



School of
Computing Science

Electronic Vehicle Sharing System

2742556 Chen, Xinyuan
2773581 Goyal, Ansh Amit
2825912 Gu, Yuxuan
2737549 Liu, Shizhen
2734476 Liu, Zilong
2813743 Omodanisi, Racheal
2739690 Tasin, Ahnaf Ismat

COMPSCI4084: Programming and Systems Development

Dr. Mireilla Bikanga

A dissertation presented in partial fulfilment of the requirements of the
Degree of Master of Science at the University of Glasgow

07/11/2022

Abstract

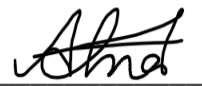
The concept of shared resources has been implemented in numerous industries, including transportation. The notion of sharing electric vehicles provides users with access to enjoy the benefits of electric vehicles of varying ranges without requiring ownership. This project enables users to hire available vehicles through a software application as long as the location is within the service region, rented vehicles may be returned anywhere for a fee. This system monitors these vehicles at all times to determine repairs, the next battery charge, and other forms of maintenance for the efficient and successful continuation of service. This system also allows the generation of reports that enhance management decision-making using appropriate data visualisation approaches.

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.



Liu, Shizhen



Tasin, Ahnaf Ismat



Goyal, Ansh Amit

Yuxuan Gu

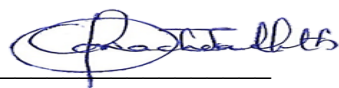
Gu, Yuxuan

Zilong Liu

Liu, Zilong

Chen xinyuan

Chen, Xinyuan



Omodanisi, Racheal

Acknowledgments

This project has increased our proficiency and confidence in system development. We would like to thank Dr. Mireilla Bikanga Ada, Dr. Mary Ellen Foster and Dr. Kevin Bryson for providing us with guidance during the many stages of development.

In addition, we would like to thank our classmates, whose interactions and contributions helped us better envision the functional and non-functional needs of this E-Vehicle Share system.

Table of Contents

Chapter 1	Introduction	6
1.1	What is a ride-sharing service?	6
1.2	Electric motors vs combustion engines	6
1.3	Motivation	6
1.4	Aim and Objectives	7
Chapter 2	Background Survey	8
2.1	Rise of Electric Scooters	8
2.2	Notable Services	8
Chapter 3	Requirements	10
Chapter 4	Design	11
4.1	The System Architecture	11
4.2	Software Development Methodology	12
4.2.1	Why do we need one?	12
4.2.2	Different Types of Methodologies	12
4.2.3	Our Adopted Methodology	13
4.3	ER Diagram	13
Chapter 5	Implementation	14
5.1	List of Technologies Used	14
5.1.1	Application Type	14
5.1.2	Programming Language	14
5.1.3	Database	15
5.1.4	Visualisation	15
5.1.5	Graphical User Interface (GUI)	16
5.2	Complexity of Implementation	16
5.3	Front End	16
Chapter 6	Testing	18
6.1	Strategy	18
6.2	Framework	19
6.3	Test table	19
Chapter 7	Conclusion & Future Work	20
References		22

Chapter 1 Introduction

1.1 What is a ride-sharing service?

A ride-hailing company (also known as a transportation network company, ride-sharing service; the vehicles are termed app-taxis or e-taxis) links customers with drivers of for-hire automobiles via mobile apps and websites. These vehicles may or may not have drivers driving them, or just be rental vehicles that are available to users. Such companies are now very common especially with the rise of inexpensive edge-devices and accessible internet, which minimises human interaction and takes away the bottleneck while hiring any kind of vehicle. These vehicles can have classic internal combustion engines or modern battery-operated electric vehicles (EVs).

1.2 Electric motors vs combustion engines

Vehicles that emit little to no carbon dioxide have seen their demand skyrocket over the past several years as contemporary businesses have made a concerted effort to shift to more environmentally friendly forms of energy. Firms that manufacture and supply electric vehicles as a service with a significant emphasis on software solutions have emerged as a result of this phenomena, which, when paired with the progressive character of technology companies, has led to their proliferation.

Driving an electric vehicle in the United Kingdom produces carbon emissions that are on par with those of the most fuel-efficient diesel vehicles and approximately 30 percent lower than the average for new vehicles powered by fossil fuels. This is because the energy mix in the United Kingdom currently favours the use of fossil fuels. The current obstacles to adoption include high costs (especially the cost of batteries), a restricted range between recharges, and lengthy recharge times. This is slowly improving due to more companies adopting EV's to provide a service for end-users. One such type of company is an electric EV ride-hailing service where users can use a software application to hire an EV for a certain time and distance for a fee. ^[1]

1.3 Motivation

The motivation behind this project is to present users with a dockless ride-hailing software system through which they will be able to see the availability of electric vehicles in their city and rent them. This system keeps constant watch on these vehicles in order to identify when repairs are needed, when the next battery charge is due, and any other types of

maintenance that may be required to ensure the continuation of service in an effective and satisfactory manner. Additionally, this system enables the development of reports that, via the use of relevant data visualisation strategies, improve managerial decision-making. The concept of sharing electric vehicles will give consumers access to experience the benefits of electric vehicles of varied ranges without the need for ownership of the vehicles themselves.

1.4 Aim and Objectives

The Aim of this project is to deploy a software system that facilitates or supports Electric Vehicle share amongst users in location of available services. The objectives are to:

- I. Use Map services to determine user location and grant access to the E-vehicle share services.
- II. Use tracking technologies to identify defects in vehicle parts for maintenance and other needs such as electric charging etc.
- III. Use a secured payment platform for charges and collections.
- IV. Present reports on key performance to management using analytic techniques.

The rest of the report is structured as follows: Chapter 2 *Background Survey*, Chapter 3 *Requirements*, Chapter 4 *Design*, Chapter 5 *Implementation*, Chapter 6 *Evaluation* and Chapter 7 *Conclusion and Future Work*.

Chapter 2 Background Survey

2.1 Rise of Electric Scooters

Electric scooters have grown in popularity as a micro mobility transportation service across the world. E-scooters are often "dockless," which means they do not have a set home location and are dropped off and picked up at specific service locations. The surprising surge of shared electric scooters, particularly in the United States, has piqued the curiosity of business companies and investors alike. A number of venture investors have already made investments in electric scooter rental companies. Customer desire for independent mobility is driving this new method of transportation. The growing demand among commuters for lightweight vehicles capable of covering short distances is a major factor driving the widespread adoption of electric scooters that may be shared. This mode of transportation not only reduces commuters' carbon footprint, but it is also environmentally friendly. The accessibility of rented electric scooters is an extra bonus. ^[2]

2.2 Notable Services

Lime was founded in San Francisco, California. Users may log onto the app, and it will instantly monitor their location and put them in the direction of the nearest docking station. Users may search for and unlock dockless automobiles using an app that uses GPS to locate accessible places. Each Lime scooter costs £1 to unlock and 15p per minute to use. When a ride is finished, users must park the scooter at approved parking areas and take a photo to certify the ride's completion. Lime has a collaboration with Uber that allows consumers to discover scooters in their region via the Uber app. Lime's success has prepared the path for its expansion into Europe, specifically Madrid. ^[3]

Bolt is another company that offers an electric scooter ride-sharing platform. This multinational brand of e-scooters is available in most major cities around the United Kingdom. Bolt scooters are among the lightest on the market, weighing only 17 kg and capable of travelling 25 kilometres on a single charge. Bolt's app is now Europe's only platform that offers ride-hailing, e-scooters, and e-bikes under one single umbrella brand. ^[4]

Bird is now the most popular e-scooter firm in the United States and the United Kingdom. Their scooters are only permitted to be ridden on roads with speed limits of 45 miles per hour or fewer, as well as within bicycle lanes. Birds, unlike others, do not have specific standing places. The firm recommends keeping a scooter away from public walkways and parking it

using the kickstand. A Bird scooter features Bluetooth, which is a unique feature. This lets you connect your gadgets to the scooter and listen to your phone's playlists or favourite podcasts, exactly as you would in a vehicle. In London, a Bird scooter costs 25p per minute plus a £1 unlocking charge.^[5]

Ginger is a UK-based firm that says riders may ride 24 hours a day, seven days a week. However, the company's website states: 'In a few cities, new journeys cannot begin after 10 p.m. or 11 p.m.' The next morning, these journeys become available again. Ginger, like Lime, employs approved parking locations that can be located on a map in its app. The Ginger app also has an area for special deals and promotions. This area is ideal for individuals who want to save money while still experiencing the complete e-scooter experience. People over the age of 18 can rent an electric scooter for £2 per 20 minutes, plus 50p each 10 minutes of pause time.^[5]

Wind is another UK based company that works in Nottingham and Derby currently. To lessen the danger of theft or vandalism, Wind is taking a more focused approach. The e-scooters are outfitted with circuitry that prevents them from operating outside of designated locations. To guarantee correct use, the operator may track the scooters. One of Wind's most appealing features is that some of its e-scooters have a removable helmet. Key employees can sign up for long-term rentals at a reduced rate. In addition, instead of leaving the motor in a specified place, people can take an e-scooter home for exclusive usage. Wind's e-scooters cost £1 to unlock and then 12p each minute after that. The ultimate charge is determined by the number of times the scooter has been unlocked and locked using the app.^[5]

Voi is a Swedish startup that provides a dockless platform, which allows customers to leave their scooters anywhere they like. Users may take a 'helmet selfie' to confirm they're wearing a helmet and earn loyalty points that can later be converted into reduced trips. The e-scooters from Voi cost £1 to unlock, plus 20p per minute. Users can also purchase daily and monthly passes that are limitless. A day pass costs £9.99, while a monthly pass costs £39.00.^[5]

In the United Kingdom, the government is currently working on ideas to legalise the use of e-scooters on public roads by introducing a brand new vehicle class. This measure is expected to be included in a Transportation Bill that is now being developed and might be approved soon.^[6]

Chapter 3 Requirements

Through the use of our software system,

Customers should be able to:

- Rent a vehicle at any location in the city, as long as there is a working vehicle available at that location.
- Return a vehicle to any location. When a customer returns a vehicle, their account is charged an amount depending on how long the vehicle rental was and what type of vehicle was used.
- Report a vehicle as defective.
- Pay any charges on their account.

Operators should be able to:

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

Managers should be able to:

- Generate reports showing all vehicle activities over a defined time period, using appropriate data visualisation techniques.

Chapter 4 Design

4.1 The System Architecture

The goal of this project is to provide a software solution for an electric vehicle sharing service delivered via a desktop application. We built a fully working end-to-end prototype and fed it relevant data. Our software solution is intended to provide an interface for customers to rent cars for a price, for operators to monitor the status of vehicles in the system and make required modifications, and for managers to observe and generate usage statistics.

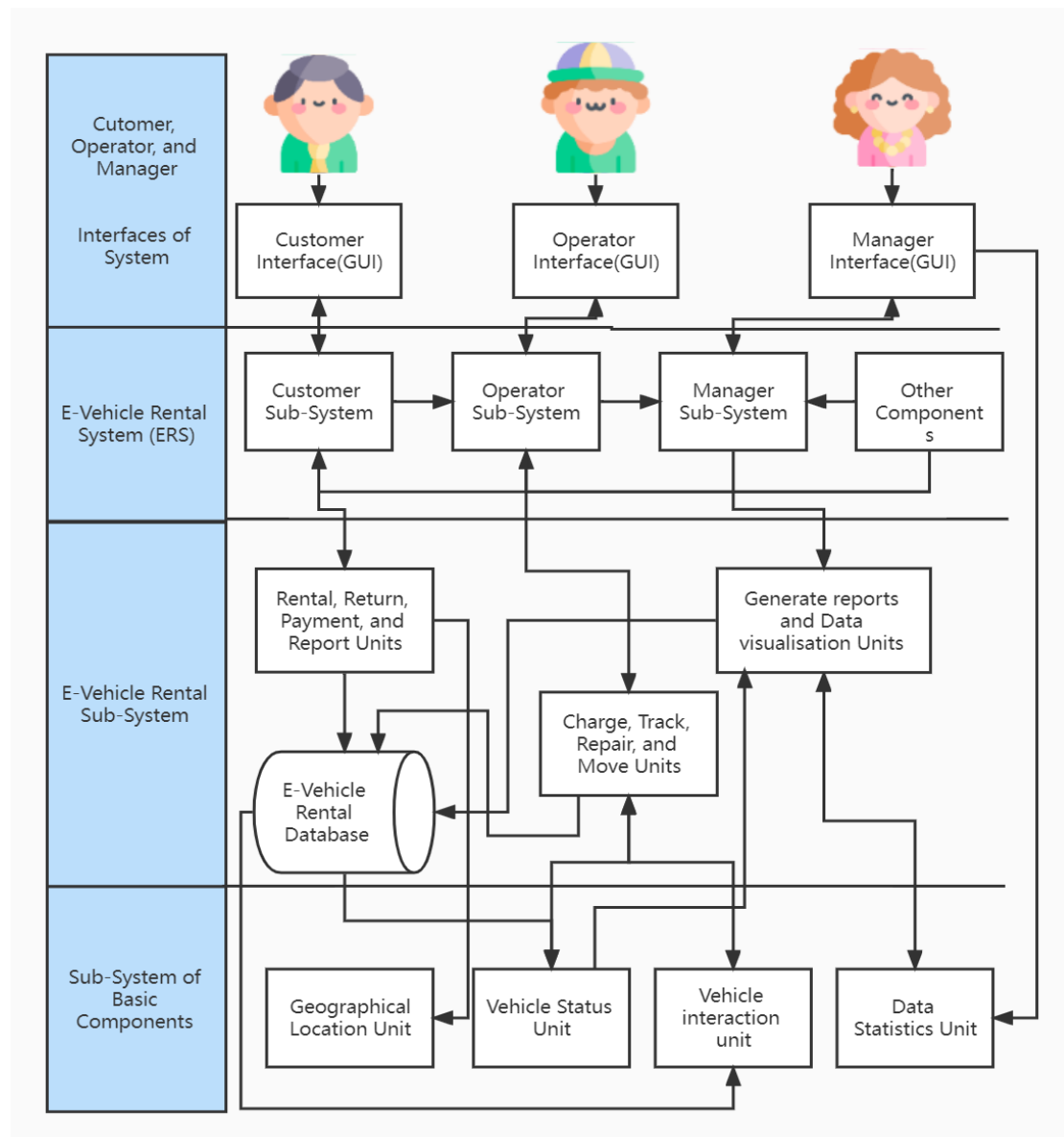


Figure 4.1: System Architecture Design

4.2 Software Development Methodology

4.2.1 Why do we need one?

It is important to decide on a particular software development approach in order to assess customer needs and construct solid solutions on schedule in a way that all developers can work in harmony. A dependable methodology is a vital component of any software project since it helps the team to meet all of the clients' business needs and guarantees a feature-rich product in the most efficient way.

Every day, several components of software are planned, designed, produced, and deployed, and each of these activities needs collaboration between developers. Collecting needs from stakeholders, creating a strategy, deciding on the possible use cases, designing the product, inventing and testing it, and eventually delivering it to the client: these are some of the primary components of any software development strategy.^[7]

4.2.2 Different Types of Methodologies

The most commonly used phrase to describe development methodologies is agile. It's frequently used as an umbrella phrase to describe any agile technique, which is an iterative process that minimises waste and maximises efficiency. The agile approach may be compared to jazz, which is created via band members' collaboration, experimentation, and iteration. It is adaptable and changes in response to new ideas, events, and directions.^[8] In contrast to traditional project management, most software development approaches are agile, with a heavy emphasis on iteration, cooperation, and efficiency.

Kanban approach is an agile method that aims for continuous improvement, task management flexibility, and improved workflow. The development of the entire project may be clearly comprehended using this illustrated technique.^[9] Kanban may be utilised in any knowledge work scenario, but it is especially useful when work arrives in an unpredictable manner and/or when we want to distribute work as soon as it is available, rather than waiting for other work items to finish. The Kanban approach is centred on the Kanban board. It is a tool that visualises the complete project in order to follow its progress. A new member or an external entity may grasp what is occurring right now, completed tasks, and upcoming tasks by using the graphical style of Kanban boards.^[10]

Scrum is a popular agile approach for product development, particularly software development. Scrum is a methodology that may be used for any project that has tight deadlines, complicated needs, and a degree of originality. Scrum projects progress through a series of revisions known as sprints. Each sprint normally lasts two to four weeks.^[11] A normal scrum

team consists of five to nine individuals, however Scrum projects may quickly expand to hundreds of workers. Scrum, on the other hand, is readily and frequently utilised by one-person teams. There are no standard software engineering jobs on this team, such as developer, designer, tester, or administrator. Everyone on the project collaborates to finish the set of tasks that they have jointly agreed to do within a sprint. Scrum teams foster a strong feeling of community and a sense that "we're all in this together."^[12]

4.2.3 Our Adopted Methodology

We adopted the Waterfall model which is a traditional, linear management system. If the agile methodology is like jazz music, the waterfall methodology is classical music with a predetermined plan for how the song should be performed. It's the polar opposite of agile, prioritising keeping to the plan above adjusting to changing situations, but that's okay since at the start of the project, we were given clear requirements and constraints.^[13] This strategy is a straight, step-by-step procedure with one activity feeding into the next, hence the term "waterfall." The approach is plan-driven and rigorous, with little space for flexibility. The requirements list allowed us to gather all relevant information at the start of a project and utilise it to develop an educated plan of action. We determined the possible Use Cases ([Appendix Diagram 1](#)) of the project, determined the UML diagram ([Appendix Diagram 2](#)) and flow-diagrams for the Customer ([Appendix Diagram 3](#)) and Operator ([Appendix Diagram 4](#)) part and finally, we divided the tasks into 4 sub-teams: the customer team, the operator team and the manager team. Each sub-team was responsible for the back-end and front-end functionality of their respective parts as well as testing their code. The other sub-team was responsible for planning the timeline and writing the report. This sub-divisions allowed each team member to become acquainted with the chosen tech stack as well as with one another.

4.3 ER Diagram

An Entity Relationship (ER) Diagram is a sort of flowchart that shows how "entities" in a system, such as people, things, or concepts, interact with one another. ER Diagrams are most commonly used in the disciplines of software engineering, and employ a predetermined collection of symbols such as rectangles, diamonds, ovals, and connecting lines to illustrate the interconnectivity of entities, relationships, and their qualities. They are also known as ERDs or ER Models.^[14]

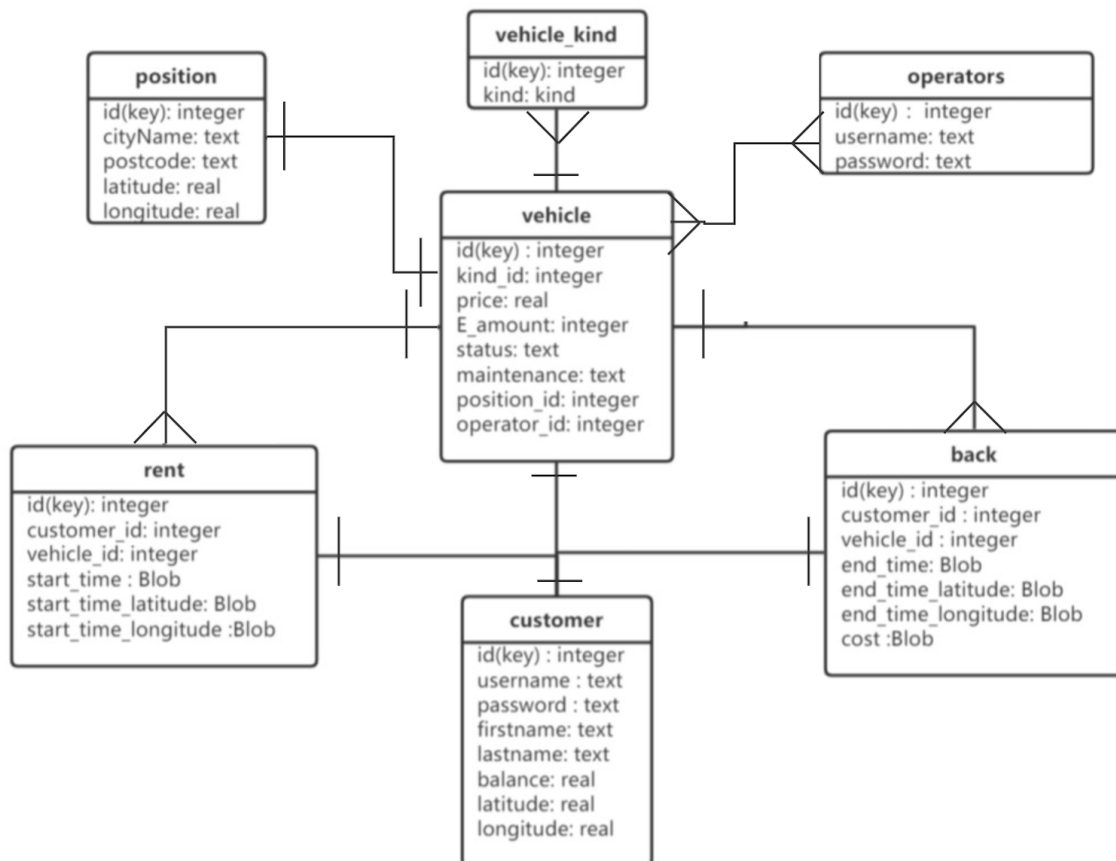


Figure 4.3: ER Diagram

Chapter 5 Implementation

5.1 List of Technologies Used:

5.1.1 Application Type

Our project uses a desktop application which are computer software programmes that are executed directly on the hardware of the computer. They must be installed on a desktop or laptop computer in order to be used since, unlike web-based applications, they cannot be accessed using a web browser.^[15]

5.1.2 Programming Language

We used Python which is a general-purpose, high-level programming language. Its design philosophy emphasises code readability, and it is frequently used for building websites and applications, automating operations, and doing data analysis. Python is a general-purpose programming language, which means it may be used to develop a range of different applications and is not specialised for solving any particular problem. Python supports dynamic typing and garbage collection. It supports several programming paradigms, including structured programming (especially procedural programming), object-oriented programming, and functional programming. It is sometimes referred to as a "batteries included" language due to its extensive standard library^[16], some of which we used are described in this chapter below.

5.1.3 Database

We used SQLite which is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world. It is designed to allow the program to be operated without installing a database management system or requiring a database administrator. Unlike client-server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, a linker integrates the SQLite library.^[17]

Our system uses SQLite3 to store the details of the vehicles, charging points, city locations, customers, and any other data as needed by our implementation. We implemented four electric vehicle types: *scooter*, *bi-cycle*, *skateboard* and *motorbike*.

5.1.4 Visualisation

The visualisation functionalities were implemented using the Pandas DataFrame and the Matplotlib library.

Pandas was used to configure the data and modify it into a representative format. Pandas DataFrame is a two-dimensional, size-mutable tabular data format with defined axes (rows and columns). A data frame is a two-dimensional data structure in which data is organised in rows and columns in a tabular form. Pandas DataFrame is made up of three major components: data, rows, and columns.^[18]

Matplotlib is used to plot the graph, modify axes, change legends, and make graphs more user friendly. Matplotlib is a cross-platform Python library that allows the creation of static, animated, and interactive visualisations^[19]. It is based on NumPy arrays and is intended to operate with the larger SciPy stack. It includes numerous plots such as line, bar, scatter, histogram etc.^[20] It provides an open source alternative to MATLAB. Developers may also incorporate plots in GUI programmes by using matplotlib's APIs (Application Programming Interfaces). In most cases, a Python matplotlib script is constructed so that only a few lines of code are necessary to produce a visual data plot.^[21]

5.1.5 Graphical User Interface (GUI)

For implementing the GUI, we used PyQt which is a Python binding of the cross-platform GUI toolkit Qt, used as a Python module. It is used for developing desktop applications with Python. PyQt provides many classes, methods and widgets. Unlike Tkinter which is rather limited in the number of widgets, PyQt comes with a large set of widgets included. The official Qt for Python library is named Qt for Python. PySide is the name of its Python package.^[22]

PyQt and PySide are both based on Qt. Their APIs are quite similar since they are based on the Qt API. As a result, converting PyQt code to PySide can be as straightforward as changing a few imports.^[23]

5.2 Complexity of Implementation

The following piece of code shows a crucial *Operator* function *get_need_charge_vehicle()* for determining which vehicles' batteries need to be charged.

```
#a crucial Operator function
def get_need_charge_vehicle():
```



```

"""
:return:list of vehicle
"""

E_amount = 20
sql = "SELECT * FROM vehicle where E_amount < ?"
a = executeQuerySql(sql, (E_amount,))
return a

```

Figure 5.2: Function for finding all vehicles which need to be charged.

5.3 Front End

This section will present a few screenshots of the actual application demonstrating how the front end was developed.

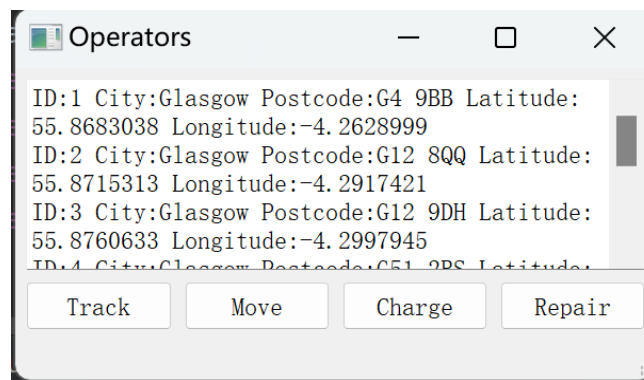


Figure 5.3a: Operator UI

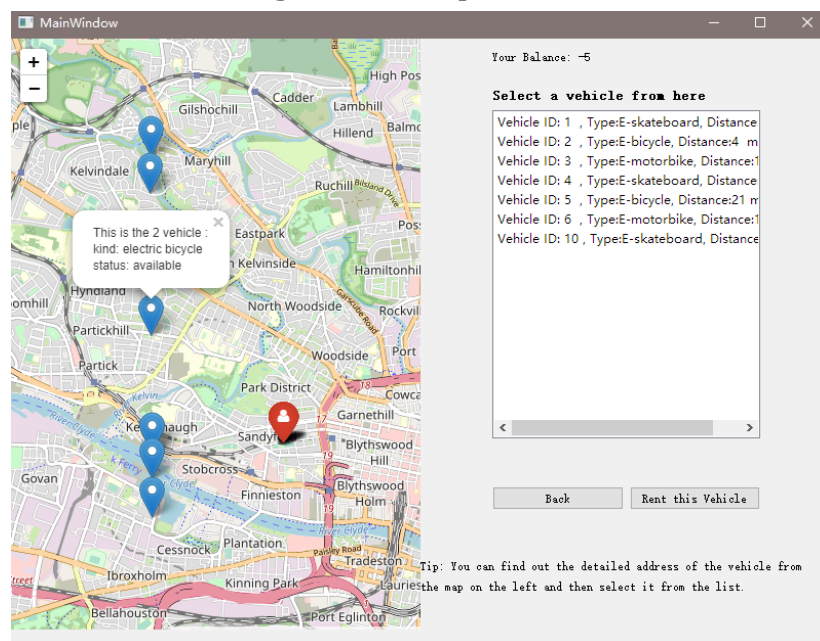


Figure 5.3b: Customer UI

The Manager Dashboard UI features a window titled "Manager" with a close button. It includes two date input fields: "Start Date" (10/4/2022) and "End Date" (11/4/2022), each with up/down arrows. To the right are three buttons: "Last 3 Months" (blue), "Last Month" (orange), and "Last Week" (grey). Below these is a dropdown menu currently showing "Post-Code Activities", with a list of options: "Gender Comparison", "Post-Code Activities" (highlighted), "Number of Vehicles Available", "Defect-Rate", and "Average Charge". At the bottom center is a green "SUBMIT" button.

Figure 5.3c: Manager Dashboard UI

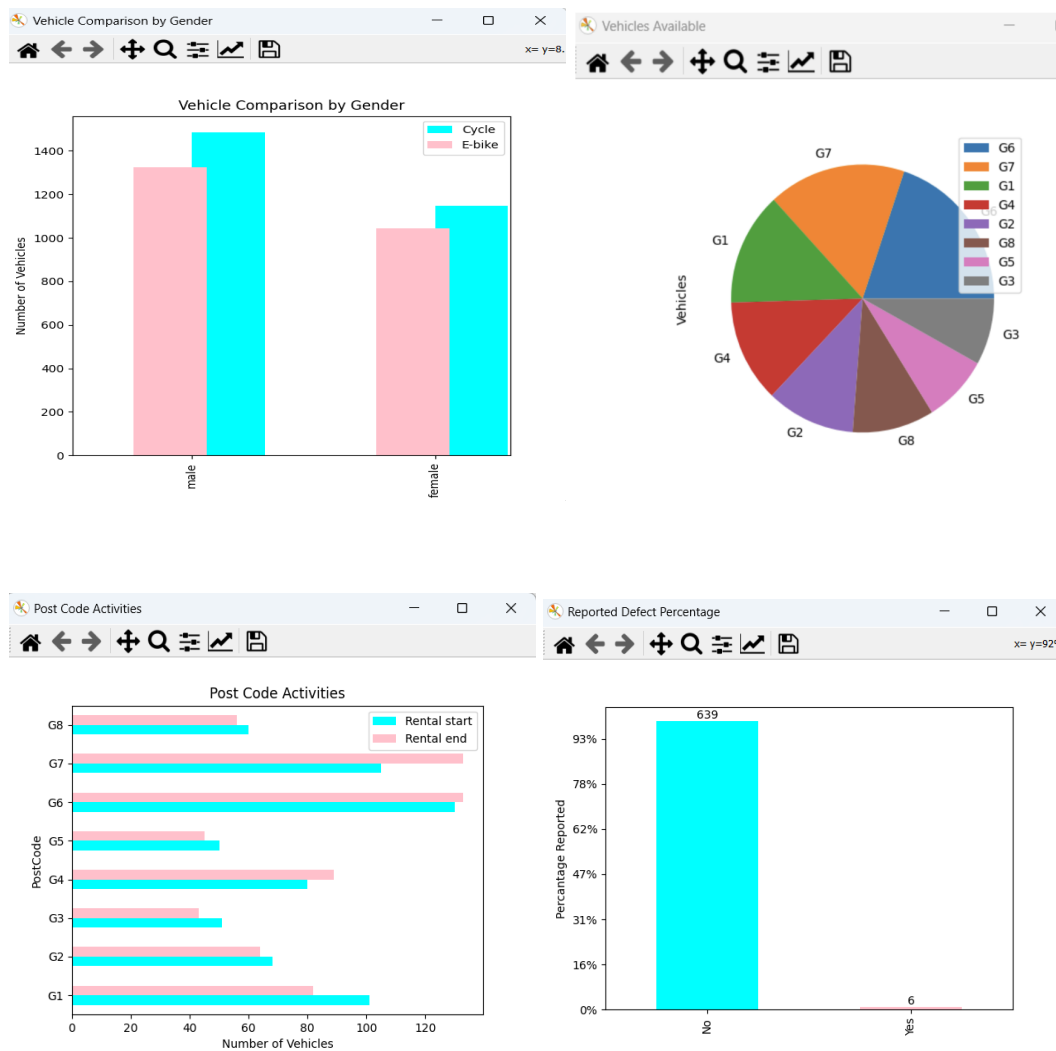


Figure 5.3d: Manager Usage Statistics Reports

[*This*](#) *Is The Link To The Video Demonstration Of Our Application*

Chapter 6 Testing

A test is a code that checks the validity of the other code. Tests are designed to help in gaining confidence that what you wrote is working. It proves that the code is working as we want and gives a safety net for future changes.

6.1 Strategy

Manual test cases were developed to validate app processes and ensure that users, operators and managers could properly use the system's different functions. The type of test we carried out is a black box test, which implies that just the inputs and outputs are considered. This is a functional test, as particular components of the system have been examined. As for the testing technique, we employ Equivalence Class Testing, which allows us to maintain a high level of test coverage while keeping the number of test cases to a minimum.

6.2 Framework

For our framework, we used PyTest to write, test, and scale to support testing for our applications and libraries. PyTest provides valuable failure data without the usage of debuggers. It includes handy plugins and extensions and may be written as a function or method. It is the most popular Python package for testing.^[24]

```

Tests passed: 3 of 3 tests - 22 ms
c:\Users\lsz\AppData\Local\Programs\Python\Python37-32\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2.3/p
Testing started at 23:52 ...
Launching pytest with arguments CustomerTesting.py::TestClass --no-header --no-summary -q in D:\Tapp_remeber_delete_it_after_submm

===== test session starts =====
collecting ... collected 3 items

CustomerTesting.py::TestClass::test_rent PASSED [ 33%]
CustomerTesting.py::TestClass::test_return PASSED [ 66%]The number of rows of data successfully executed 1
0
5
55.8600791 -4.2998451

CustomerTesting.py::TestClass::test_pay PASSED [100%]

===== 3 passed, 1 warning in 1.27s =====

Process finished with exit code 0

```

Figure 6.2: Our Implementation of PyTest

6.3 Test Table

Test Case	Description	Result
New Customer Registration	The user enters the username and password of the account to register by clicking the register button. Their information will be stored in the database and after that, the user can login.	Pass
Existing Customer Login	The user enters a username and password to get access by clicking the login button. The user will be directed to the vehicle renting menu, which will display the renting menu.	Pass
Rent a Vehicle	The user can see both their personal information and the information of vehicles available on the map. Then, the user can compare the distance shown on the map and click the button called "Rent this Vehicle" to rent a vehicle. Also, the user can enlarge the map to see the locations more clearly.	Pass
Return a Vehicle	After using a vehicle, the user can return the vehicle. Firstly, the user can see the information of the vehicle, if they click a button called "End Current Ride", the app will stop billing and the vehicle will stop either.	Pass
Pay Charges	The user can pay the bill displayed on the menu by clicking on the button called "Pay" after checking out that the information of this trip shown on the menu is right.	Pass
Report a Vehicle as Defective	The user can report something wrong about the trip by typing in the textbox below. After clicking a button called "Report a fault and stop rental" the trip will be stopped and our engineer will check the problem.	Pass
Track a Vehicle	The operator finds the vehicle information by clicking the track button and displays it in the GUI text box	Pass

Move a Vehicle	The operator changes the vehicle location information through the vehicle ID, first displays the original location of the vehicle, and redisplay the vehicle location information through the move function.	Pass
Charge a Vehicle	The operator vehicle for charging operation, first will find the vehicle power less than 20 vehicle id and charging operation, if not will display all vehicles in good condition, finally display all vehicle information.	Pass
Repair a Vehicle	The operator vehicle for repair operation, first will find the vehicle in a broken state vehicle id and repair, if not will display all vehicles in good condition, and finally display all vehicle information.	Pass
Manager Login	If the Manager enters Username and password correct it will redirect him/her to a dashboard and pop up with an error for incorrect credentials..	Pass

Table 6.3: Test Table demonstrating the test descriptions and results of pertinent functions of our application

Chapter 7 Conclusion & Future Work

In summary, this project has successfully achieved the goals set in Chapter 3. We've created functions that allow customers to rent a car at any place in the city where a working vehicle is available, return a vehicle to any location, charge their accounts based on the length of the rental and the kind of vehicle rented, and report a defective vehicle. For the Operator component, we incorporated the ability to track the position of all vehicles in the city, charge a vehicle when its battery is exhausted, repair a faulty vehicle, and move vehicles to new city sites as needed. Lastly, for the management portion, we've created a dashboard to generate reports displaying all vehicle operations over a specified time period, and more using appropriate data visualisation approaches.

There are a plethora of possibilities for future work. If we had more time to continue the project, we would:

- Research another GUI framework/toolkit instead of PyQt as from our experience, its response time is too long.
- Create a web-application with the Django framework, or better yet, create a smartphone application since it is the most reasonable way one would hire a vehicle for rent.
- Implement QR code functionality to start and end rides
- Use a map API to show best route to reach available vehicles
- Explore safety features by imposing automatic speed limits by using location data from GPS
- Integrate payment gateways to offer multiple secure modes of payment
- Conduct surveys to measure performance of our product and make necessary improvements

In conclusion, working on this project with new team members and learning new technologies and programming languages was an exciting challenge. This project has been rewarding and enlightening to work on.

References

- [1] Robison, R. (2010, October). Electric Vehicles. *Houses of Parliament, Parliamentary Office of Science & Technology*.
https://www.parliament.uk/globalassets/documents/post/postpn365_electricvehicles.pdf
- [2] Zacks Equity Research. (2019, June 7). Big Firms Are Investing in Rented E-Scooters. Here's Why. *Yahoo! Finance*.
<https://finance.yahoo.com/news/big-firms-investing-rented-e-125112398.html>
- [3] White Label Fox. (2022, February 28). *Top Electric Scooter Rental Sharing Companies in 2022*. White Label Fox.
<https://whitelabelfox.com/top-electric-scooter-rental-sharing-in-2022>
- [4] Editorial Team. (2020, November 13). *Uber-rival Bolt to invest €100M+ in European e-scooter market; plans to expand to 100+ cities next year*. Silicon Canals.
<https://siliconcanals.com/news/startups/bolt-announces-plans-become-europes-largest-micro-mobility-provider/>
- [5] CyclePlan. (2021, March 31). *The Top 7 Electric Scooter Apps Of 2021 - Cycle Savvy | The Cycleplan Blog*. Cycleplan.
<https://www.cycleplan.co.uk/cycle-savvy/the-top-7-electric-scooter-apps-of-2021/>
- [6] Attwood, J. (2022, July 7). *Rental e-scooter trials to be extended until May 2024*. Move Electric.

<https://www.moveelectric.com/e-scooters/rental-e-scooter-trials-be-extended-until-may-2024>

- [7] Mehta, P. (2022, January 31). *What is Software Development Methodology?* Positiwise.

<https://positiwise.com/blog/what-is-software-development-methodology/>

- [8] Iordanidis, J. (2021, March 14). *8 Software Development Methodologies Explained.* Easy Agile.

<https://www.easyagile.com/blog/software-development-methodologies/>

- [9] Inflectra. (2014, June 1). *Guide To Kanban In Software Development.* Inflectra Corporation.

<https://www.inflectra.com/Methodologies/Kanban.aspx>

- [10] Radigan, D. (2016, August 23). *Kanban - How the Kanban Methodology Applies to Software Development.* Atlassian.

<https://www.atlassian.com/agile/kanban>

- [11] Schwaber, K. (2022, October 28). *What is Scrum?* Scrum.org.

<https://www.scrum.org/resources/what-is-scrum>

- [12] Cohn, M. (2010, January 10). *Scrum Overview: Agile Software Development.* Mountain Goat Software.

<https://www.mountaingoatsoftware.com/agile/scrum/resources/overview>

- [13] Lvivity Team. (2018, February 7). *The Most Popular Software Development Methodologies Overview*. Lvivity.
<https://lvivity.com/software-development-methodologies>
- [14] LucidChart. (2016, August 5). *What is an Entity Relationship Diagram (ERD)?* Lucidchart.
<https://www.lucidchart.com/pages/er-diagrams>
- [15] Rohn, S. (2022, July 28). *What Is a Desktop Application? +Challenges, Use Cases*. Whatfix. <https://whatfix.com/blog/desktop-application/>
- [16] Kuchling, A. M. (2021, February 9). *PEP 206 – Python Advanced Library*. Python Enhancement Proposals.
<https://peps.python.org/pep-0206/>
- [17] SQLite. (2021, October 6). *About SQLite*. SQLite.
<https://www.sqlite.org/about.html>
- [18] Lynn, S. (2017, December 12). *The Pandas DataFrame – loading, editing, and viewing data in Python*. Shane Lynn.
<https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/>
- [19] Srivastava, A. (2021, January 1). *Data Visualization with Matplotlib*. Medium.
<https://medium.com/analytics-vidhya/data-visualization-with-matplotlib-4a962cb94a3d>
- [20] VanderPlas, J. (2016). Chapter 4. Visualization with Matplotlib. In *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, Incorporated.

<https://www.oreilly.com/library/view/python-data-science/9781491912126/ch04.html>

- [21] Parmar, J. (2019, October 15). *Python Matplotlib Tutorial: Plotting Data And Customisation*. QuantInsti's Blog.
<https://blog.quantinsti.com/python-matplotlib-tutorial/>
- [22] Riverbank Computing. (2002, March 31). *What is PyQt?* Riverbank Computing. <https://riverbankcomputing.com/software/pyqt/>
- [23] PennState, Department of Geography. (2018, March 23). *2.5.2.1 PyQt vs. PySide*. PennState College of Earth and Mineral Sciences, Department of Geography.
<https://www.e-education.psu.edu/geog489/node/2225>
- [24] Software Testing Help. (2022, October 25). *Pytest Tutorial – How To Use Pytest For Python Testing*. Software Testing Help.
<https://www.softwaretestinghelp.com/pytest-tutorial/>

Appendix

Diagram 1. Use Case Diagram

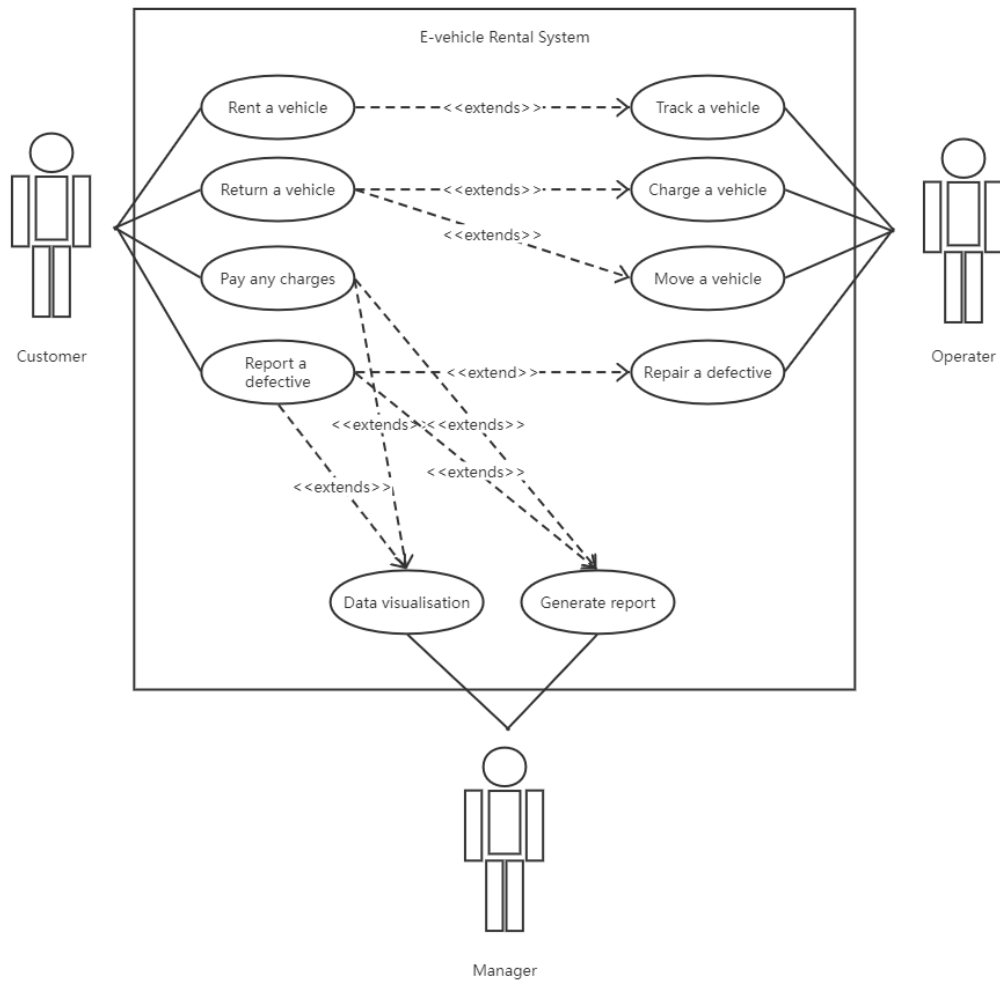


Diagram 2. UML Diagram

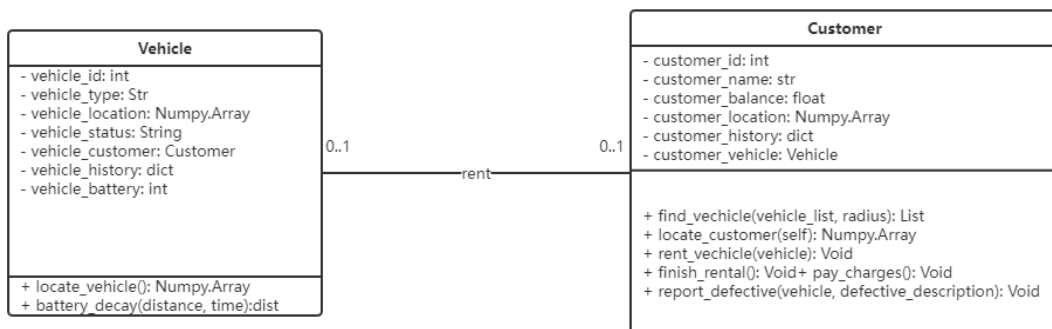


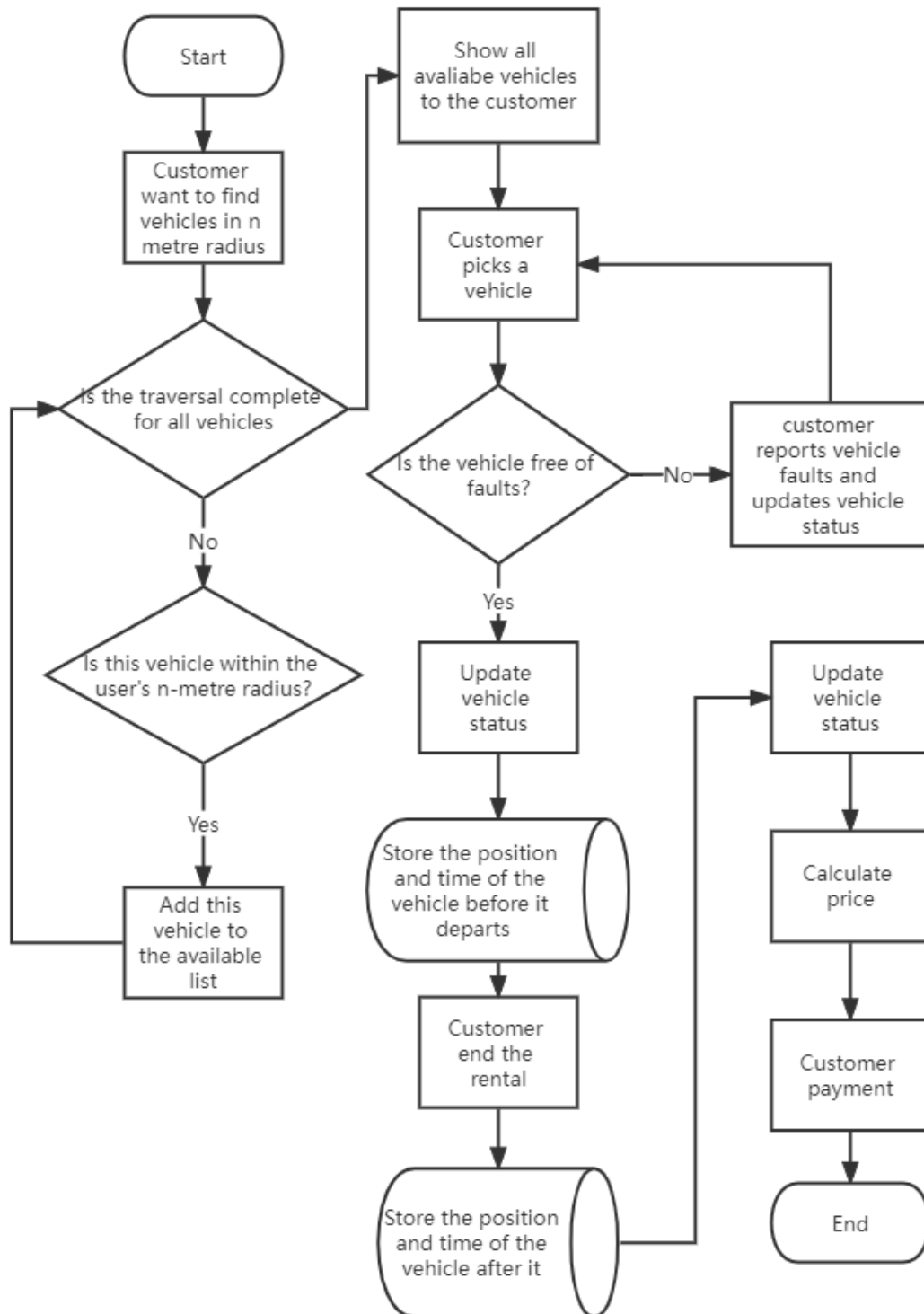
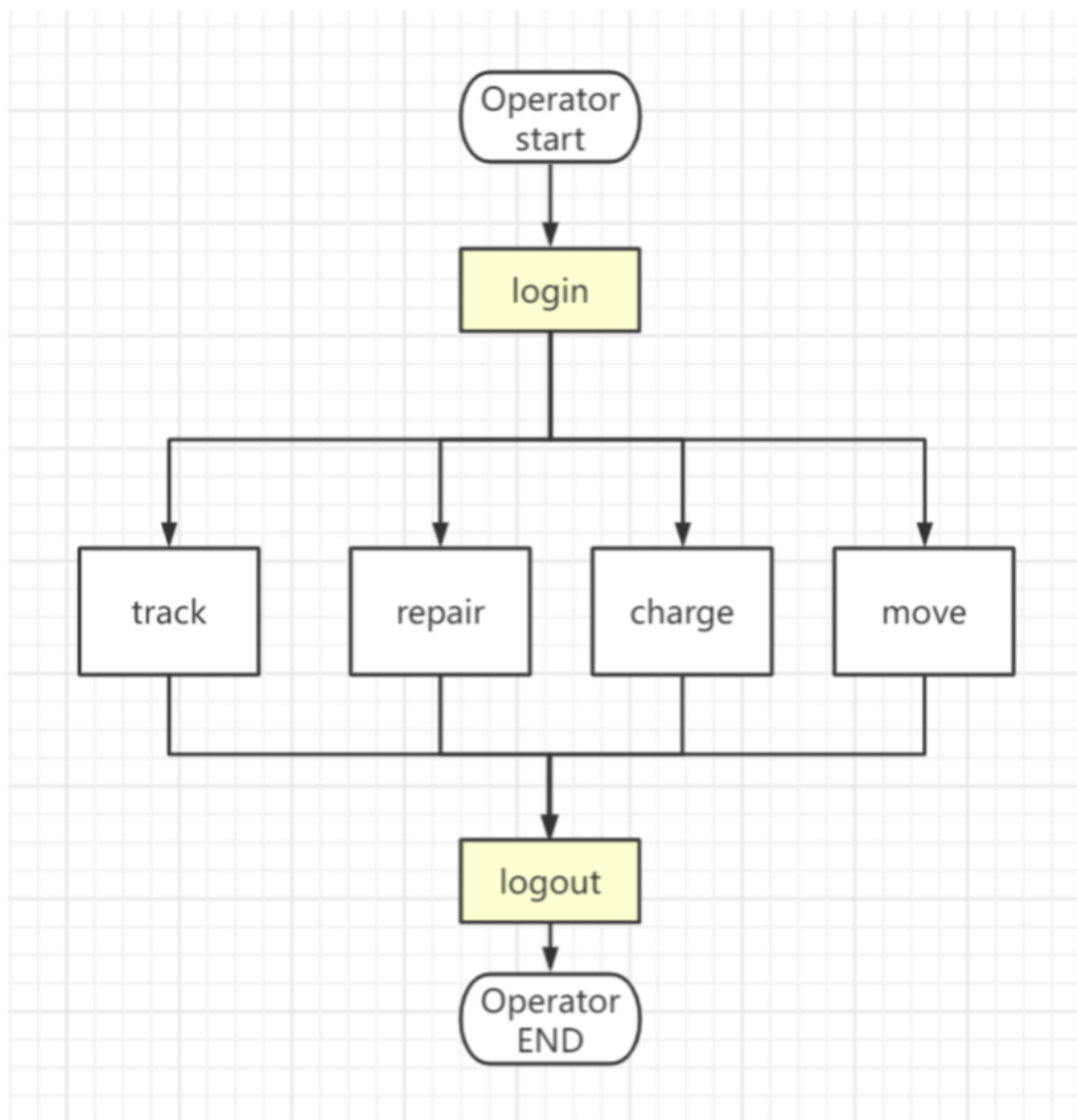
Diagram 3. Flow Diagram (Customer)

Diagram 4. Flow Diagram (Operator)



Electronic Vehicle Sharing System

USER GUIDE

Table of Contents

Chapter 1	Customer	
1.1	Customer Login	1
1.2	Rent a Vehicle	2
1.3	Return a Vehicle	3
1.4	Report a Vehicle	4
1.5	Pay Charges & Dues	5
Chapter 2	Operator	
2.1	Track Vehicle	7
2.2	Move Vehicle	7
2.3	Charge Vehicle	8
2.4	Repair Vehicle	8
Chapter 3	Manager	
3.1	Manager Login	9
3.2	Manager Dashboard	9
3.3	Usage Statistics	11
3.4	More Options	13

1 Customer

1.1 Customer Login:

This login page is the initial element that catches the user's attention. The user is required to provide a username and password before clicking the "login" button.

If the user enters an erroneous username or password, "Incorrect Account or Password" will appear underneath the password text field to inform the user.

If the user successfully logs in, the primary window for renting an E-Vehicle will be displayed.

The figure displays two states of the 'E-Vehicle Rental System' login window. Both windows have a title bar labeled 'MainWindow' and standard minimize, maximize, and close buttons. The window content is titled 'E-Vehicle Rental System'.

Top Screenshot (Initial State):

- Account: [Empty text field]
- Password: [Empty text field]
- Buttons: 'Register' and 'Login' (both disabled/greyed out).

Bottom Screenshot (Failed Login State):

- Account: [Text field containing '123123']
- Password: [Text field containing '123123']
- Error Message: 'Wrong Account or Password' (displayed below the Password field)
- Buttons: 'Register' and 'Login' (both disabled/greyed out).

Figure 1.1: Customer Login Window

1.2 Rent a Vehicle

Upon logging in, the customer is provided with a window that is equally split in half. The left half of the window is created using HTML and it displays an interactive map interface that shows the location of the customer (red pin) and the location of the different vehicles (blue pins) across the city. The customer can zoom in and out of the map using the zoom buttons on the top-left. They can hover their mouse pointer over the blue pins on the map to view the type of vehicle and its availability status.

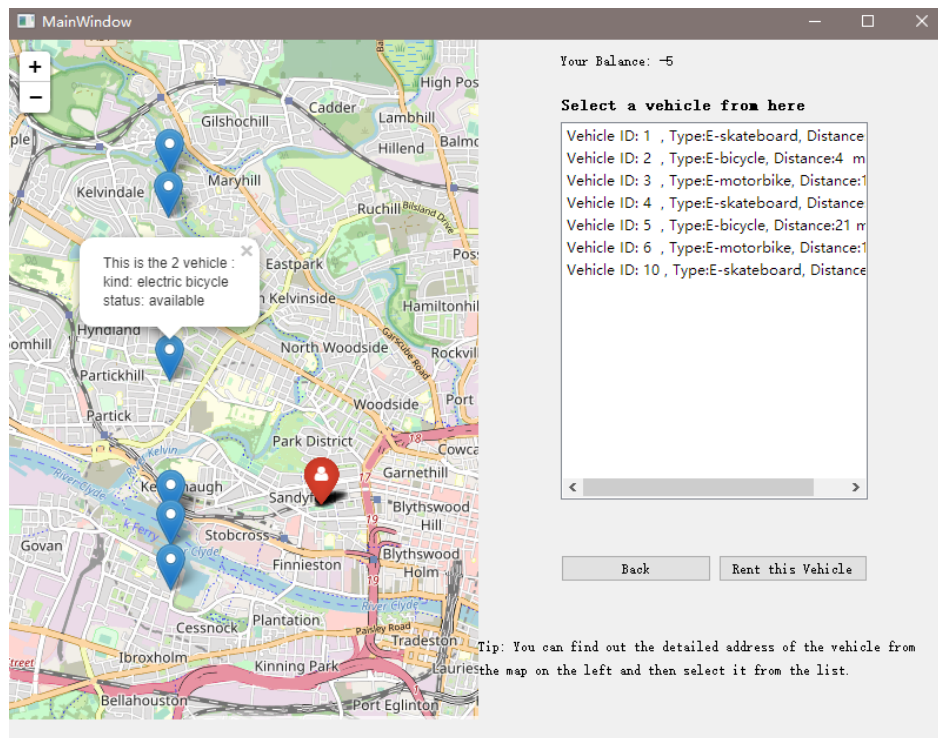


Figure 1.2a: Rent a Vehicle Window

The right side of the window is implemented using PyQt and it displays a list of the available vehicles sorted by vehicle ID and also shows the distance from the user's location. If the customer decides which vehicle they want to rent, they can select that item from the list on the right-hand side of the window and click on the *Rent This Vehicle* button. The specific customer's available account balance is displayed just above the list on the right side.

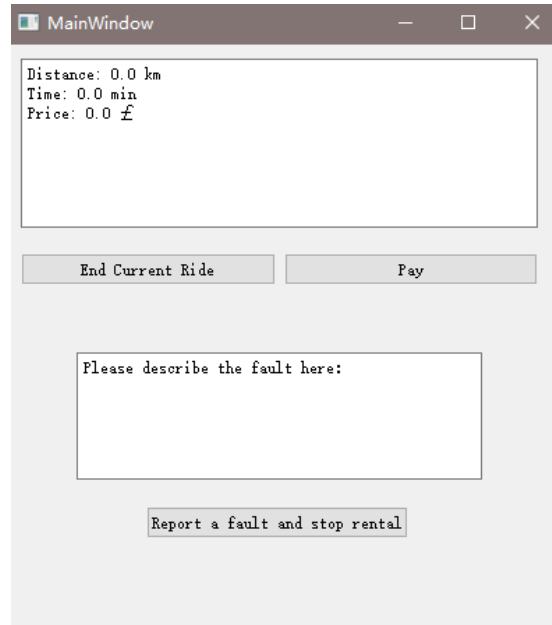


Figure 1.2b: Interface after selecting a vehicle to rent

1.3 Return a Vehicle

Once the customer selects the *Rent This Vehicle* button, their selected vehicle will be displayed as unavailable for rent for other users and this availability change will take effect at the database level. Upon pressing the *Rent This Vehicle* button, a new window appears that calculates the time and distance traveled using the vehicle. Once the customer reaches their destination they'll select the *End Current Ride* button. At this instant, the fare of the ride will be displayed to the customer after information on the total distance traveled and the total time rented. When the user reaches the destination and wishes to end the ride, the user presses the "End Current Ride" button. The text box above will display some information for the customer to check. At this point, the user should leave the vehicle and lock it. After the customer has done this, the customer RETURNS the vehicle. The database will be updated with the new position and status of the vehicle.

The customer can then press the *Pay* button to terminate the current ride.

As soon as they click the *Pay* button:

- the displayed fare will be automatically deducted from their available balance.

- the vehicle that they were renting all this time will now be made available for rent again.
- The initial window for renting a vehicle will be loaded and displayed to the customer to initiate another ride.

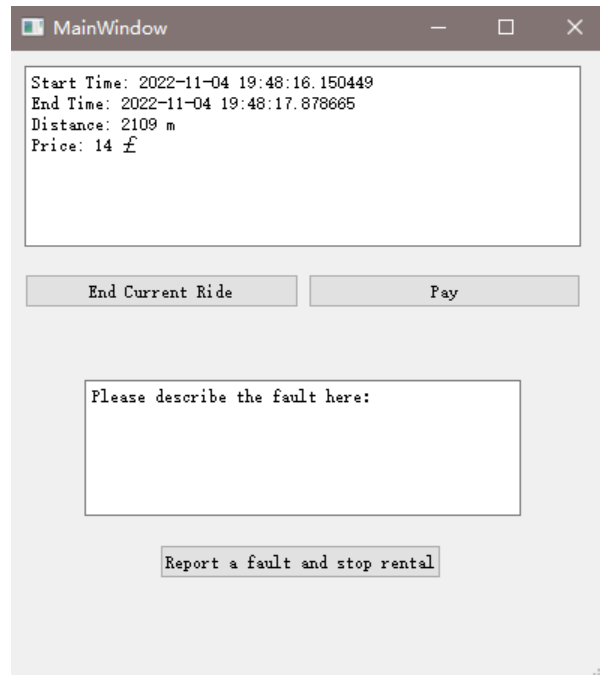


Figure 1.3: Return a Vehicle Window

1.4 Report a Vehicle

The customer has the option of reporting a vehicle as defective. To describe the fault, the customer can write text in the message box and click the *Report a fault and stop Rental* button. Upon doing so,

- The response message acknowledging the feedback will be displayed to the customer
- the window for renting a vehicle will be loaded and displayed to the customer to initiate another ride.
- The reported vehicle will immediately be made unavailable to rent until the operator manually makes it available again.

After a successful report, if the customer doesn't choose to start the rental, they still click the "End Current Ride" button. However, this time the cost is 0 pounds. The

user continues to press "Pay" to return to the previous window and the customer's balance is not deducted during this process.

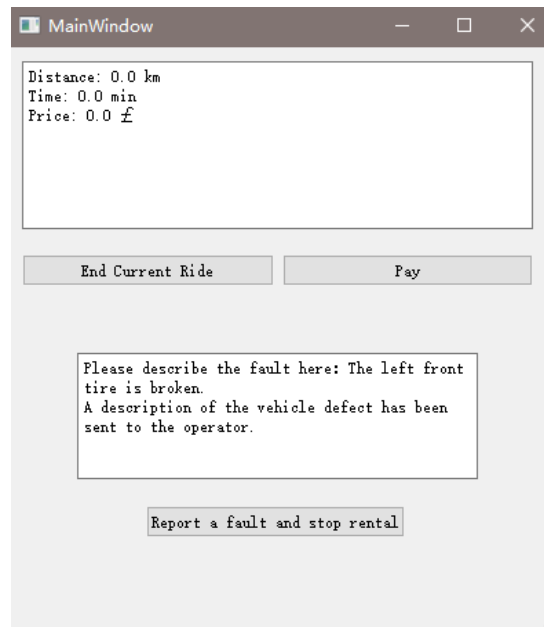


Figure 1.4: Report a Vehicle Window

1.5 Pay Charges and Dues Automatically

The system will let any user rent a vehicle regardless of their available balance. If a customer ends a ride and it happens that their fare is more than their available balance, the difference will be shown in the available balance as a negative balance

Your Balance: -5

The next time the customer rents a vehicle, their outstanding balance will be added to their new fare, and the total amount will be automatically deducted from their available balance and resented to the customer in the receipt. This “rent now, pay later” functionality allows users to rent a vehicle in emergencies with the promise of payment in the subsequent ride.



Figure 1.5: Pay Dues Window

2 Operator

2.1 Track Vehicles

A click on the Track option will display the current location of all the vehicles in the city with the Postcode, latitude, and longitude information

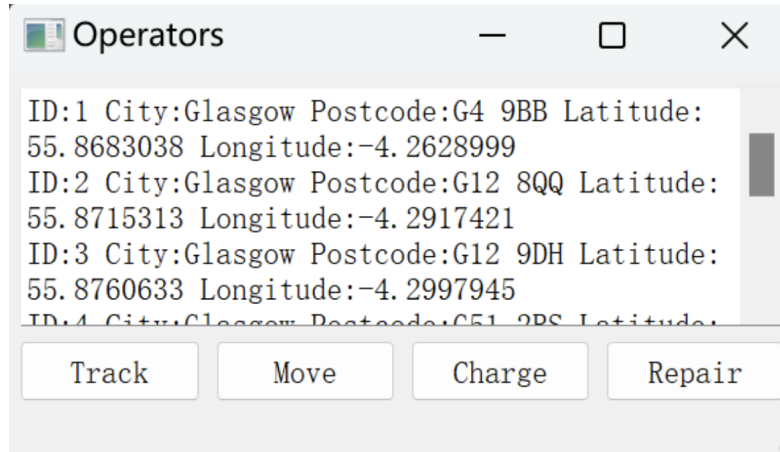


Figure 2.1: Operator Interface after clicking the *Track* option

2.2 Move Vehicles

Clicking on the *Move* option will reposition a vehicle from the original position to a new one by re-adjusting its latitude and longitude values but it will not change its ID. If the vehicle moves successfully, it will show the information in the textbox.

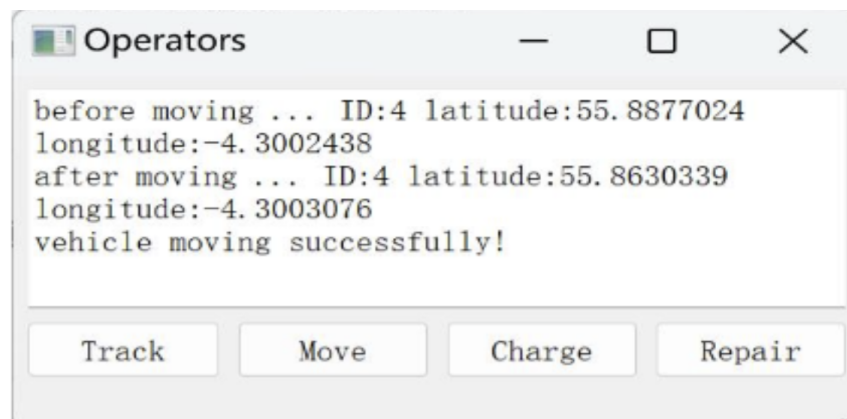


Figure 2.2: Operator Interface after clicking the *Move* option

2.3 Charge Vehicles

Operators can charge all vehicles which need to be charged and if there are no vehicle need to charge, it will display 'all vehicles are full E_amount'
If charging the vehicle is done successfully it will also show the information in the textbox.

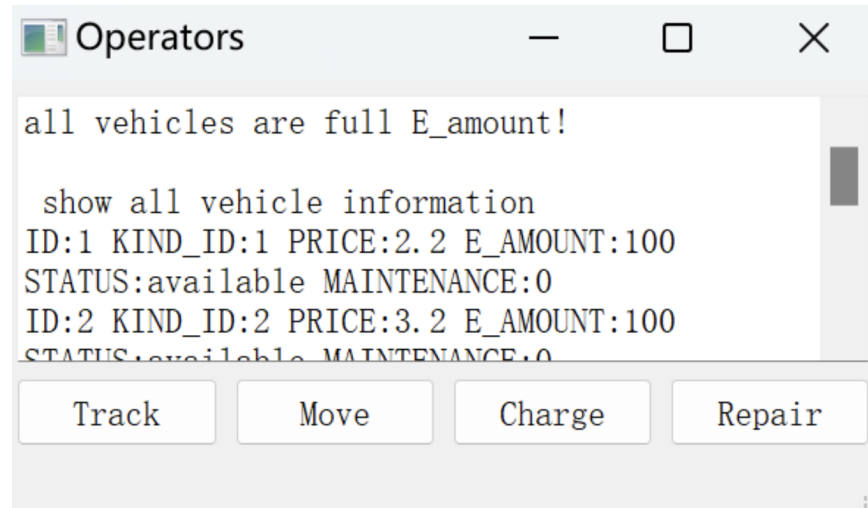


Figure 2.3: Operator Interface after clicking the *Charge* option

2.4 Repair Vehicles

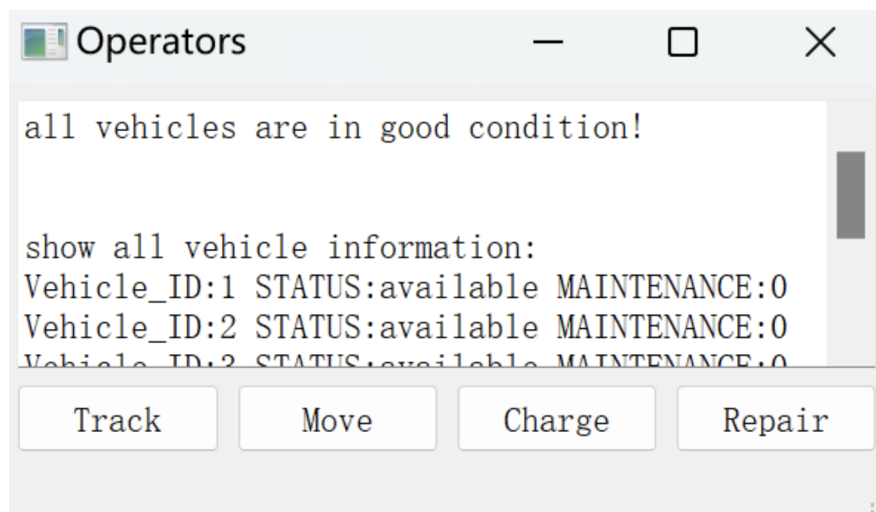


Figure 2.4: Operator Interface after clicking the *Repair* option

3 Manager

3.1 Manager Login

The Manager can only login with the correct information, otherwise the following pop-up dialog box will appear.

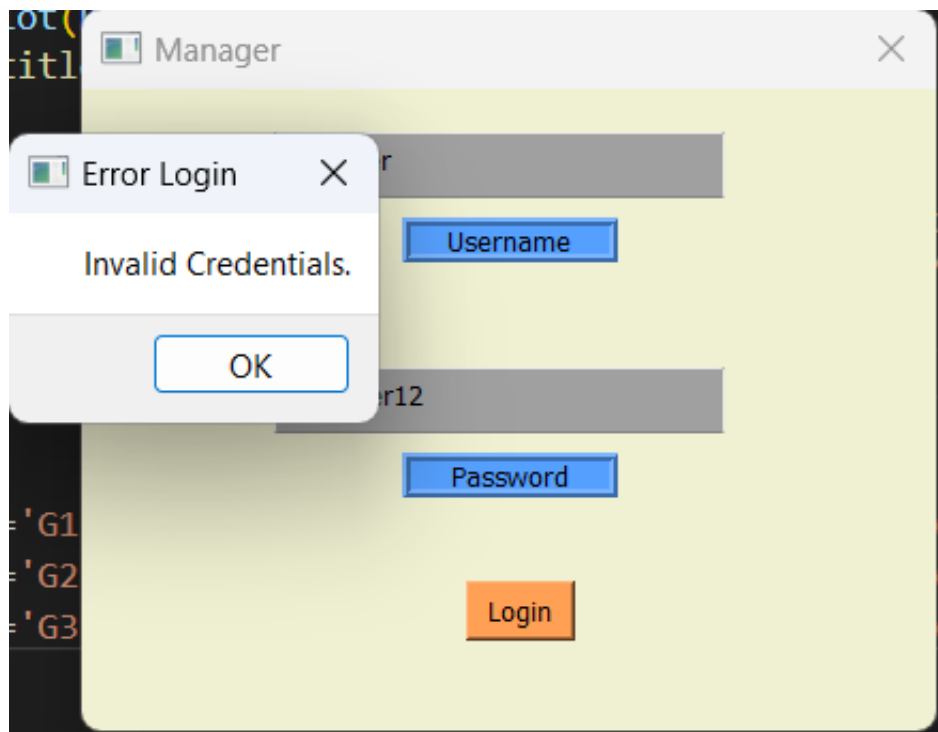


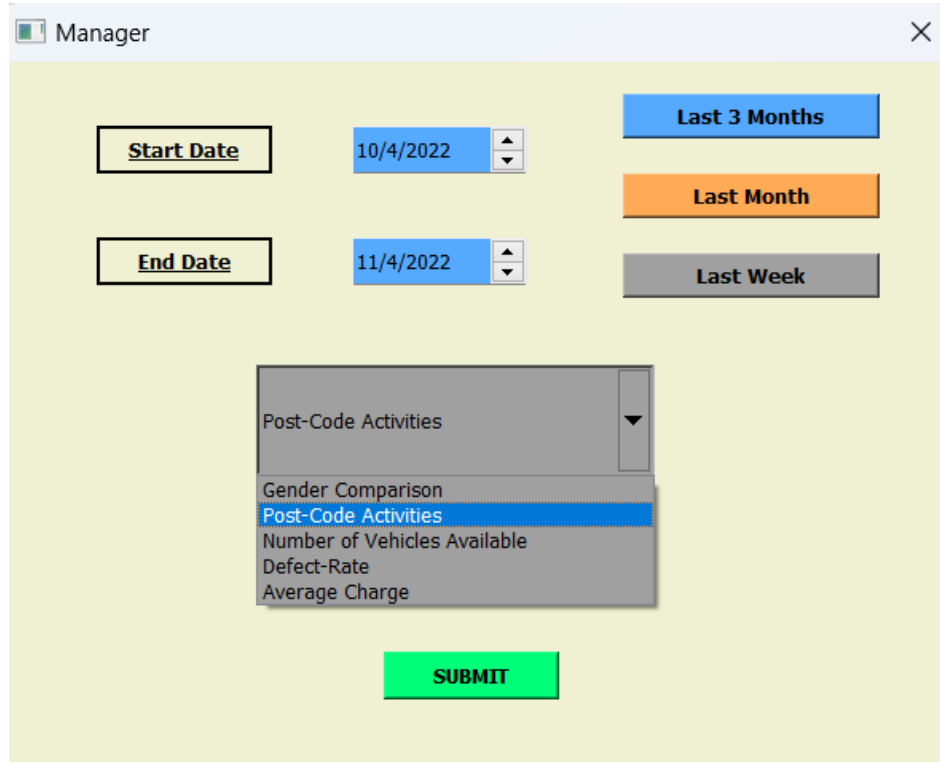
Figure 3.1: Manager Login Window

3.2 Manager Dashboard

Upon successful login, the manager will be displayed a dashboard with a variety of operations to choose from. On the right-hand side, the manager is given the options to quick-select a period of time to generate usage statistics in the form of *Last 3 Months*, *Last Month* or *Last Week*. If the manager would like to specify a different period of time, they could put in the start date and end date manually. Once the manager has selected their desired date(s), they can click on the drop-down menu in the centre of the window to select the type of usage statistics that they would like to see and then press the *Submit* button. The current available options are:

1. *Gender Comparison*: view rental statistics of the genders
2. *Post Code Activities*: view usage statistics based on city post codes
3. *Number of Vehicles Available*: view availability of vehicles

4. *Defect Rate*: view the customer-reported defect rate
5. *Average Charge*: view the average battery-charge of the different vehicles



The image shows a web application window titled "Manager" with a close button (X) in the top right corner. The interface has a light yellow background. On the left, there are two date selection fields: "Start Date" and "End Date". The "Start Date" field shows "10/4/2022" and the "End Date" field shows "11/4/2022". To the right of these fields are three buttons: "Last 3 Months" (blue), "Last Month" (orange), and "Last Week" (gray). Below these buttons is a dropdown menu labeled "Post-Code Activities" with a downward arrow. The dropdown menu is open, showing a list of options: "Gender Comparison", "Post-Code Activities" (highlighted in blue), "Number of Vehicles Available", "Defect-Rate", and "Average Charge". At the bottom center of the interface is a green "SUBMIT" button.

Figure 3.2: Manager Dashboard

3.3 Usage Statistics

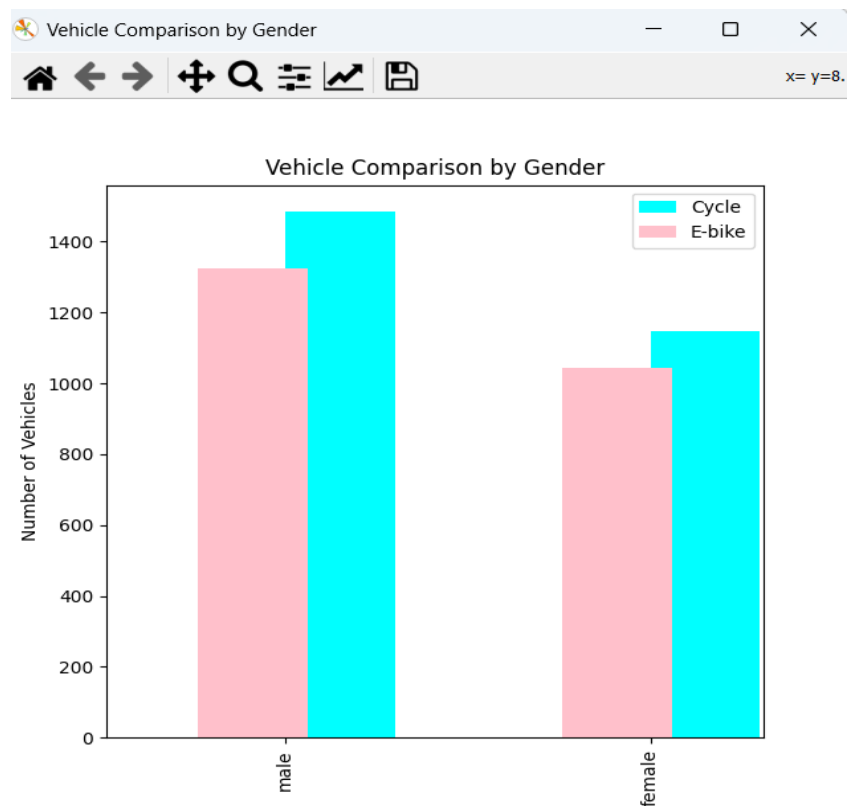


Figure 3.3a: Vehicle Comparison by Gender

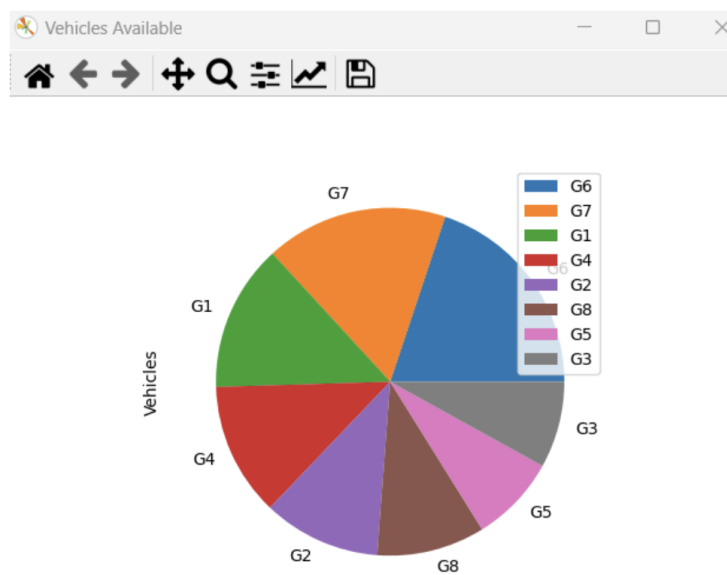


Figure 3.3b: Available Vehicles

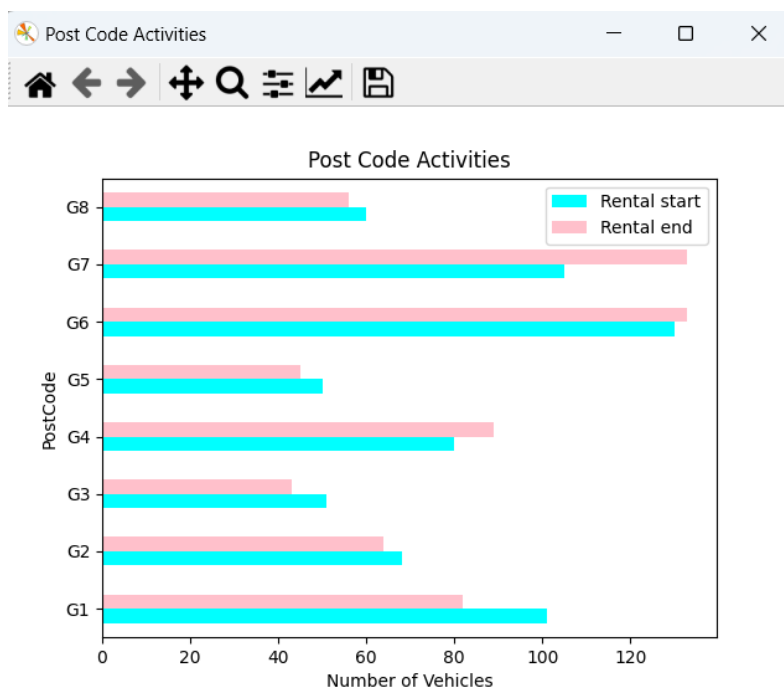


Figure 3.3c: Post-Code Activities

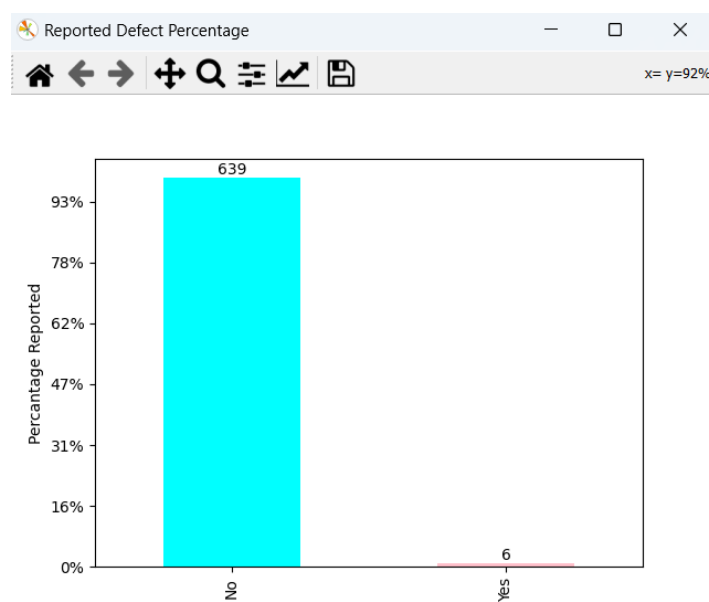


Figure 3.3d: Customer-reported Defect rate

3.4 More Options

It is possible for the manager to save any report to their hard-disk in an image format by clicking on the *Save* button located on the far-right of the top menu:



Figure 3.4a: Save button

If the manager would like to modify the graphs/pie-charts they can do that by clicking on the *Edit* button below. It will open a new window with customization options of changing the title, and changing the scale of the axes.

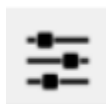


Figure 3.4b: Edit button

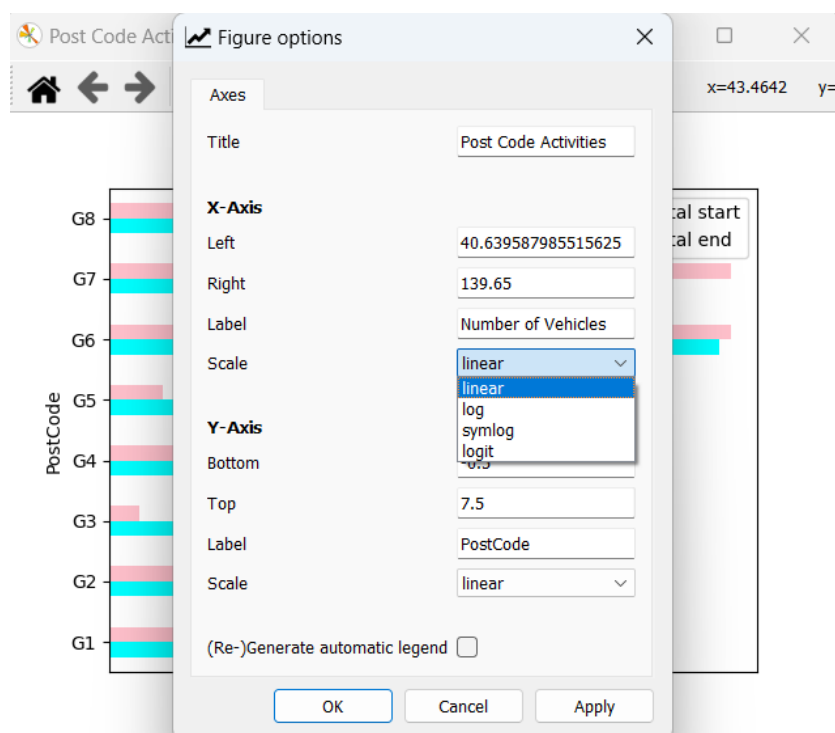


Figure 3.4c: Customization Options

END