<E-Vehicle Share System>

<Jiachen Dong, Shuhao Zhang, Xinyu Guan, Zhuoyu Tian, Zichi Wang, Akash Ananda Kumar Murali, Ran He>

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

<2022.11.7>

Abstract

Electric vehicles are a green and clean means of transport and in recent years the number of shared electric vehicles in supermarkets has been increasing. How to effectively manage electric vehicles, improve their utilization and properly maintain each one has become a very important issue, which is an important prerequisite to ensure the sustainable development of the electric vehicle industry. This paper develops an electric vehicle sharing system using python language and sqlite3 database technology. The purpose of this system is to facilitate the management of electric vehicles by making it easier for users to rent and return the vehicles and for managers to keep track of the location and usage of each vehicle.

Contents

1. Introduction

With the development of the sharing economy, bicycle sharing is a new form of transport that is now very popular and promoted by everyone. It is more flexible and cheaper than buses and taxis. Especially for short trips, people can hire bicycles at any time to go to various destinations according to their itinerary. Compared with cars, it is also an environmentally friendly mode of transport, with benefits for the health of the population, air quality and the urban road infrastructure network (Pelechrinis, et al., 2017). So in this context we wanted to develop an efficient, lightweight bike-sharing rental system that would respond to this new form of healthy transport.

Our project is mainly developed using the python language, with a SQLite database. There are three roles in the system, they are: in this system to achieve the user car rental car return, operator repair and other functions, manager data visualization, in this article, a detailed description of each function and the logic behind each function and design ideas, in addition at the end of the article we analyse the shortcomings of our project and the technologies and features that need to be improved in the future.

2. Background Information

In our research, we referred to the lime(2022) bike share website to understand the key customers-side functions and how the system works. Customers function is mainly to view nearby vehicles and then select the type of vehicle to be rented. After renting a bike you have to settle the order and top up when the balance is low. When a bike is broken it can be reported and can be repaired by the manager in time. The website also gave us some real life rental information such as rental prices.

We also refer to Chinese bike-sharing systems, such as Mobay and ofo, which use GPS positioning and online payment for renting and returning bikes. Due to technical limitations, we refer to the logic in which the location is written into the database for positioning and the user account is also written into the database. For the price algorithm, ofo team developed a dynamic price algorithm based on bike distribution, riding route and rental time. It also served as a reference for our price calculation (Xiaoze, et al., 2017).

3. Requirement

The project aims to design an electric car sharing system with several key function:

(1) Users can rent and return shared electric vehicles anywhere in the city and pay for them according to the type of shared electric vehicle the user has rented and the duration of use. Users can report to the system when they notice a vehicle breakdown.

(2) The operator can track the vehicle's location in the city, recharge the vehicle when its battery runs out, fix the vehicle reported by the user as faulty and transfer the vehicle to a different location in the city depending on the need.
(3) Managers need to produce reports on the activity of all vehicles over a period of time based on all vehicle data.

4. Technical Introduction

4.1 Python

As a team project, we had to choose a computer language that all group members would be able to use well. So compared to Java, all the team members found python to be more concise and easier to use. So we chose python as the computer language for our project.

Python(2022) is a object-oriented, interpreted, and interactive programming language. Python has dynamic data types such as modules and classes. There are many library interfaces that are an easy to extend. It is one of the most popular computer languages in the world today.

4.2 SqLite3

We chose SQLite because it is the database that our teacher provides in the class material and we can get help from our teacher when we have difficulties. And it is free. SQLite(2022) is a small, fast, full-featured SQL database engine. It has a very large number of users. There are over 1 trillion SQLite databases in use.

5. Database Design

5.1 Data Dictionary

Table 5-1 Vehicles

Field name	Field type	Length	Primary key or not	Can be empty
vehicle_id	INT	INFINITE	Y	Y
vehicle_type	INT	INFINITE	N	Y
battery	INT	INFINITE	N	Y
vehicle_status	INT	INFINITE	N	Y
earned_money	INT	INFINITE	N	Y
total_time	Datatime	INFINITE	N	Y
vehicle_use	INT	INFINITE	N	Y
location_x	INT	INFINITE	N	Y
location_y	INT	INFINITE	N	Y
place_name	TEXT	INFINITE	N	Y

Table 5-2 Customers

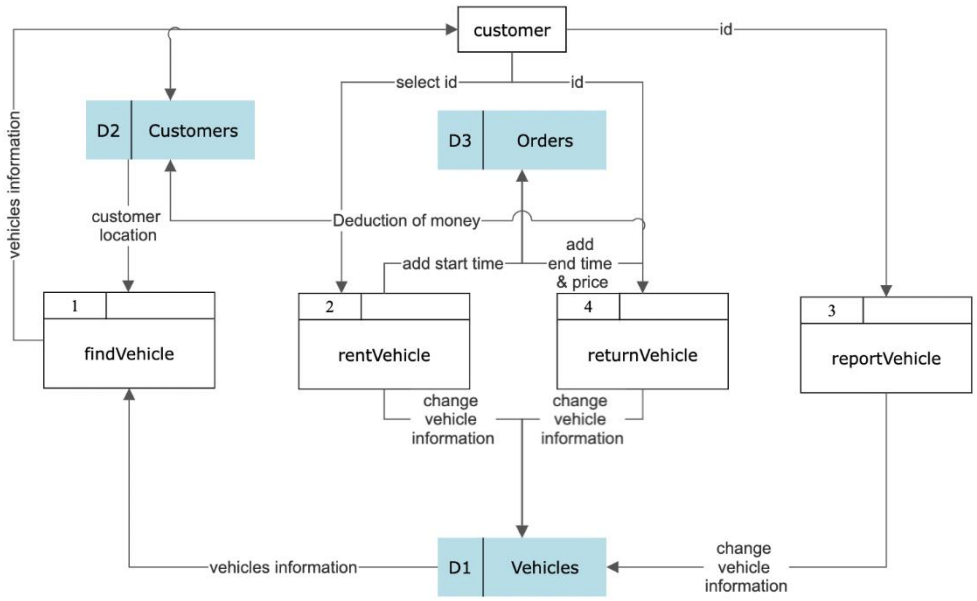
Field name	Field type	Length	Primary key or not	Can be empty
customer_id	INT	INFINITE	Y	Y
accout_money	INT	INFINITE	N	Y

Table 5-3 Order

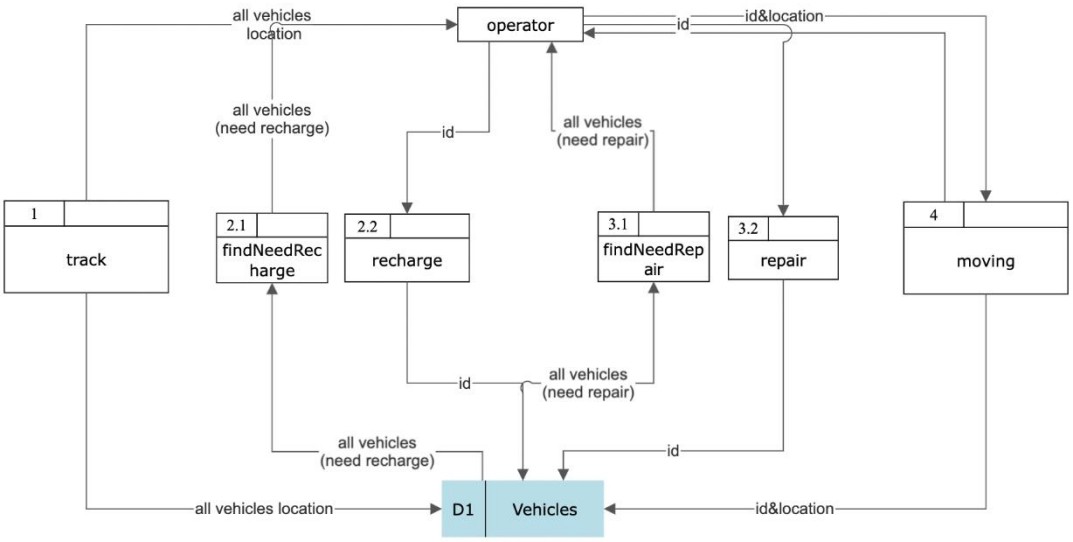
Field name	Field type	Length	Primary key or not	Can be empty
id	INT	INFINITE	Y	Y
vehicle_id	INT	INFINITE	N	Y
vehical_start_time	INT	INFINITE	N	Y
vehical_end_time	INT	INFINITE	N	Y

5.2 Data flow diagram

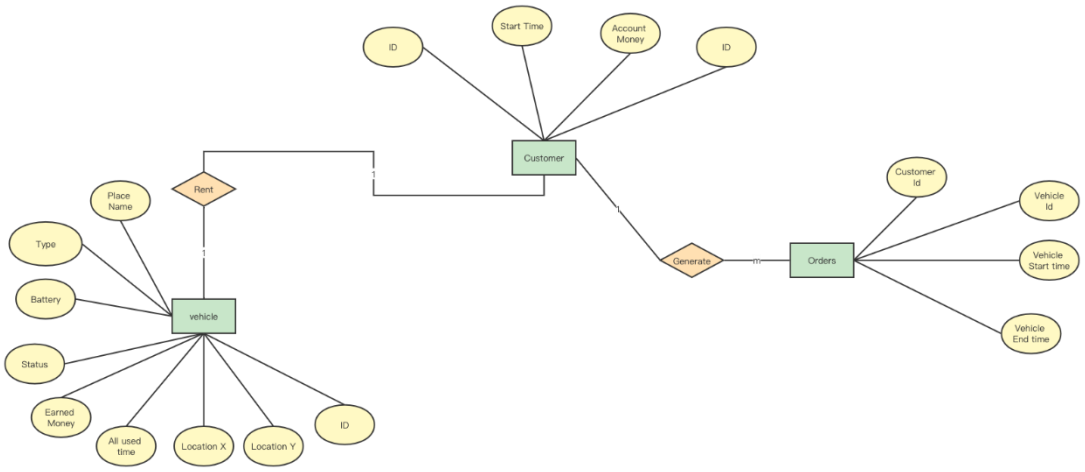
Customers:



Operators:



5.3 E-R diagram



6. Implementation

We are using python statements to implement a shared bike rental system. The database used is SQLite3. In the following pages we will present the functionality and key code for each section in the order of the three roles of customers, operators, managers. Finally we show the design of the database.

Video_link:
https://drive.google.com/file/d/1_EVpsSYT2Qbkr3FeFnLxYemDkqUTiOkZ/view?usp=share_link

6.1. Customers

After selecting login customers, select the type of car rental. This displays the user area location and coordinates. It also displays information about vehicles and their location for straight line distances less than or equal to 180 unit lengths. As shown in Figure 1 in Appendix A. If there are no vehicles within a distance of 180, information on the location of all vehicles of this type is displayed. As shown in Figure 2 in Appendix A. Here in the user function we mainly use the insert, delete, update and select of the database. For calculating the distance between the user and the vehicle we call the math function. As shown in Figure 3 in Appendix A.

Next, select the rental car id, at this moment the car running status changes to 1 (running) and the timer starts. y is to continue riding the car, the car will continue to automatically move at random location coordinates, and the power is reduced by 5%. As shown in

Figure 4 in Appendix A. If the bike runs out of battery (y count > 20), it will be returned automatically and the running status will change to stopped. As well as displaying the contents of this order and the account balance. As shown in Figure 5 in Appendix A. Power consumption design code, 5 power deductions for each advance (one y). As shown in Figure 6 in Appendix A.

When the customers choice return bike. The running status of the bike changes to 0 (stopped running) and the timing stops. N is for stopping the bike. Shows the ride time, the type of tool used, the account balance before billing, the order price and the account balance after billing. As shown in Figure 7 in Appendix A. When the account balance is low, it will prompt for a minimum amount of money to be topped up. The account balance will be displayed when the top up is complete. As shown in Figure 8, 9 in Appendix A. Select if repairs are required and return to the role selection screen. If n is selected, the vehicle can then continue to be hired. If y is selected, the vehicle's repair status changes to repair required and the user cannot continue to hire until it is repaired. As shown in Figure 10 in Appendix A.

6.2. Operators

Select operators to enter the operators function, which will display the location information of all vehicles. As shown in Figure 11 in Appendix A. These codes implement the creation of vehicle area names based on randomly generated coordinates. As shown in Figure 12 in Appendix A.

Next the id of the vehicle to be repaired is displayed and repaired manually. If there are no vehicles to be repaired it shows that there are no vehicles to be repaired. As shown in Figure 13, 14 in Appendix A.

Next all vehicles that need to be charged are displayed and charged manually. If there are no vehicles to be charged the display shows that there are no vehicles to be charged. As shown in Figure 15, 16 in Appendix A.

Operators can move the specified id vehicle to the specified name location. As shown in Figure 17 in Appendix A. These codes enable the random generation of new vehicle location coordinates within the location name range based on the location name entered. As shown in Figure 18 in Appendix A. Finally the updated position is displayed again. As shown in Figure 19 in Appendix A.

6.3. Managers

Select the managers function. Enter the time period you wish to check. As shown in Figure 20 in Appendix A. Here the current status of the bike movement appears (running, ready to run and in need of repair). As shown in Figure 21 in Appendix A. Here the

current status of the Scooter movement appears (running, ready to run and in need of repair). As shown in Figure 22 in Appendix A. Next, it shows the number of vehicles in use. As shown in Figure 23 in Appendix A. Next, it shows total and average time of bicycle use. As shown in Figure 24 in Appendix A. Next, it shows total and average time of scooter use. As shown in Figure 25 in Appendix A.

7. Testing

TEST	DESCRIPTION/ACTION	EXPECTED RESULT	ACTUAL RESULT
CHOOSE ROLE	The user selects one of the three roles of customers, operators and managers and accesses the corresponding interface.	One of the three roles of customers, operators or managers successfully accesses the corresponding interface.	As expected, passed.
CUSTOMERS: CHOOSE VEHICLES TYPE	Once selected into the customers, the system will ask the customer to select the model they wish to lease and when they have done so, the system will display information on the vehicles currently available for that model.	The system will display all selected model ids and locations.	As expected, passed.
CUSTOMERS: CHOOSE VEHICLE ID	Enter the id of the selected vehicle and if the vehicle is in working order, the customer will successfully rent the vehicle.	The customer will successfully rent the vehicle.	As expected, passed.
CUSTOMERS: CHOOSE VEHICLE ID	Enter the id of the selected vehicle, if the vehicle has been reported for repair or the battery is below 20% the system will indicate that it cannot be rented.	The system is unable to rent out the vehicle and transmits information about the vehicle in question to the	As expected, passed.

		operator	
CUSTOMERS: RETURNING THE VEHICLE	When the customer has finished using the bike, he or she enters "n" into the system to indicate that the bike needs to be returned and the system calculates the time spent on the bike and displays the amount to be paid.	The system will display the time spent on the bike and the amount to be paid.	As expected, passed.
CUSTOMERS: TROUBLESHOOTING	The system asks if the vehicle needs to be reported and if the customer selects that it needs to be reported, the system will send the information about the faulty vehicle to operators	Information about the faulty vehicle is transmitted to operators and the current rental service is ended.	As expected, passed.
CUSTOMERS: TROUBLESHOOTING	The system asks if the car needs to be repaired and if the customer selects that no repair is required, the rental service will be ended.	The system will end this car rental service.	As expected, passed.

TEST	**DESCRIPTION/ACTION**	**EXPECTED RESULT**	**ACTUAL RESULT**
CHOOSE ROLE: OPERATORS	Once selected as an operator, the system will automatically return information on the current location of all vehicles and display information on vehicles in need of repair.	The system displays information on the location of all vehicles as well as information on vehicles in need of repair.	As expected, passed.
OPERATORS: ENTER THE ID OF THE REPAIRED VEHICLE	When the faulty vehicle has been repaired, operators enter the vehicle ID into the system for rental by the customer.	The vehicle can be used again.	As expected, passed.

OPERATORS: Move vehicle position	Operators enter the ID of the vehicle they wish to move into the system and enter the x-axis y-axis coordinates of the new location they are moving to.	The vehicle was successfully moved to the new location.	As expected, passed.
OPERATORS: Display of the new general vehicle information	When there is a change in vehicle information, the system will ask operators if they need to redisplay all vehicle information, if so then all vehicle information will be redisplayed.	The system displays all information about the vehicle.	As expected, passed.

8. Conclusion

In conclusion, we have completed a complete bicycle sharing rental system. On the functional side we completed the various basic functions of customers, operators and managers. On the technical side we used python statements and a SQLite database, using almost all the material given by the teacher in class. When we started designing the program, our group faced a lot of problems such as whether to use java or python and how to present our program. We also encountered many bugs in the process of writing the program and we asked our teacher to give us advice and guidance on our project. At the end of the day, we were able to complete the project through the joint efforts of each member of our team. However, there is still room for improvement in the project, such as making the pages more user-friendly. The vehicle location function could be visualised to give a clearer view of the vehicle's location, and the managers function could add vehicle rental data for each area to allow for more accurate dispatch of vehicles to maximise the benefits. What's more, this project cannot yet implement multi-threading because it is a pure console project, so we should explore how to do multi-threading in future development

Reference

Lime, 2022. URL https://li.me/

Pelechrinis, K., Zacharias, C., Kokkodis, M. & Lappas, T. 2017, "Economic impact and policy implications from urban shared transportation: The case of Pittsburgh's shared bike system", PloS one, vol. 12, no. 8, pp. e0184092-e0184092.

Python, 2022. URL https://wiki.python.org/moin/FrontPage. Online; accessed 1 November 2022.

SQlite, 2022. URL https://www.sqlite.org/index.html. Online; accessed 1 November 2022.

Xiaoze, C., Jingwen, Z. & Changqing, Y. 2017, "The Development of China's Bike-Sharing Business Model - A Case Study About OFO", , eds. M. Kuek, W. Zhang & R. Zhao, St Plum-Blossom Press Pty Ltd, HAWTHORN EAST, pp. 472.

Appendix A

```
Enter a role here(customers, operators, managers) (enter quit to quit the system): customers
Which knid of vehicle that you want to rent? (bike/scooter): bike
Your location is at Kinning Park (452, 280)
These availablie vehicles are near you:
id: 3 location: Cessnock (139, 121)
id: 9 location: Kinning Park (458, 453)
id: 13 location: Hillhead (232, 256)
id: 25 location: Bridge Street (95, 59)
id: 39 location: Kelvinhall (322, 366)
Type an id that you want to rent:
```

Figure1

```
Which knid of vehicle that you want to rent? (bike/scooter): scooter
Your location is at Hillhead (211, 421)
These avaliablie vehicles are near you:
Sorry, there are no vehicles near you. All available vehicles are shown below:
id: 2 location: Bridge Street (57, 140)
id: 4 location: Cessnock (190, 89)
id: 6 location: Kinning Park (488, 471)
id: 8 location: Kelvinhall (331, 497)
id: 10 location: Kelvinbridge (449, 53)
id: 12 location: Kelvinbridge (469, 169)
id: 14 location: Cessnock (173, 59)
id: 16 location: Ibrox (316, 49)
id: 18 location: Kinning Park (432, 482)
id: 20 location: Govan (288, 90)
id: 22 location: Cowcaddens (182, 260)
id: 24 location: Cessnock (113, 196)
id: 26 location: Cowcaddens (116, 334)
id: 28 location: Cessnock (160, 37)
id: 30 location: Buchanan Street (100, 437)
id: 32 location: Kinning Park (454, 282)
id: 34 location: Bridge Street (71, 202)
id: 36 location: Kelvinhall (391, 261)
id: 38 location: Ibrox (388, 34)
id: 40 location: Kelvinhall (310, 360)
```

Figure2

```
print("Your location is at {} ({}, {})".format(place, cus_location_x, cus_location_y))

if rent_vehicle == 'bike':
    cursor.execute("SELECT vehicle_id, place_name, location_x, location_y FROM Vehicles WHERE vehicle_type = 'bike' AND vehicle_status = 0 AND battery > 0")
    print("These availablie vehicles are near you:")
    for x in cursor.fetchall():
        # customers_id = int(str(x)[str(x).find("(") + 1:str(x).find(",")])
        # print("id: {} ({}, {})".format(x[0],x[1],x[2]))
        if math.sqrt(((cus_location_x - cus_location_y) **2) + ((x[2] - x[3]) **2)) <= 180 :
            print("id: {} location: {} ({}, {})".format(x[0],x[1],x[2],x[3]))
```

Figure3

```
Type an id that you want to rent: 9
Do you want to continue? (y/n)
(If you still want to use the vehicle, enter y. If you want to return the vehicle, enter n.)y
Do you want to continue? (y/n) y
```

Figure4

```
Type an id that you want to rent: 3
Do you want to continue? (y/n)
(If you still want to use the vehicle, enter y. If you want to return the vehicle, enter n.)y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
Do you want to continue? (y/n) y
The vehilce battery is zero, it cannot work for now.
The vehicle has been returned, the location is Kelvinbridge (481, 68).
Your account have 5 pounds
You rode 11 seconds, and you rode bike, the time you rode is between 0 to 30 seconds
Your account will be charged 2 pound(s) automatically
Your account have 3 pounds.
Is this vehicle needs to be repaired(y/n)?
```

Figure5

```
bike_battery = int(str(bike_battery)[str(bike_battery).find("(") + 1:str(bike_battery).find(",")])
cursor.execute("UPDATE Vehicles SET battery = ?, location_x = ?, location_y = ?, place_name = ? WHERE vehicle_id = ?",
[bike_battery - 5, location_x, location_y, place, vehicle_id_bike])
db.commit()
```

Figure6

```
Do you want to continue? (y/n) n
The vehicle has been returned, the location is Kelvinhall (321, 312).
Your account have 5 pounds
You rode 39 seconds, and you rode bike, the time you rode is between 30 to 60 seconds
Your account will be charged 4 pound(s) automatically
Your account have 1 pounds.
```

Figure7

```
(If you still want to use the vehicle, enter y. If you want to return the vehicle, enter n.)n
The vehicle has been returned, the location is Kelvinhall (321, 312).
Your account have 1 pounds
You need to pay for your account at least 1 pounds: 1
You rode 4 seconds, and you rode bike, the time you rode is between 0 to 30 seconds
Your account will be charged 2 pound(s) automatically
Your account have 0 pounds.
```

Figure8

```
if rent_vehicle == "bike" and customers_account < charge_bike:
    num = int(input("You need to pay for your account at least {} pounds: ".format(charge_bike - customers_account)))
    cursor.execute("UPDATE Customers SET account_money = ? WHERE customer_id = ?",[num + customers_account, customers_id])
    db.commit()
    cursor.execute("SELECT account_money FROM Customers WHERE customer_id=?", [customers_id])
    for x in cursor.fetchall():
        customers_account = int(str(x)[str(x).find("(") + 1:str(x).find(",")])
```

Figure9

```
Is this vehicle needs to be repaired(y/n)? n
Enter a role here(customers, operators, managers) (enter quit to quit the system):
```

Figure10

```
Enter a role here(customers, operators, managers) (enter quit to quit the system): operators
id:1, type:bike, location: Buchanan Street (96,464)
id:2, type:scooter, location: Hillhead (233,258)
id:3, type:bike, location: Kelvinbridge (481,68)
id:11, type:bike, location: Cowcaddens (156,275)
id:12, type:scooter, location: Kelvinbridge (469,169)
id:13, type:bike, location: Hillhead (232,256)
id:14, type:scooter, location: Cessnock (173,59)
id:15, type:bike, location: Govan (293,116)
id:16, type:scooter, location: Ibrox (316,49)
id:17, type:bike, location: Kelvinbridge (441,200)
id:18, type:scooter, location: Kinning Park (432,482)
id:19, type:bike, location: Bridge Street (40,155)
id:20, type:scooter, location: Govan (288,90)
id:21, type:bike, location: Cowcaddens (193,433)
id:22, type:scooter, location: Cowcaddens (182,260)
id:23, type:bike, location: Govan (263,81)
id:24, type:scooter, location: Cessnock (113,196)
id:25, type:bike, location: Bridge Street (95,59)
id:26, type:scooter, location: Cowcaddens (116,334)
id:27, type:bike, location: Kelvinbridge (491,109)
id:28, type:scooter, location: Cessnock (160,37)
id:29, type:bike, location: Govan (255,10)
id:30, type:scooter, location: Buchanan Street (100,437)
id:31, type:bike, location: Buchanan Street (18,437)
id:32, type:scooter, location: Kinning Park (454,282)
id:33, type:bike, location: Kelvinbridge (489,241)
id:34, type:scooter, location: Bridge Street (71,202)
id:35, type:bike, location: Hillhead (279,386)
id:36, type:scooter, location: Govan (277,165)
id:37, type:bike, location: Hillhead (202,312)
id:38, type:scooter, location: Ibrox (388,34)
id:39, type:bike, location: Kelvinhall (322,366)
id:40, type:scooter, location: Kelvinhall (310,360)
```

Figure11

```python
place_names = ['Bridge Street','Buchanan Street','Cessnock','Cowcaddens','Govan','Hillhead','Ibrox','Kelvinhall', 'Kelvinbridge', 'Kinning Park']

for i in range(20):

    location_start_x = random.randint(0, 501)
    location_start_y = random.randint(0, 501)

    if location_start_x <= 100:
        if location_start_y <= 250:
            place = place_names[0]
        elif 250 <= location_start_y < 500:
            place = place_names[1]
    elif 100 < location_start_x <= 200:
        if location_start_y <= 250:
            place = place_names[2]
        elif 250 <= location_start_y < 500:
            place = place_names[3]
```

Figure12

```
The list below contains the vehicle that should be repaired:
id: 2
id: 9
id: 36
Type the id above one by one which needs to be repaired: 2
Type the id above one by one which needs to be repaired: 9
Type the id above one by one which needs to be repaired: 36
```

Figure13

```
There's no vehicle needs to be repaired.
```

Figure14

```
The list below contains the vehicle's battery which should be charged:
id: 2
id: 5
Type the id above one by one which needs to be charged: 2
Type the id above one by one which needs to be charged: 5
```

Figure15

```
There's no vehicle needs to be charged.
```

Figure16

```
Do you wnat to move any vehicle (y/n)?
You can enter the place among (Bridge Street, Buchanan Street, Cessnock, Cowcaddens, Govan, Hillhead, Ibrox, Kelvinhall, Kelvinbridge, Kinning Park)
If you want to move, you can type y.y
Type an vehicle id which do you want to move (from 1 to 40): 40
Enter the name of the lacation: Hillhead
Do you wnat to move any other vehicle (y/n)? y
Type an vehicle id which do you want to move (from 1 to 40): 39
Enter the name of the lacation: Cessnock
Do you wnat to move any other vehicle (y/n)? n

Do you want to show all the vehicle's location again(y/n)?
```

Figure17

```
move = input("Do you wnat to move any vehicle (y/n)? " + '\n' +
"You can enter the place among (Bridge Street, Buchanan Street, Cessnock, Cowcaddens, Govan, Hillhead, Ibrox, Kelvinhall, Kelvinbridge, Kinning Park)" +
'\n' + "If you want to move, you can type y.")
while move == 'y':
    id = int(input("Type an vehicle id which do you want to move (from 1 to 40): "))
    move_name = input("Enter the name of the lacation: ")

    if move_name == 'Bridge Street':
        location_x = random.randint(0,101)
        location_y = random.randint(0,251)
    elif move_name == 'Buchanan Street':
        location_x = random.randint(0,101)
        location_y = random.randint(251,501)
```

Figure18

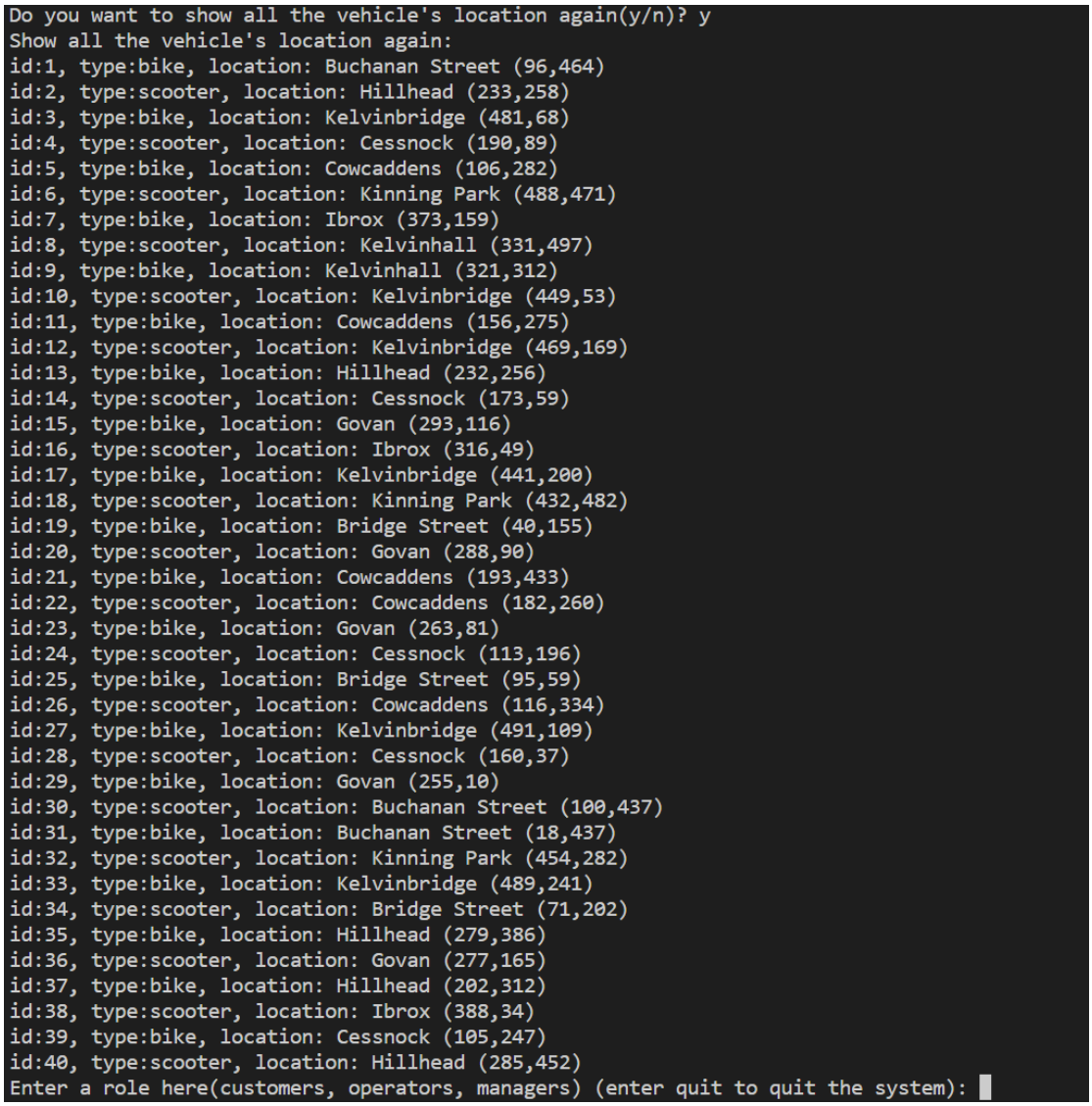


Figure19

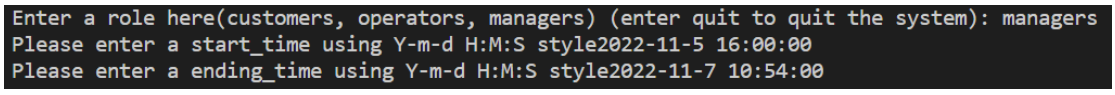


Figure20

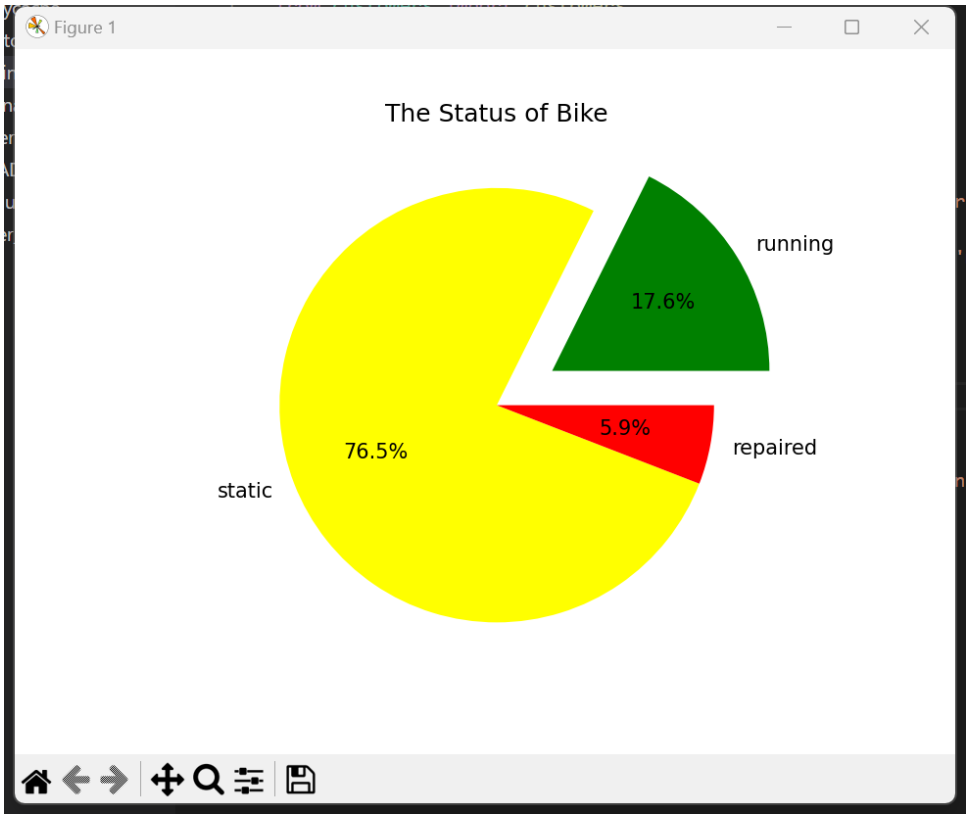


Figure21

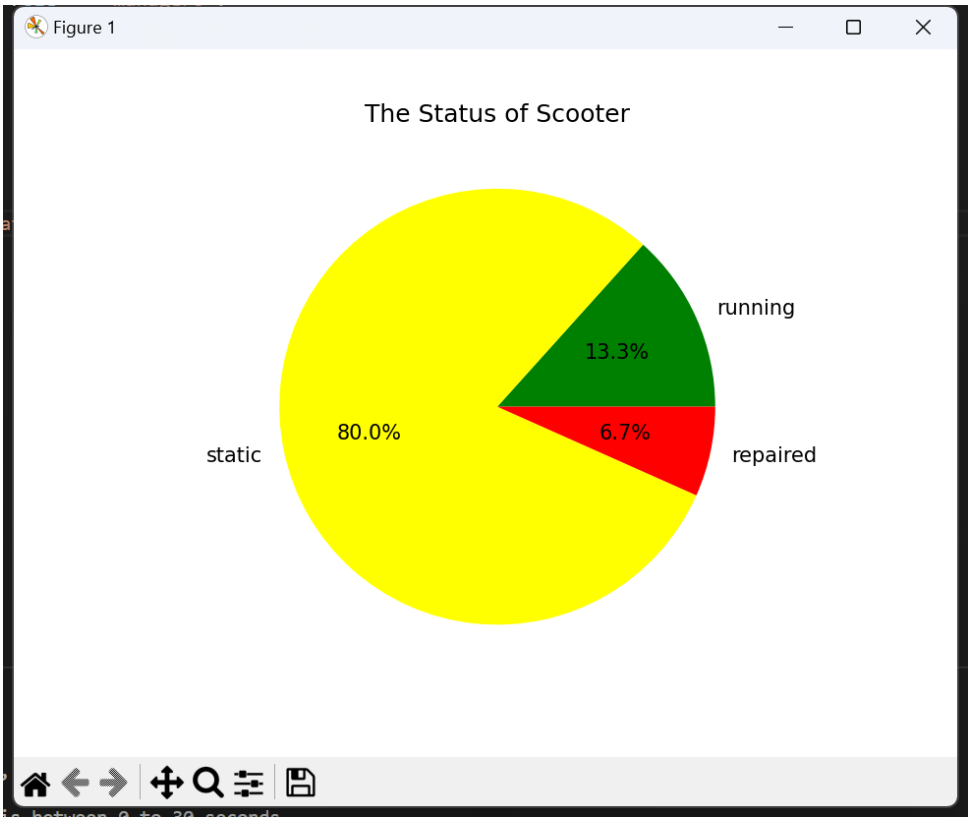


Figure22

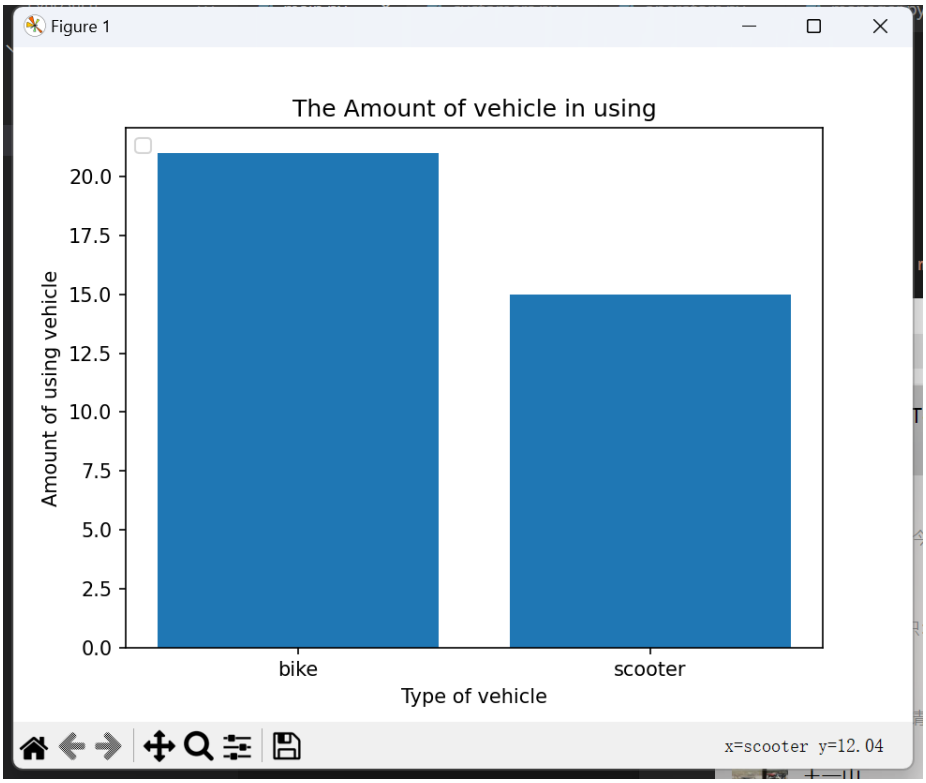


Figure23

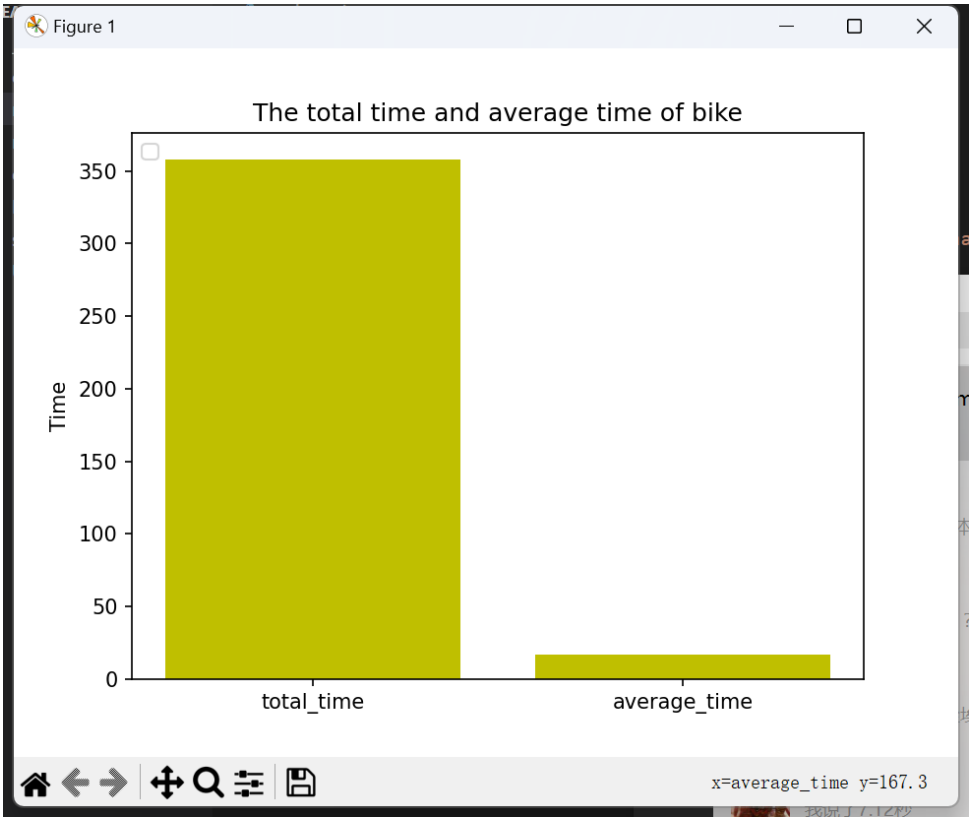


Figure24

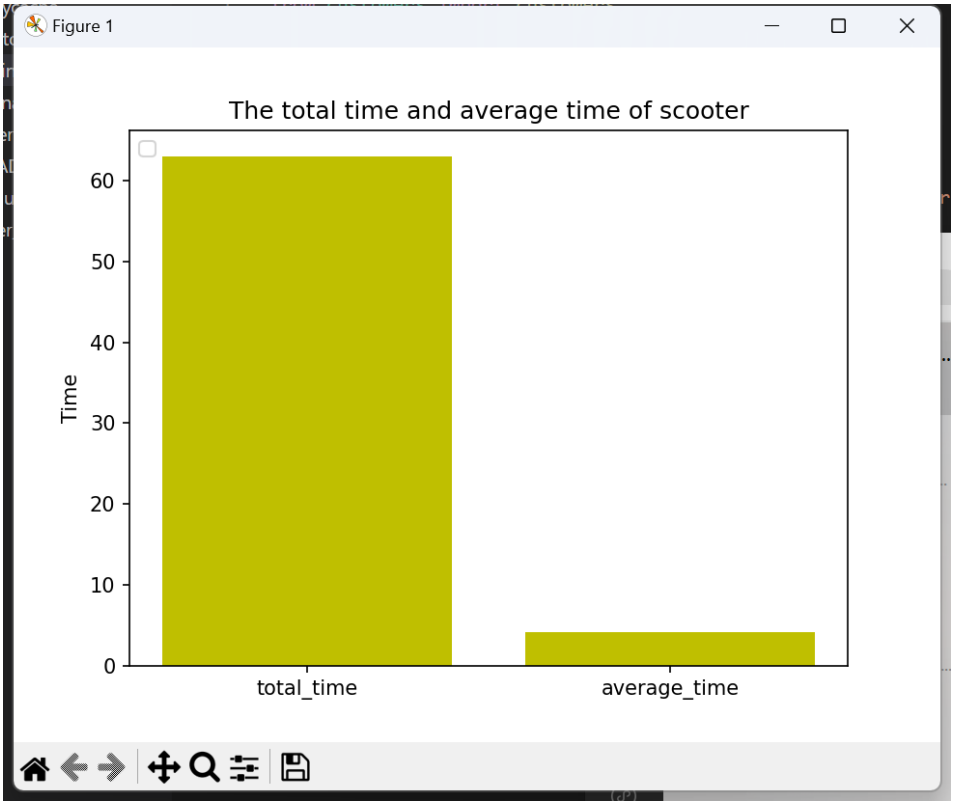


Figure 1
The total time and average time of scooter
Time
60
50
40
30
20
10
0
total_time
average_time

Figure25