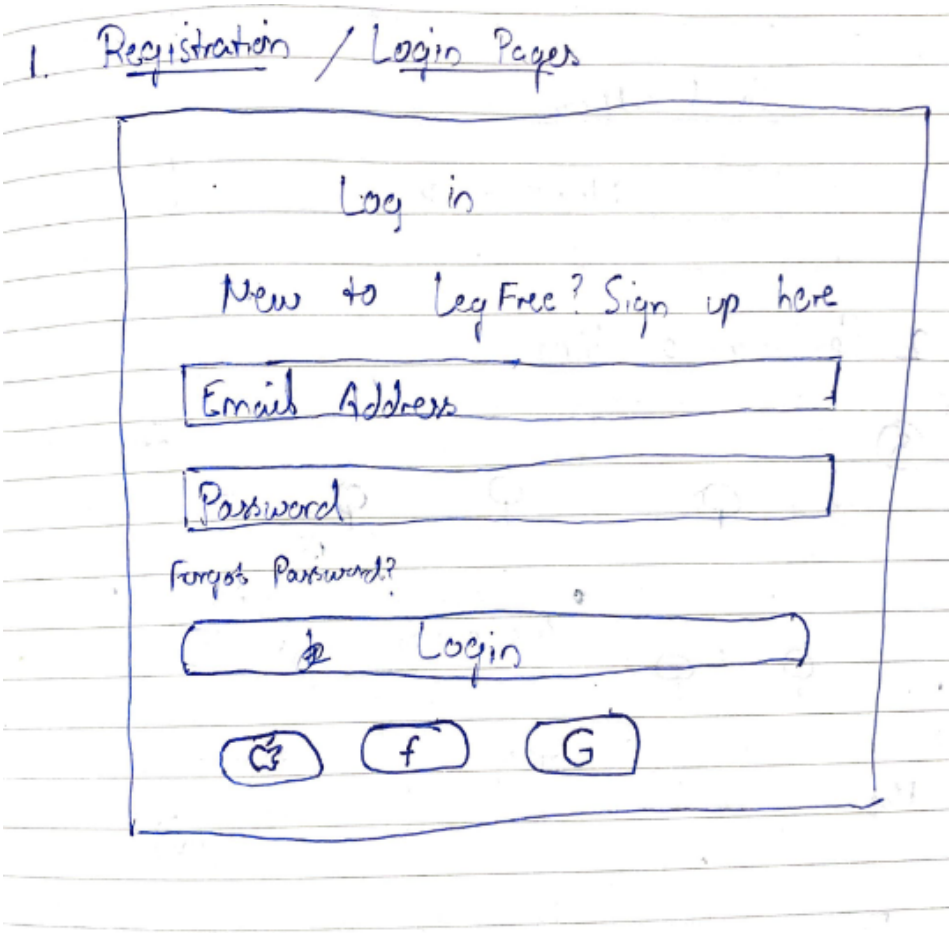


Figure E.1: *User Login Page*

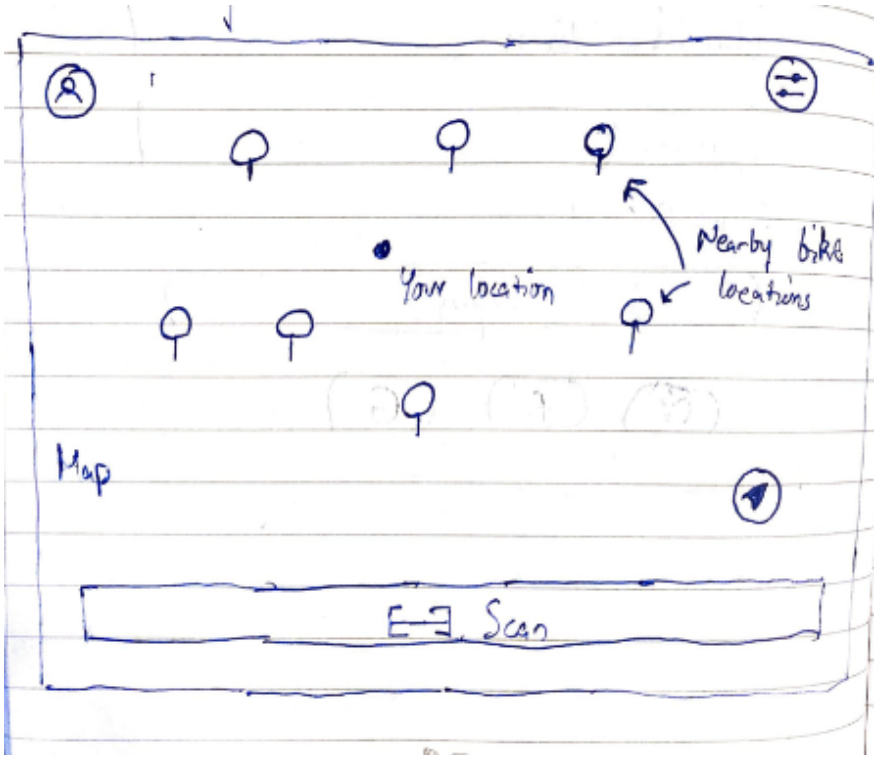


Figure E.2: *Home Screen Draft Design*

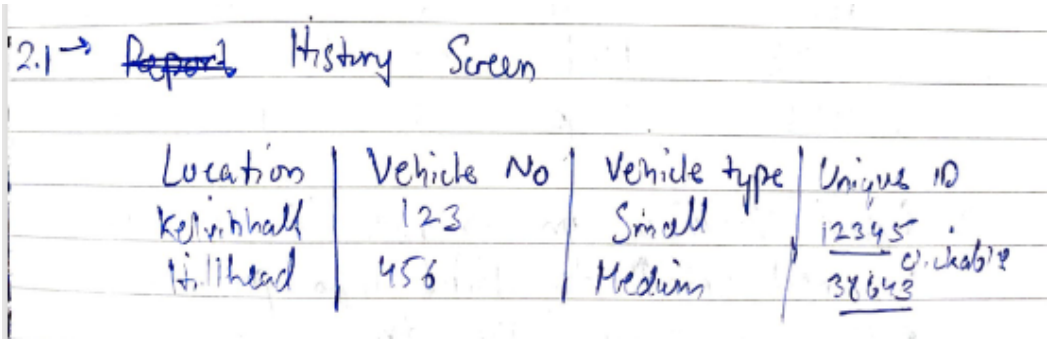


Figure E.3: *Ride History Wireframe*

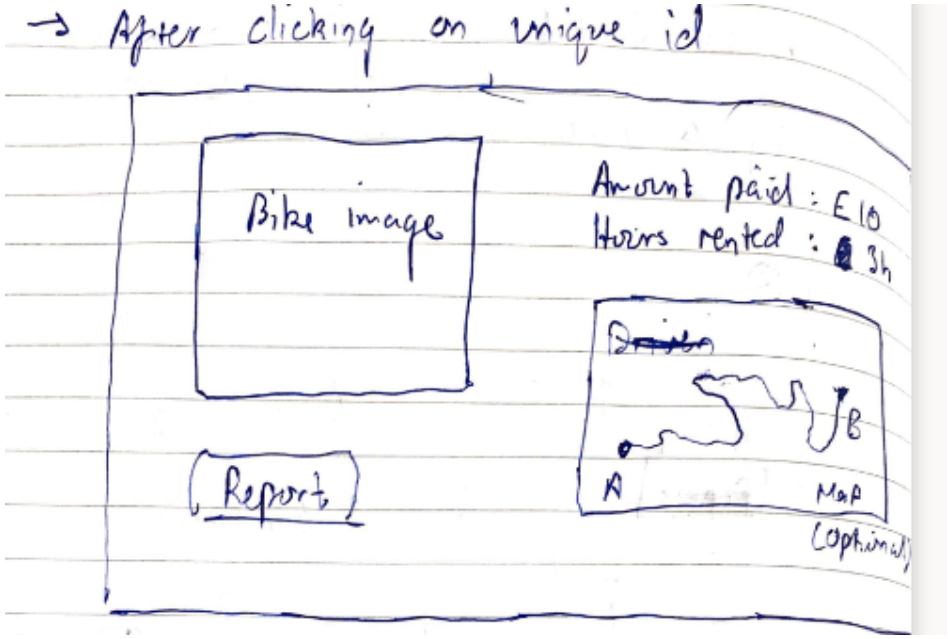


Figure E.4: *Ride information and invoice from history*

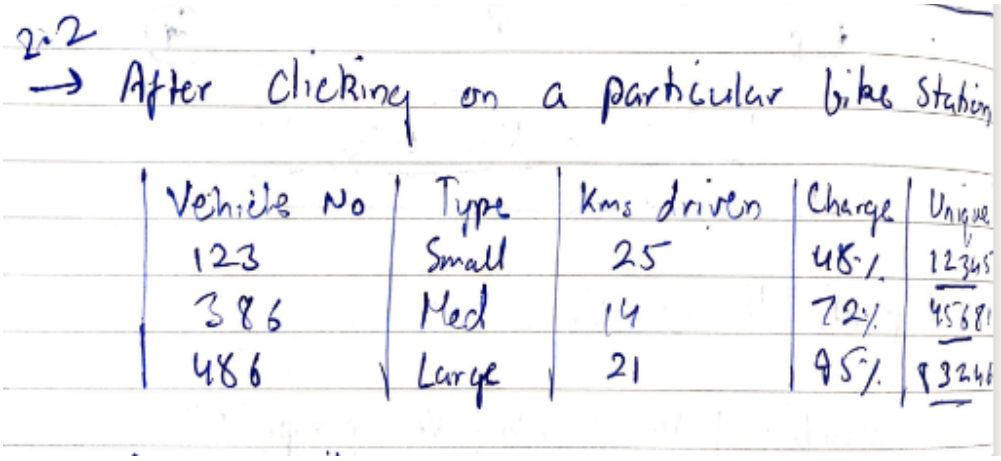


Figure E.5: *Bike options after clicking on individual bike stations*

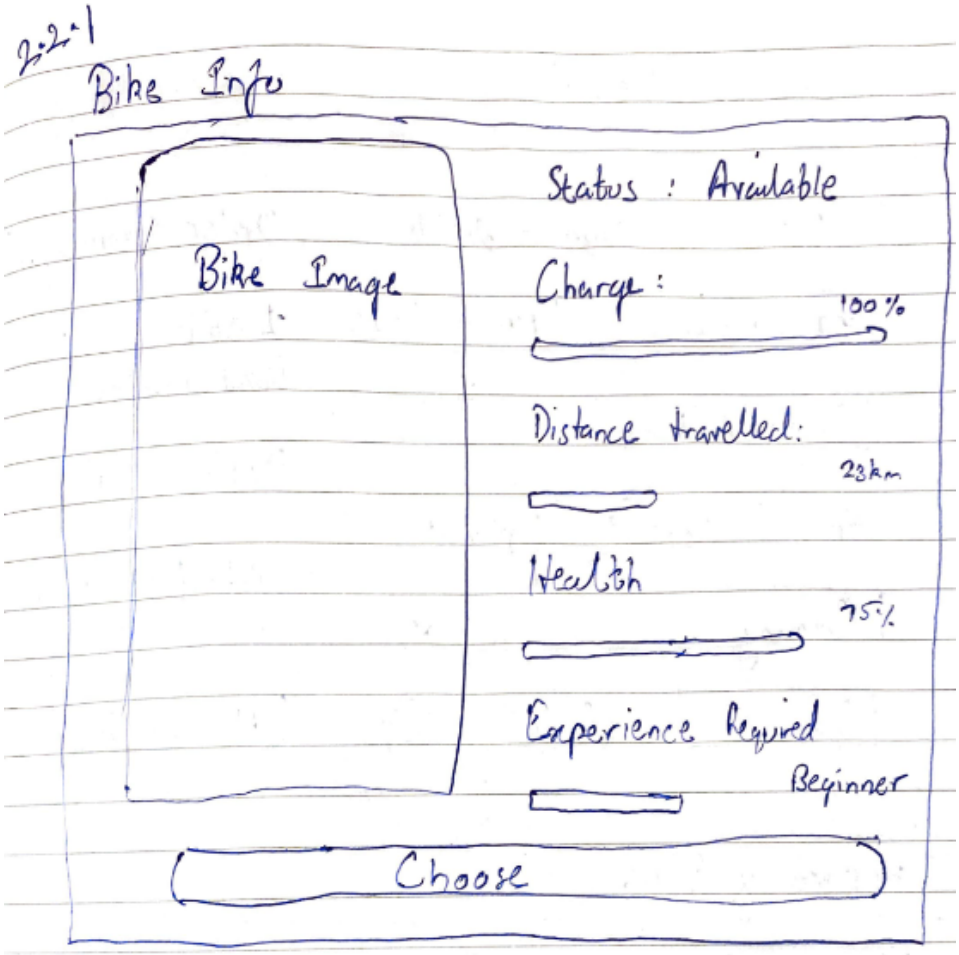


Figure E.6: *Choosing a vehicle to rent*

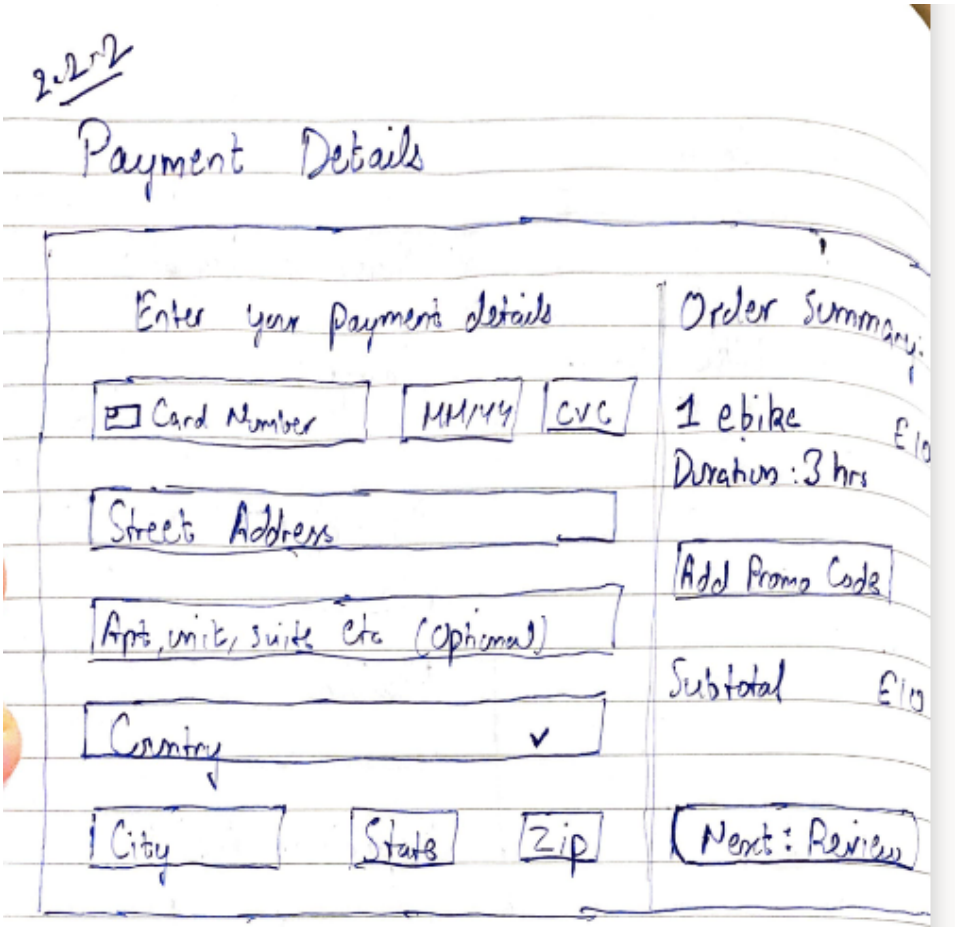


Figure E.7: *Payment screen*

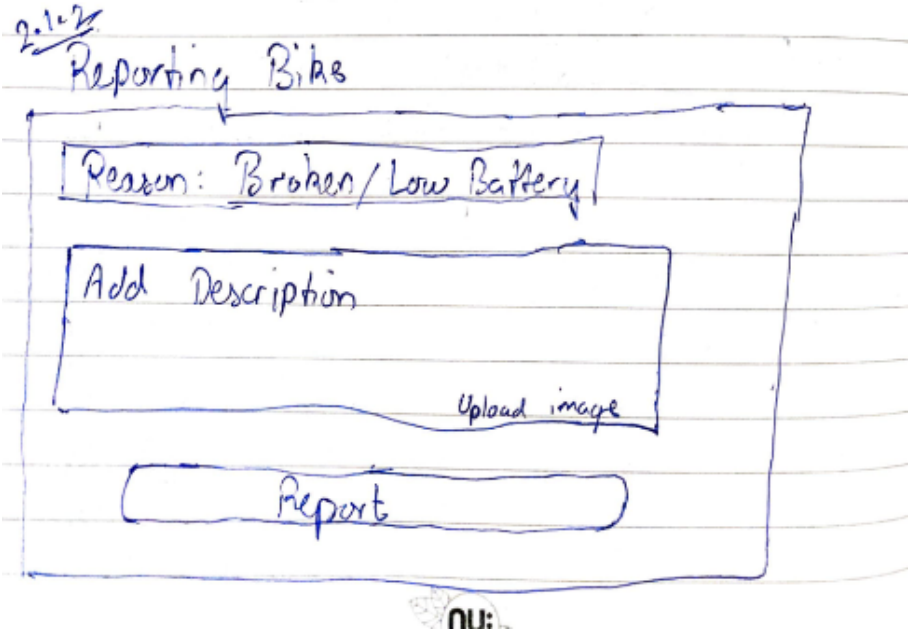


Figure E.8: *Reporting a vehicle defect or low battery state*

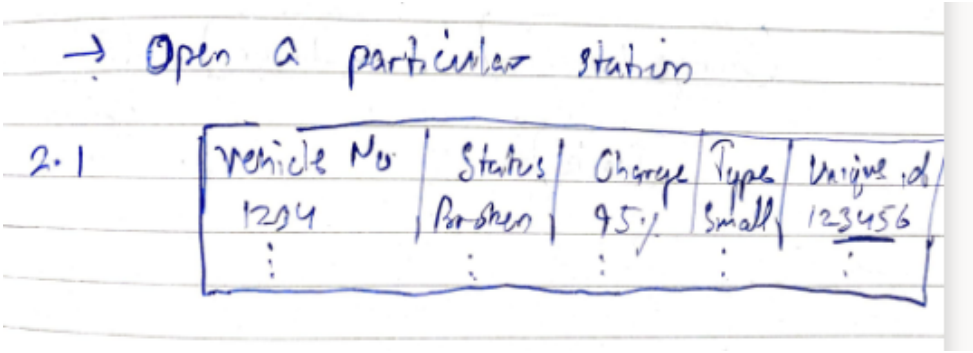


Figure E.9: *Station view for operator*

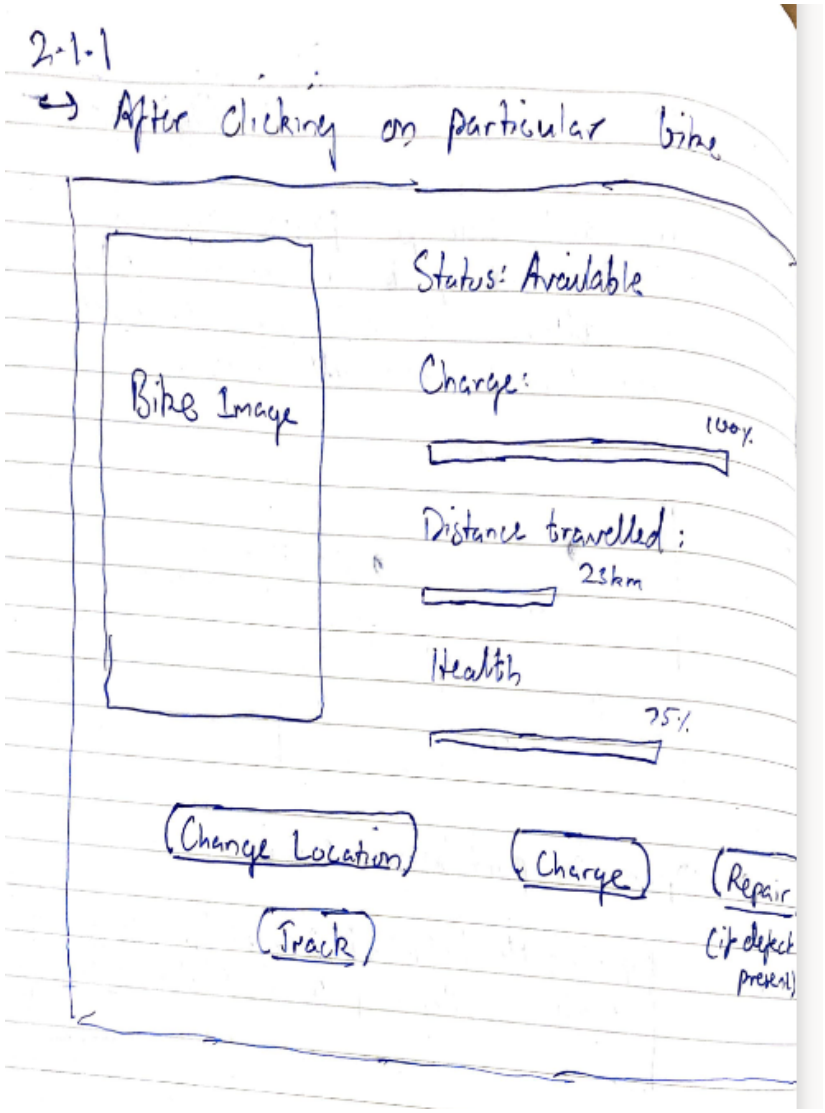


Figure E.10: *Individual vehicle options for operator*

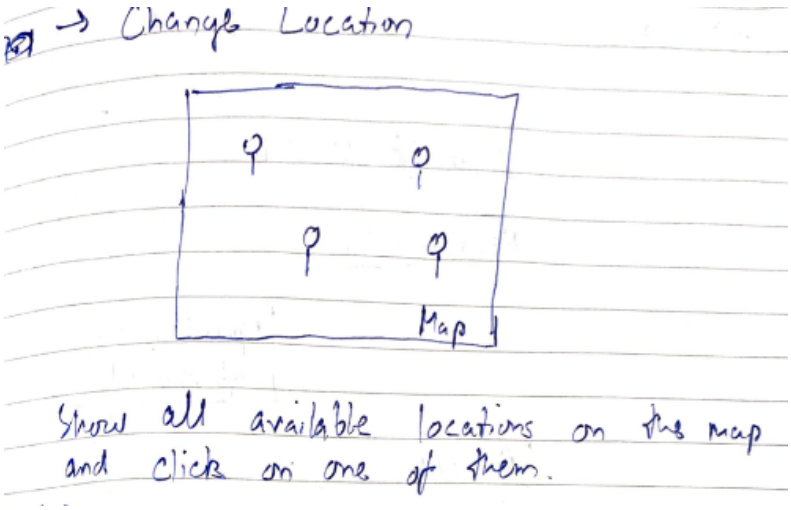


Figure E.11: *Change location view for operator*

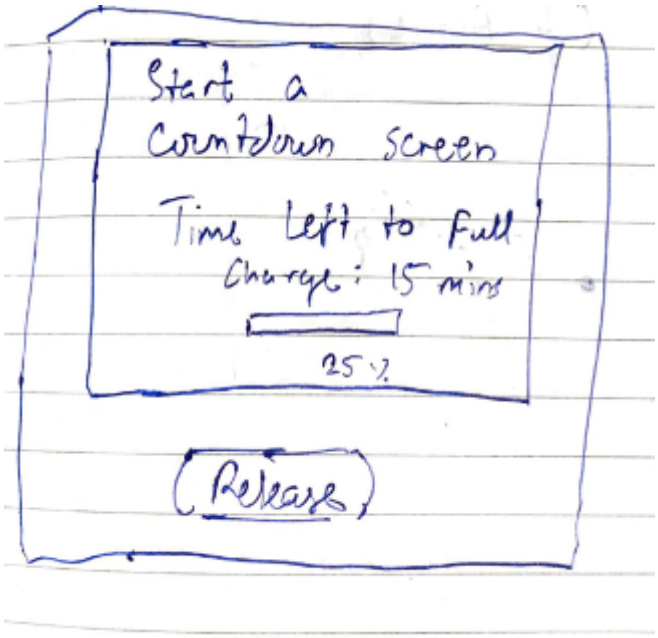


Figure E.12: *Charge vehicle view for operator*

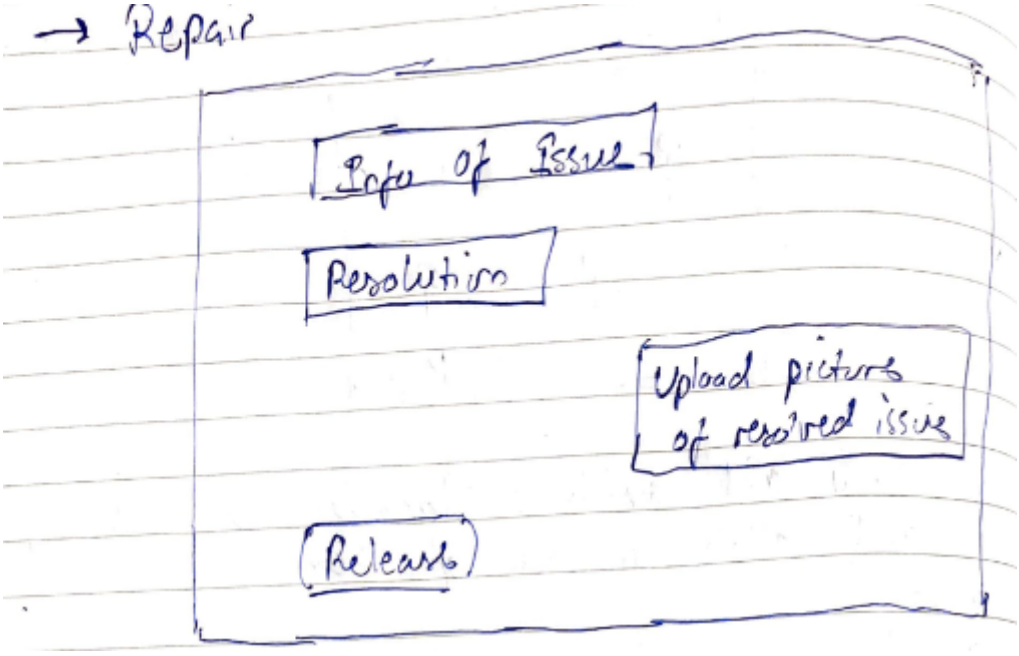


Figure E.13: *Repair vehicle view for operator*

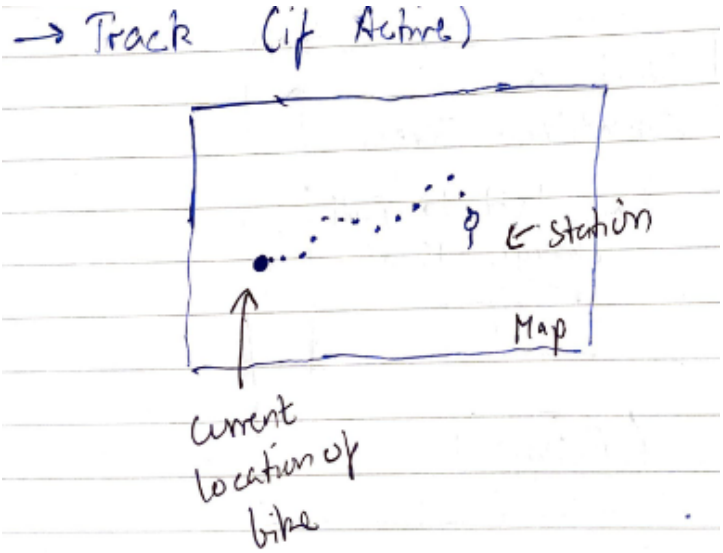


Figure E.13: *Track vehicle view for operator*

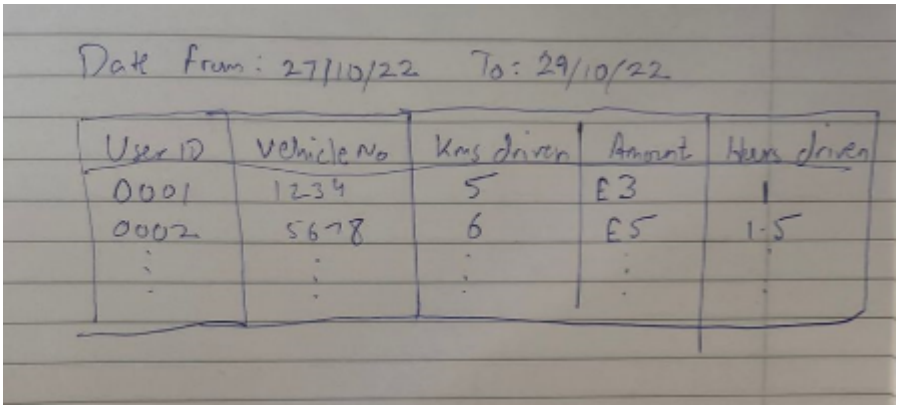


Figure E.14: *Report generation view for Manager*

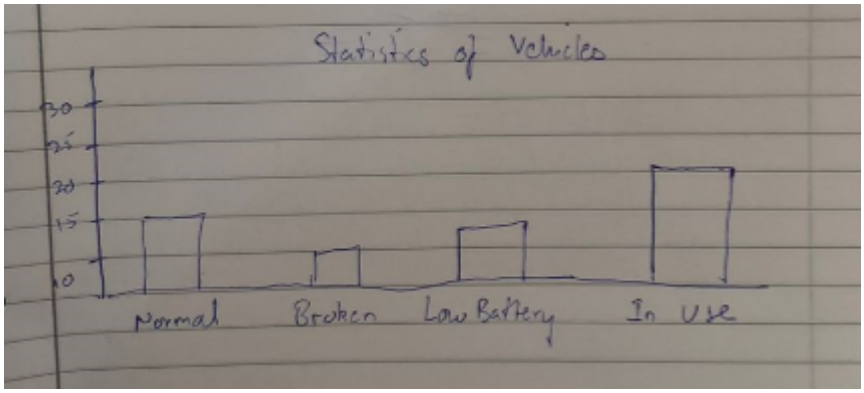


Figure E.15: *Visualizing insights for Manager*

F | Implementation

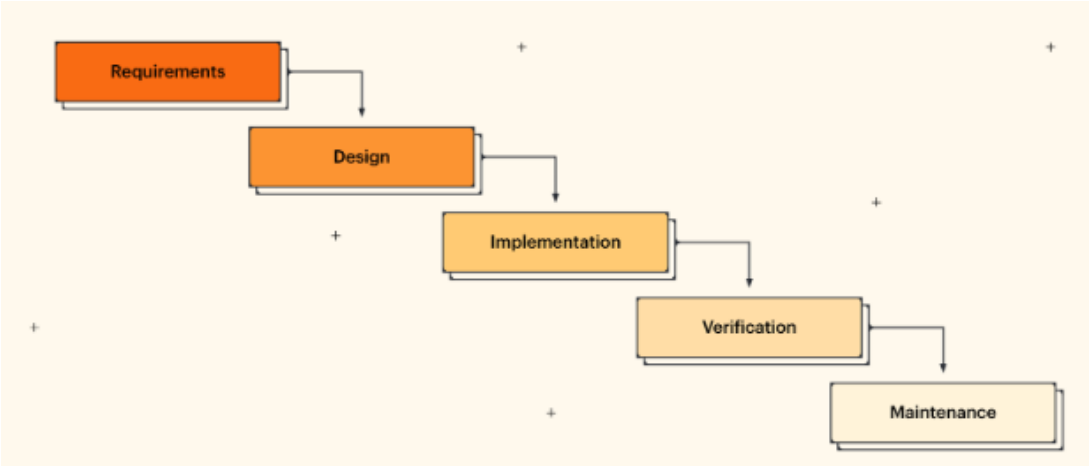
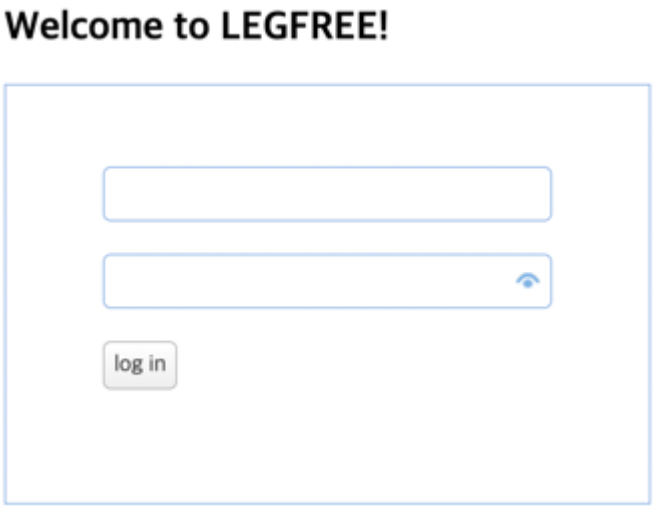
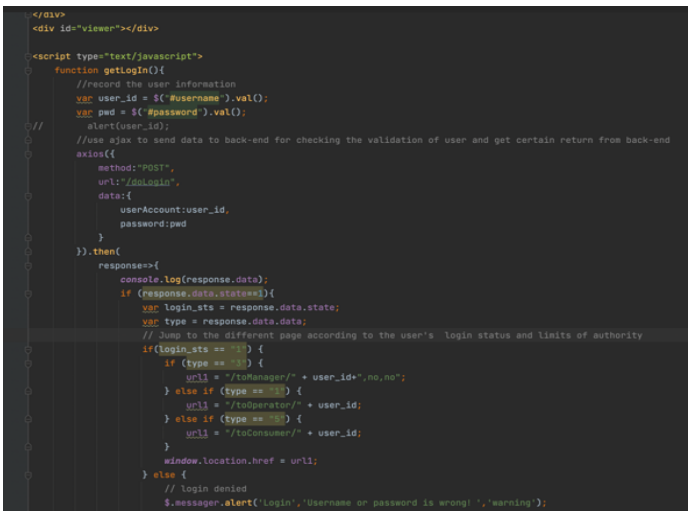


Fig F.1: *Waterfall Model by LucidChart*



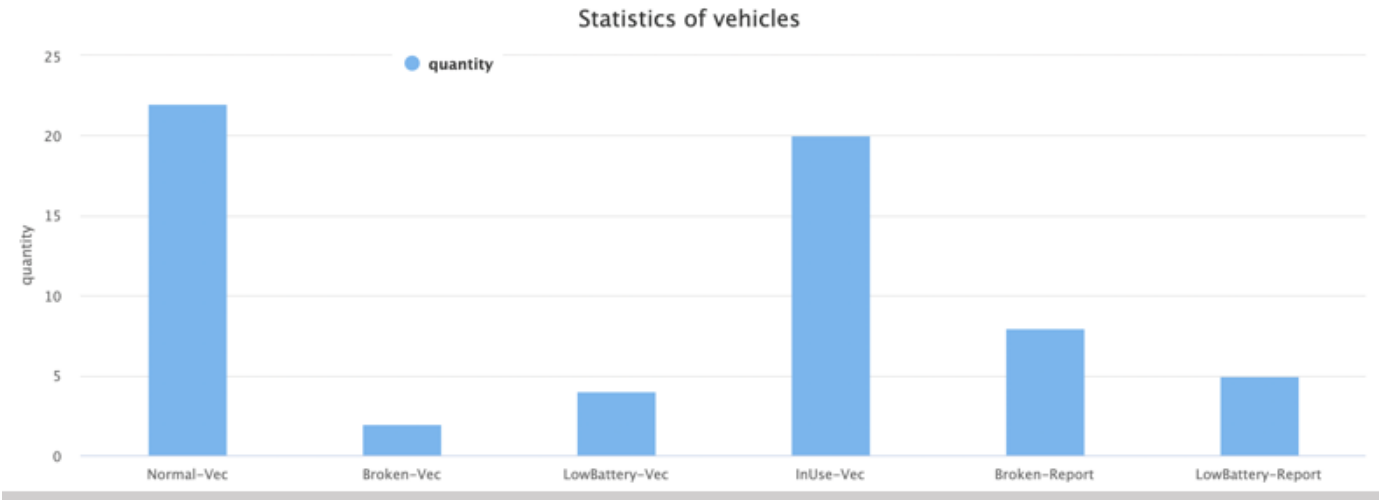


```
</div>
<div id="viewer"></div>

<script type="text/javascript">
    function getLogIn(){
        //record the user information
        var user_id = $("#username").val();
        var pwd = $("#password").val();
//          alert(user_id);
        //use ajax to send data to back-end for checking the validation of user and get certain return from back-end
        axios({
            method:"POST",
            url:"/doLogin",
            data:{
                userAccount:user_id,
                password:pwd
            }
        }).then(
            response=>{
                console.log(response.data);
                if (response.data.state==1){
                    var login_sts = response.data.state;
                    var type = response.data.data;
                    // Jump to the different page according to the user's  login status and limits of authority
                    if(login_sts == "1") {
                        if (type == "1") {
                            url1 = "/toManager/" + user_id+",no,no";
                        } else if (type == "3") {
                            url1 = "/toOperator/" + user_id;
                        } else if (type == "5") {
                            url1 = "/toConsumer/" + user_id;
                        }
                        window.location.href = url1;
                    } else {
                        // login denied
                        $.messager.alert('Login','Username or password is wrong! ','warning');
```

Figure F.2: (a) *Realised login page for users* ***(b)*** *Implementation of login page*

Figure F.2: (a) *Realised renting system for customers* ***(b)*** *Implementation of the renting system*



```
    -->
    <script type="text/javascript" src="/js/jquery.min.js"></script>
    <script type="text/javascript" src="/js/jquery.easyui.min.js"></script>
    <script type="text/javascript" src="/js/axios.js"></script>
    <script src="/js/highcharts.js"></script>
</head>
<body>
    <h2>Manager Page</h2>
    <div style="..."></div>


    <table id="dg1" class="easyui-datagrid" title="Data of Renting" style="..."
           data-options="rownumbers:true,singleSelect:true,url:url,method:'get',toolbar:'#tb',footer:'#ft'">
        <thead>
        <tr>
            <th data-options="field:'orderId',width:120">Order id</th>
            <th data-options="field:'userId',width:80">User Id</th>
            <th data-options="field:'vehicleId',width:80">vehicle Id</th>
            <th data-options="field:'time',width:80">Using time</th>
            <th data-options="field:'orderMoney',width:80">Fees</th>
            <th data-options="field:'state',width:80">Order state</th>
            <th data-options="field:'createTime',width:140">Create Time</th>
            <th data-options="field:'fromLocation',width:100">Departure</th>
            <th data-options="field:'toLocation',width:100">Destination</th>
        </tr>
        </thead>
    </table>
    <div id="tb" style="...">
        Date From: <input class="easyui-datebox" id="date1"  style="...">
        To: <input class="easyui-datebox" id="date2" style="...">
        <a href="#" class="easyui-linkbutton" iconCls="icon-search" onclick="doSearch()">Search</a>
        <a href="#" class="easyui-linkbutton" iconCls="icon-back" onclick="doQuit()">Log out</a>
    </div>
```

Figure F.3: (a) *Visualization for the manager* ***(b)*** *Implementation for manager visualizations*

G | Finished System

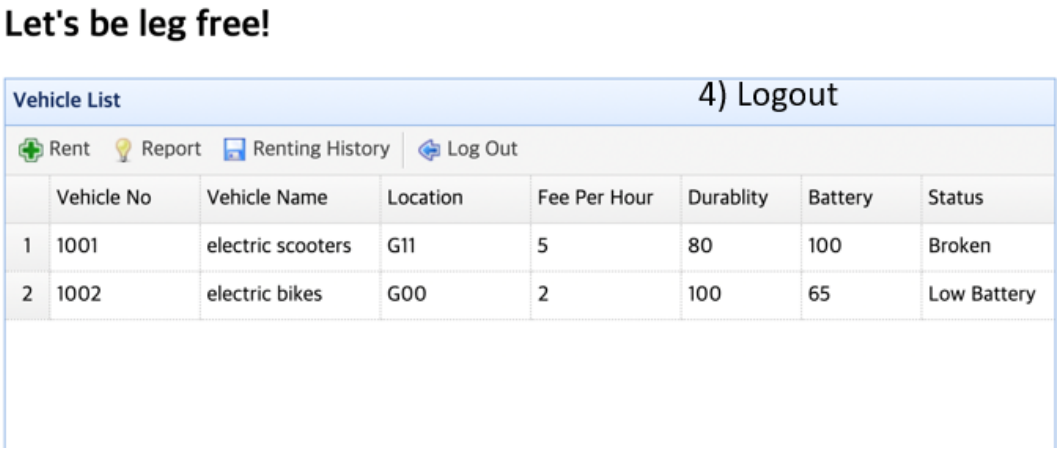


Fig G.1: *Customer view of vehicle list*

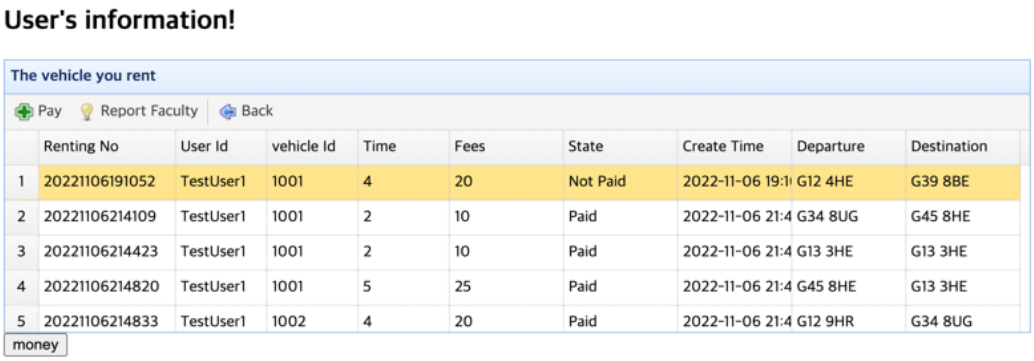


Fig G.2: *Customer rent history page which shows all the vehicles the user has rented, the time (in hours) the vehicle was rented for, the status of payment for a particular ride, fee associated with a ride, creation time, and ride departure and destination locations*

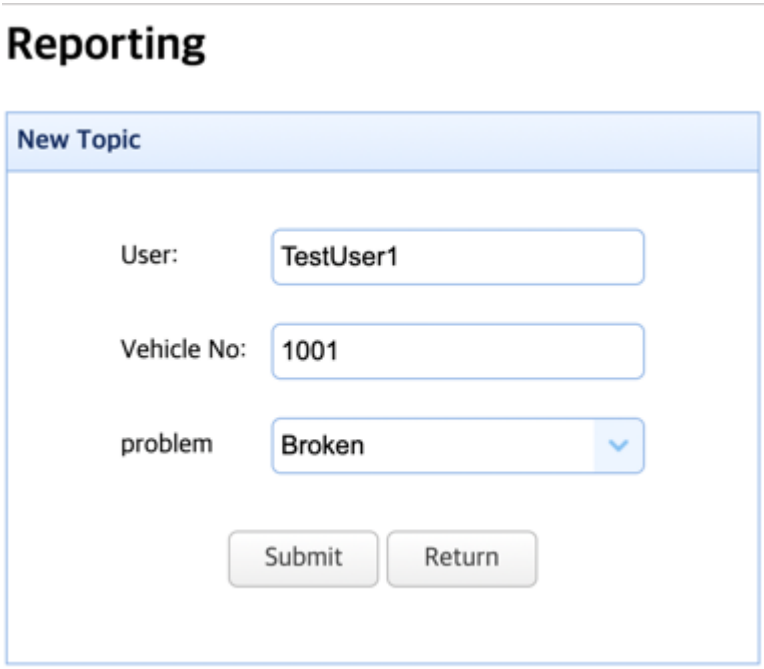


Fig G.3: *Reporting page to submit vehicle defect or low battery state*

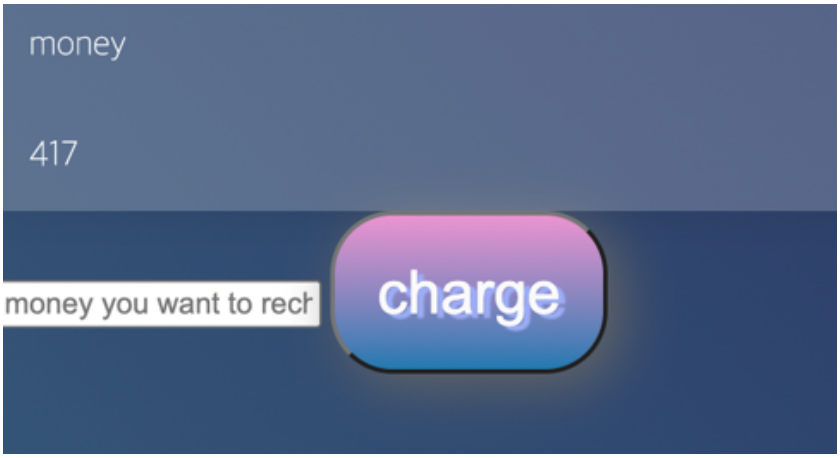


Fig G.4: *The money is the current available balance and the customer can top up by pressing the charge button*

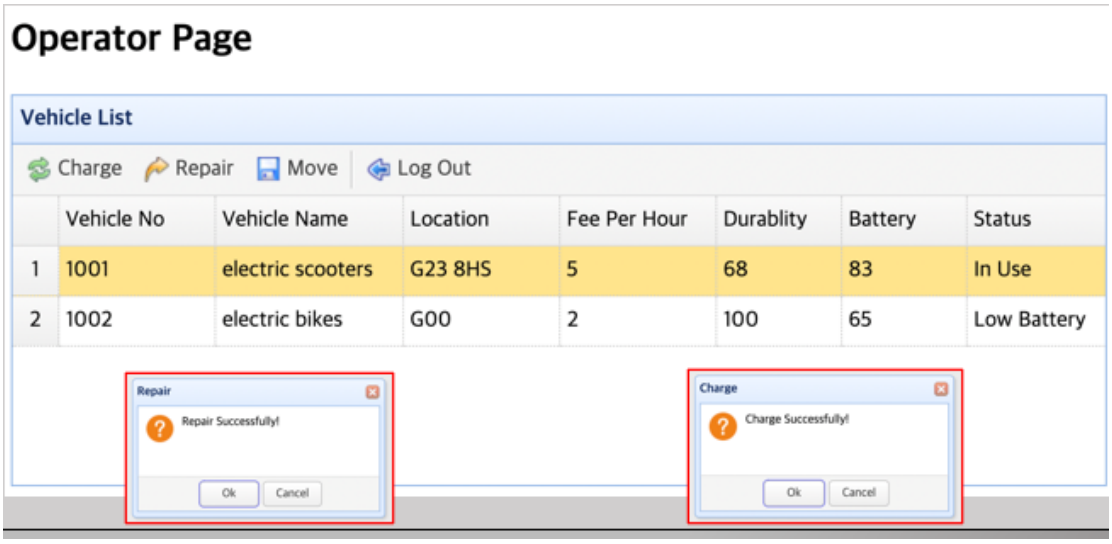


Fig G.5: *Operator dashboard showing list of vehicles along with current battery and durability levels. The operator has an option to repair broken vehicles or charge them which will show respective popups*

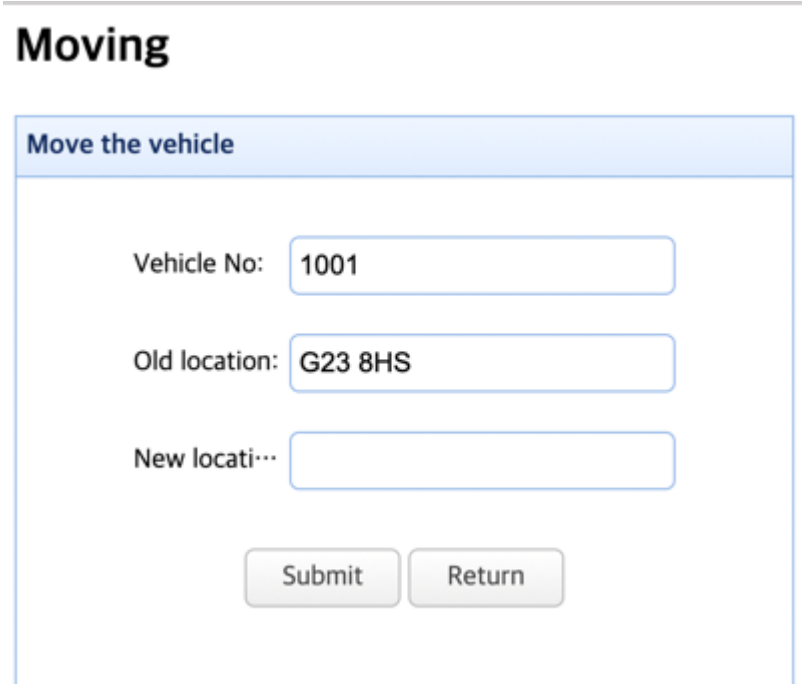
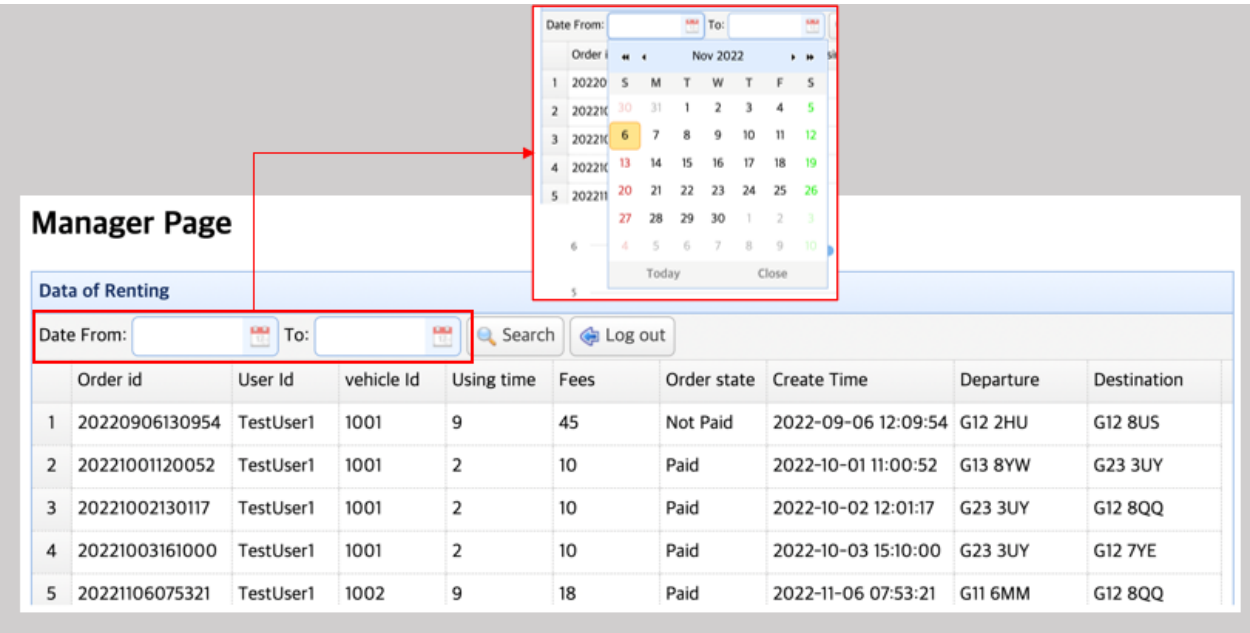


Fig G.6: *Moving the vehicle from one location to another*



	Order id	User Id	vehicle Id	Using time	Fees	Order state	Create Time	Departure	Destination
1	20220906130954	TestUser1	1001	9	45	Not Paid	2022-09-06 12:09:54	G12 2HU	G12 8US
2	20221001120052	TestUser1	1001	2	10	Paid	2022-10-01 11:00:52	G13 8YW	G23 3UY
3	20221002130117	TestUser1	1001	2	10	Paid	2022-10-02 12:01:17	G23 3UY	G12 8QQ
4	20221003161000	TestUser1	1001	2	10	Paid	2022-10-03 15:10:00	G23 3UY	G12 7YE
5	20221106075321	TestUser1	1002	9	18	Paid	2022-11-06 07:53:21	G11 6MM	G12 8QQ

Fig G.7: *Manager dashboard which shows all the information for all the vehicles currently in the fleet. Manager can fetch reports based on date range through this dashboard.*

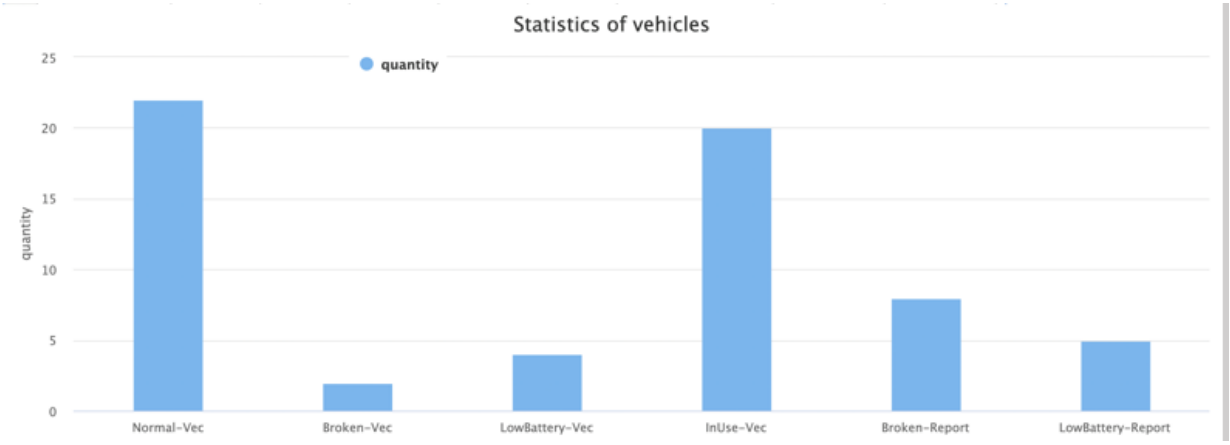


Fig G.8: *Manager bar-chart representing number of normal, broken, low battery, in use and reported vehicles currently in the fleet*