

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT018-3-1-PYP**

**PROGRAMMING WITH PYTHON**

**APU2F2011IT(BIS) / APD2F2011CS(CYB) / APD2F2011IT(BIS) / UC2F2011IT(ISS) / APU2F2011IT / APD2F2011IT(CC) / APD2F2011MMT / APU2F2011IT(MBT) / APU2F2011IT(NC) / APD2F2011IT(MBT) / APU2F2011CS(CYB) / APD2F2011IT(NC) / APU2F2011IT(ISS) / APD2F2011IT(ISS) / APU2F2011IT(CC) / APU2F2011MMT / UC2F2011IT(BIS) / APU2F2011IT(IOT) / APD2F2011IT(IOT)**

**NAME: DIPTA PROTIM GUHA**

**STUDENT ID: TP063351**

**HAND OUT DATE: 23rd April 2021**

**HAND IN DATE: 27th June 2021**

**WEIGHTAGE: 100%**

**INSTRUCTIONS TO CANDIDATES:**

1. Submit your assignment online in Moodle.
2. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

Table of Contents

Cover Page……………………………………………………………………………………………………………………………………………1

Table of contents………………………………………………………………………………………………………………………………….2

Introduction………………………………………………………………………………………………………………………………………….3

Assumptions…………………………………………………………………………………………………………………………………………4

Design…………………………………………………………………………………………………………………………………………………. 8

1. Flowchart…………………………………………………………………………………………………………………………………9
2. Pseudocode……………………………………………………………………………………………………………………………31

Program Source code with explanation……………………………………………………………………………………………… 42

Samples of input/output with explanation………………………………………………………………………………………… 52

Conclusion………………………………………………………………………………………………………………………………………… 67

References……………………………………………………………………………………………………………………………………….. 68

**Introduction**

SUPER CAR RENTAL SERVICES (SCRS) is one of the fast-growing Online Car Rental

Service in Malaysia which help customer to save their time by booking rental cars online from their place. SCRS decided to enhance their online Car Rental booking services by allowing customers to book cars for rent, online. My task was to create a python-based program which users can use to rent cars online. The program has three key functionalities, Admin, Registered User and Non-Registered User, on which other functionalities depend on.

The following functionalities are broken down by the three key functionalities:

Functionalities of Admin

i. Login to Access System.

ii. Add Car with details, to be rented out.

iii. Modify Car Details

iv. Display All records of

a. Cars available for Rent

b. Customer Payment for a specific time duration

v. Search Specific record of

a. Specific Car Booking

b. Specific Customer Payment

vi. Return a Rented Car.

vii. Exit

Functionalities of All Customers (Registered / Not-Registered)

i. View all cars available for rent.

ii. New customer Register to Access other Details.

iii. Exit

Functionalities of Registered Customer

i. Login to Access System.

ii. View Personal Rental History.

iii. View Detail of Cars to be Rented Out.

iv. Select and Book a car for a specific duration.

v. Do payment to confirm Booking.

vi. Exit

**Assumptions**

**Main Menu**

The program has a main menu which provides the following options which users can key in to move forward with the program. The programs are:

1. Login
2. Signup
3. View All Cars
4. Exit

When users want to login they will be required to key in their username and password which the program will try to read from a text file, SCRSdatabase.txt. It will let users move forward if the login information provided matches with the text file. The text file can be updated when users select the register function. Here they can register with new username and password and select whether they wish to register as a user or an admin. The text file has a 3rd column which distinguishes between the types of users. If the username and password belong to an admin, they will be presented with the admin menu and if the username and password belongs to a user, they will be presented with the user menu.

If user press 3 and select the option to view all cars, data from another text file known as SCRScardatabase.txt, will be read by the program and presented to the user to see.

Finally, selecting option 4 will kill the program.

**Non-registered user**

Non-registered users can only view the main menu and will have to signup to update the SCRSdatabase.txt text file and then login to see the rest of the functionalities of the program.

**Admin view**

When a user login as admin, they will be presented with the following admin view:

-- Admin menu (enter an option) –

1) Register Car

2) View cars

3) Modify Car

4) View all rental history

5) Return To Main Menu

6) Exit

If the user keys in 1, it will let the user add a new car to the car database, SCRScardatabase.txt, text file.

If the user selects the option number 2, the program will do a similar thing the option 3 in the main menu. It will read from the text file, SCRScardatabase.txt, and present all the car information to the admin.

Option 3 lets user to modify car specifics in the SCRScardatabse.txt file. The text file is broken down in 6 columns. Plate no. of the car, the brand of the car, the model of the car, rent price per day of the car, the availability of the car which is represented by either A or B in the text file and lastly the next available date. User can search the specific car they want to modify by searching a car with the plate number. Once found, they can modify the car name, rent price and availability of the car. If the car is selected as available, the next available date will always show the date of the current day.

Admin can also return a car when the renting period is over by simply searching for the car and modifying the availability of the car from B (booked out) to A (available) and keeping the rest of the car details the same.

Selecting option 4, “View all rental history” presents the user with 3 options:

1) Search by a plate#

2) Search by customer username:

3) See all rental records

When either of these options are selected, information from another text file, SCRSrentaldatabase.txt, will be pulled by program and displayed to the user based on the options they select.

If the user selects option 1, “Search by a plate#”, admin can key in the plate number of a specific car and see all rental history of that car.

If the user selects option 2, “Search by customer username”, user can all the rental details of a specific customer.

If option 3,” See all rental records”, is selected, user will be presented with all rental details of all cars and customers without any filtration.

Lastly, selecting option 5 from the admin menu, will take user back to the main menu and selecting option 6 will kill the program.

**Registered-User view**

Registered-User view presents the user with the following view:

“

Hi, (Name of the user or username) (Registered-user)

-- Registered User menu (enter an option) –

1) View Cars

2) Rent Car

3) View Personal Rental History

4) Return To Main Menu

5) Exit

”

When user selects option 1, “View Cars”, the same process of the program pulling data from SCRScardatabase.txt text file and representing all the information of the cars to the user, will be executed.

If user selects option 2, “Rent Car”, the program will read from the SCRScardatabase.txt text file and filter it to show user with only the cars available for rent. Then the program will ask the customer to key in the plate number of the car they want to book. Once the plate number is keyed in, the program will ask user to select the number of days, the user wants to rent the car for. Once program takes the number of days input from the user, it will do a quick calculation to provide the customer with the total amount it will take to rent the car for the specific duration. Then the program will ask if the user wants to continue with the payment (Y for yes and N for no). If the user selects Y, the program will print a ticket with information:

“!!! Thank you for your purchase !!!

Booking person username:

Booking Date:

Car: KL001

Return Date:

Cost per day:

Total payable”

Simultaneously, the program will also modify, SCRScardatbase.txt text file to change the availability of the car from A (available) to B (booked-out) and also add a new record to the SCRSrentaldatabase.txt text file. The SCRSrentaldatabse.txt text file is the following format:

Car plate number, user who rented the car, number of days the car has been rented for, the total amount to rent the car depending on the amount of days the car had been rented for, the date it was rented and next available date of the car.

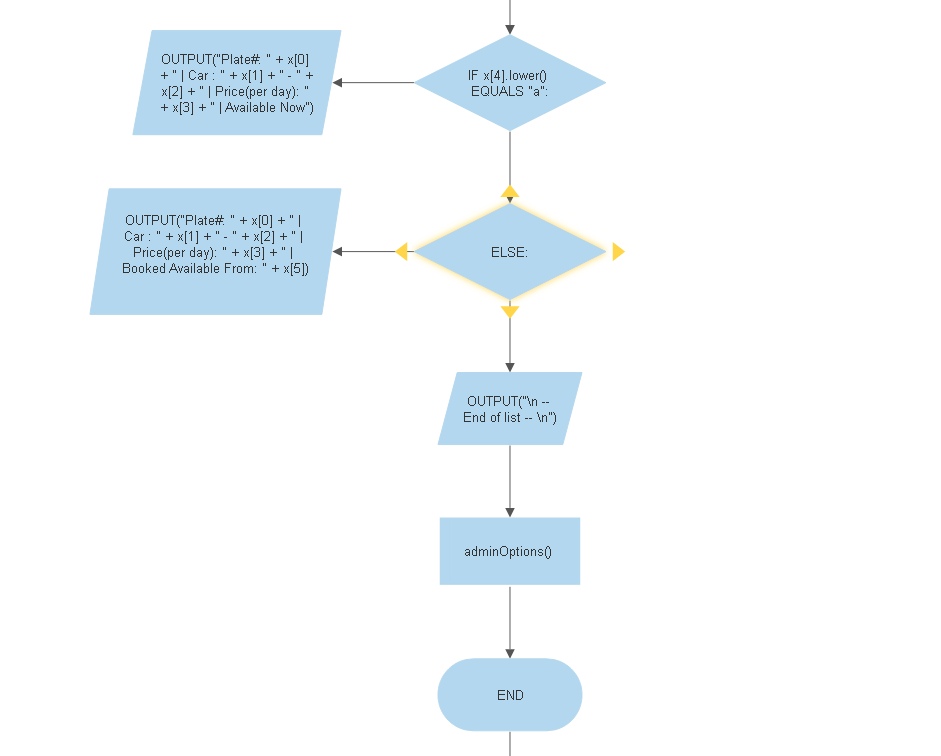
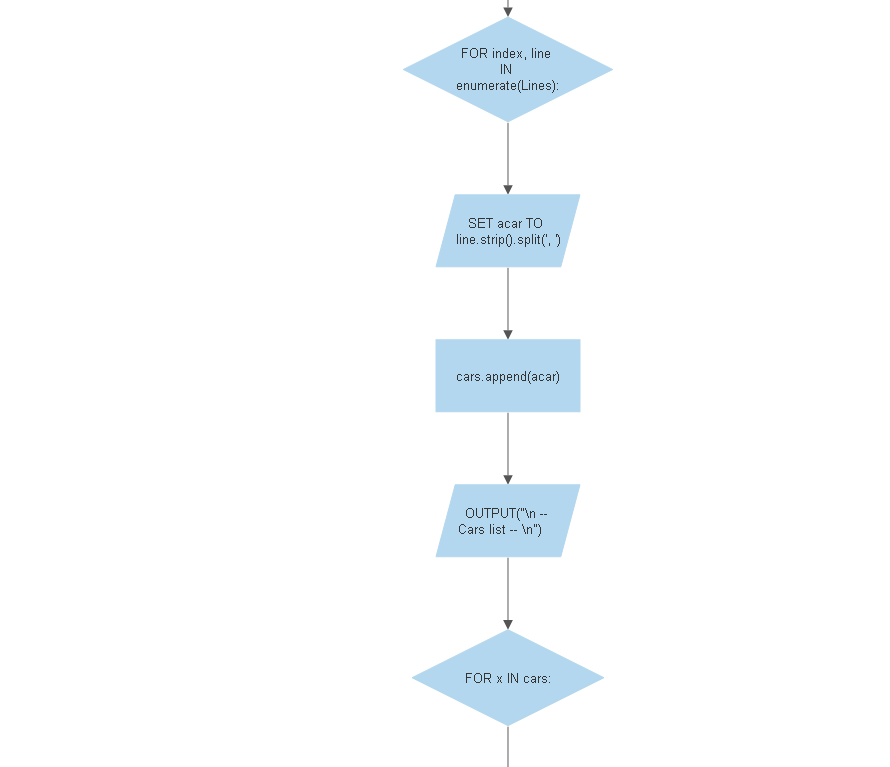
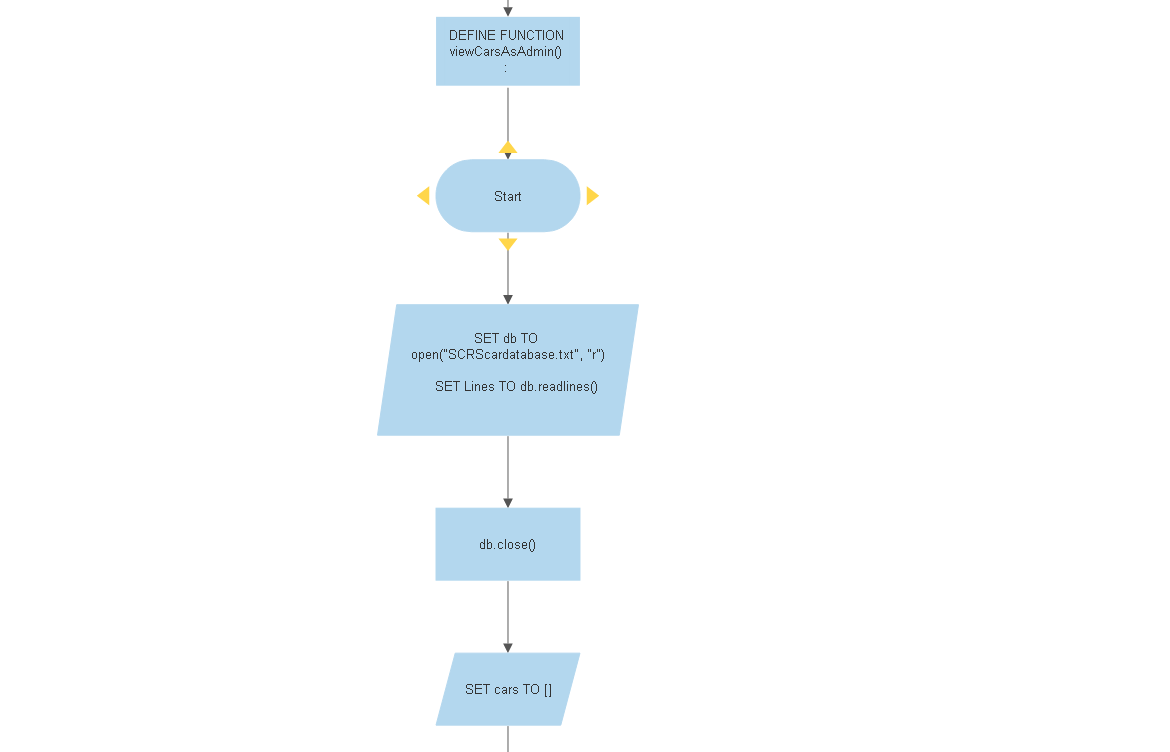
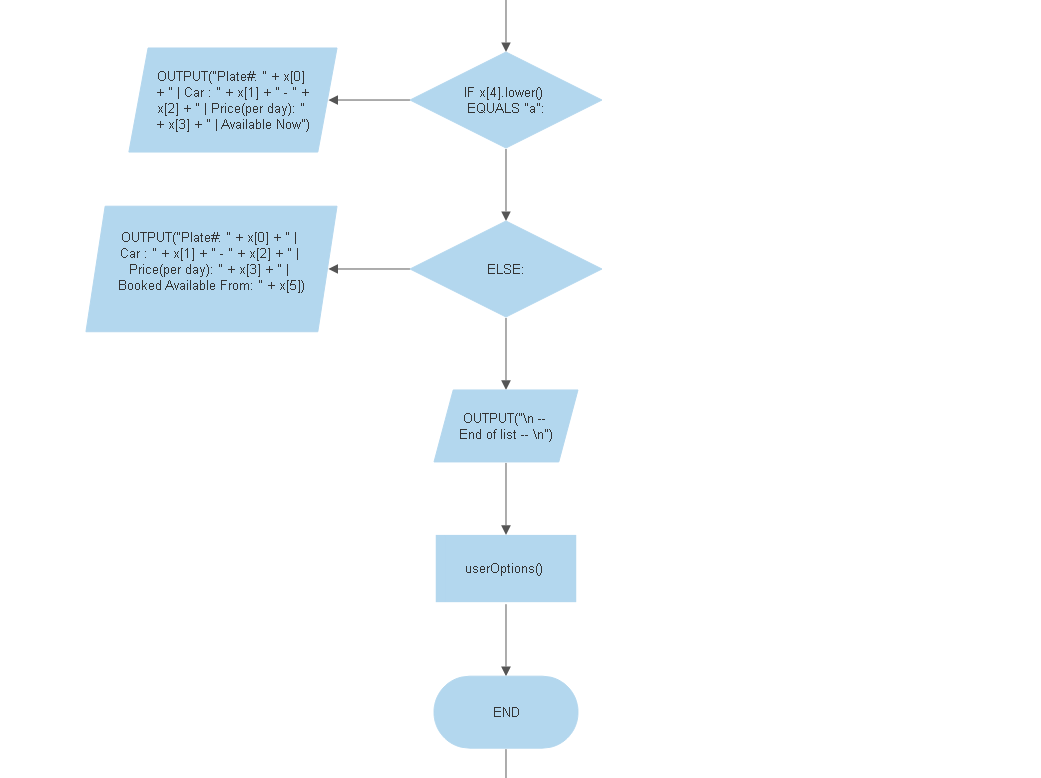
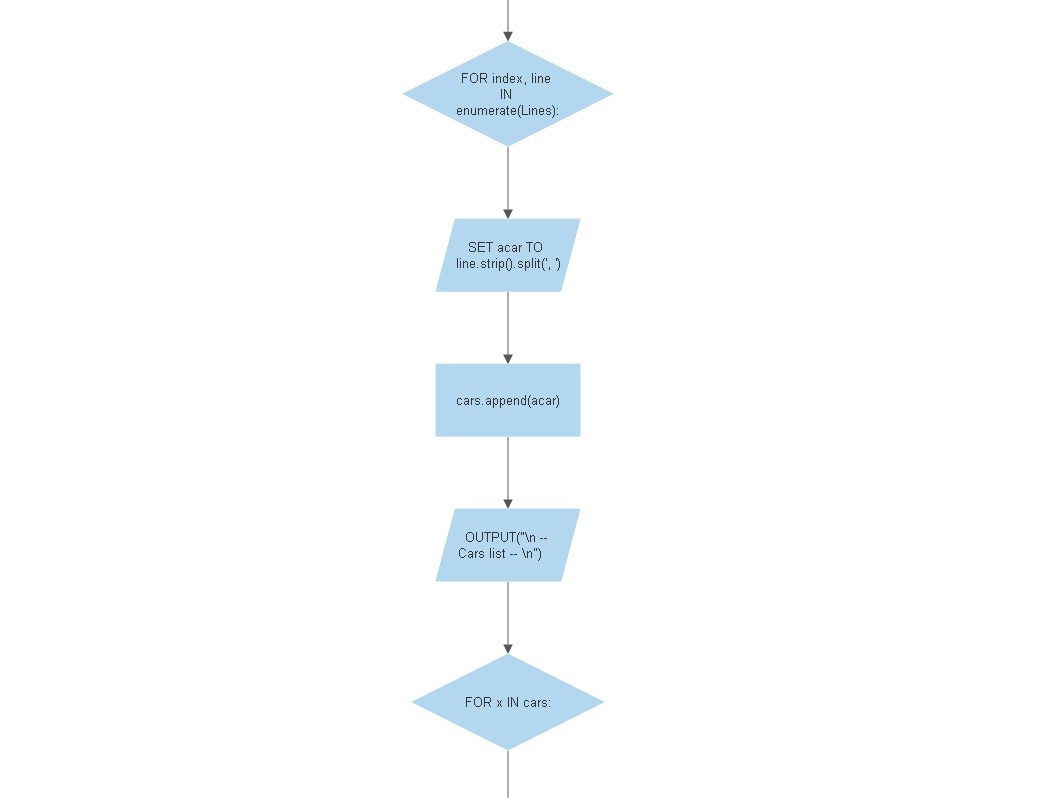
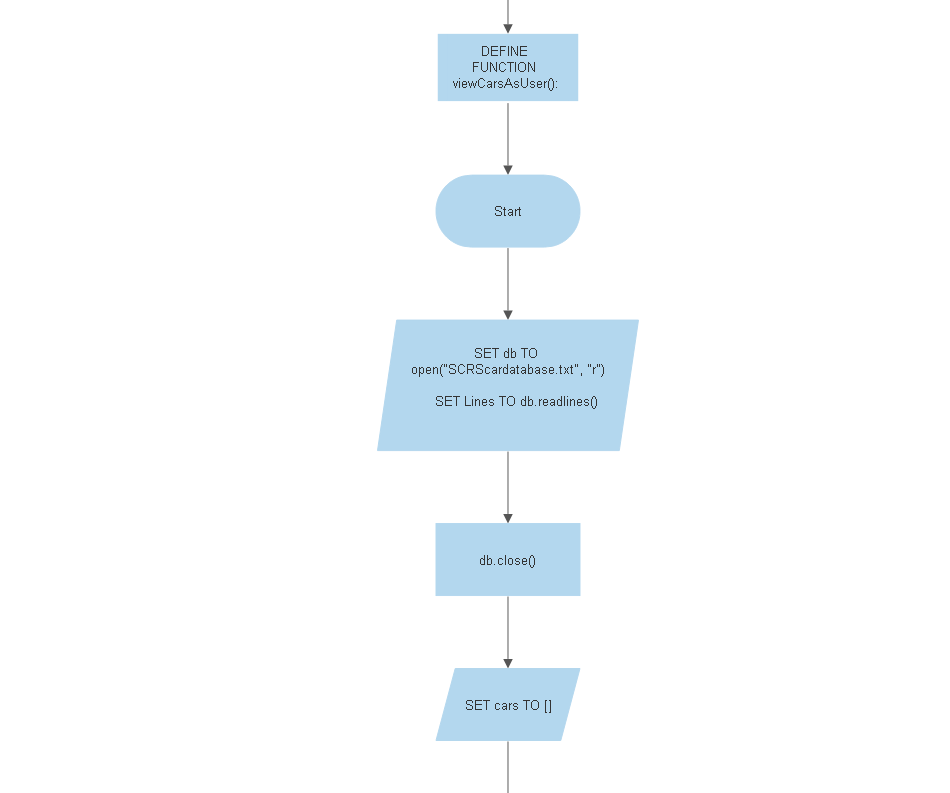
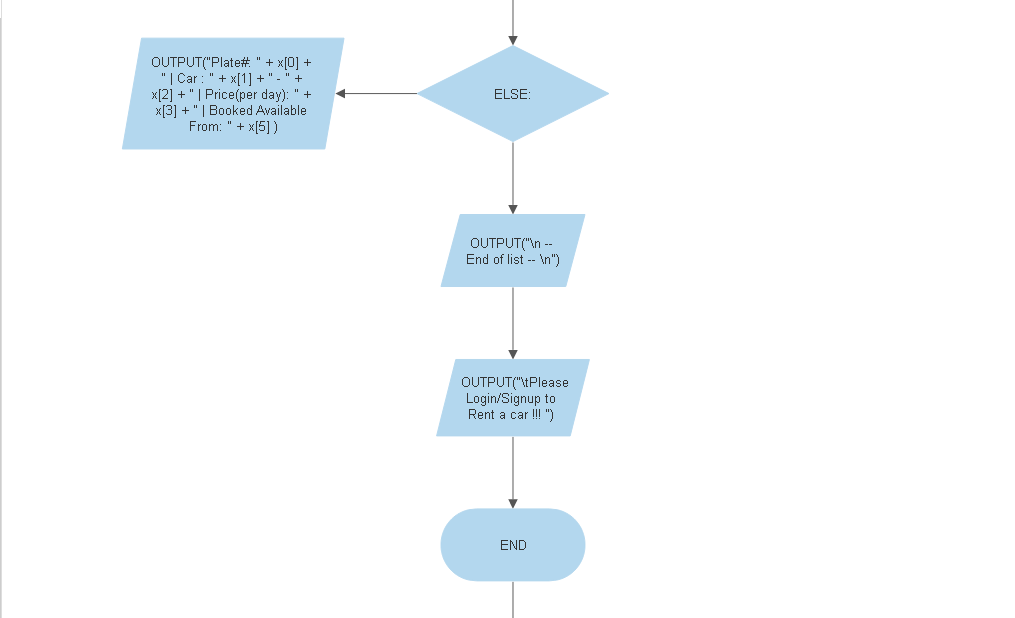
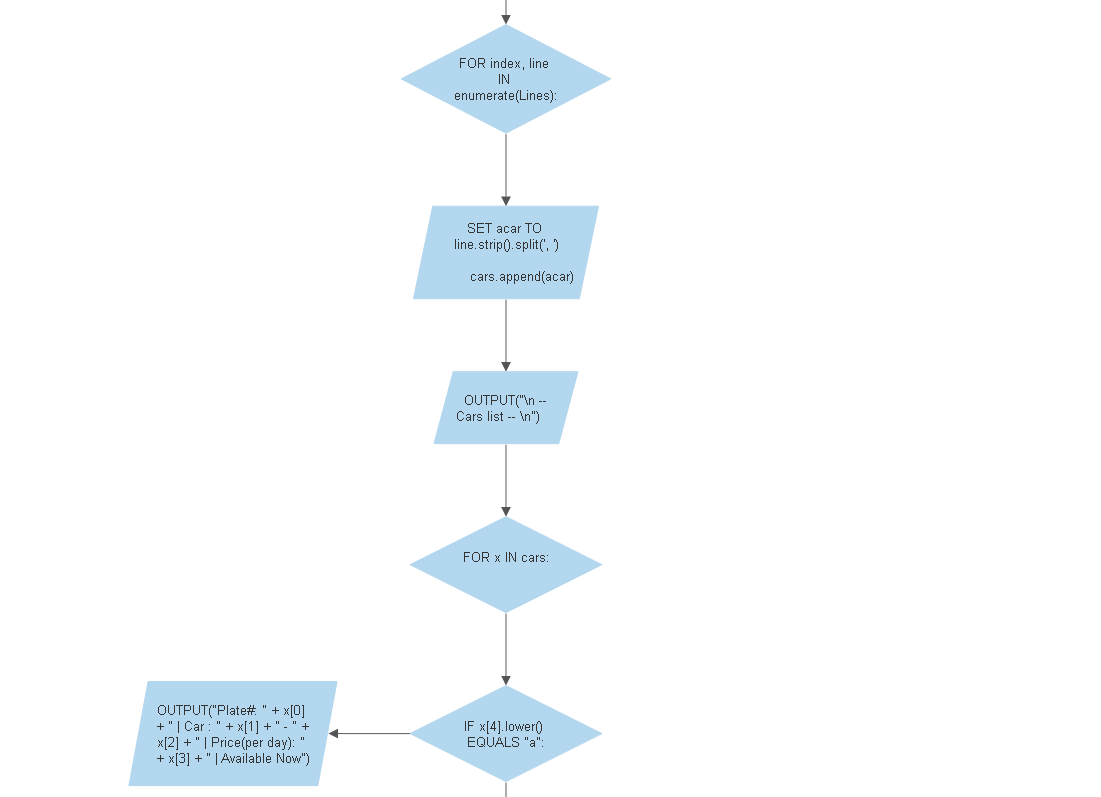
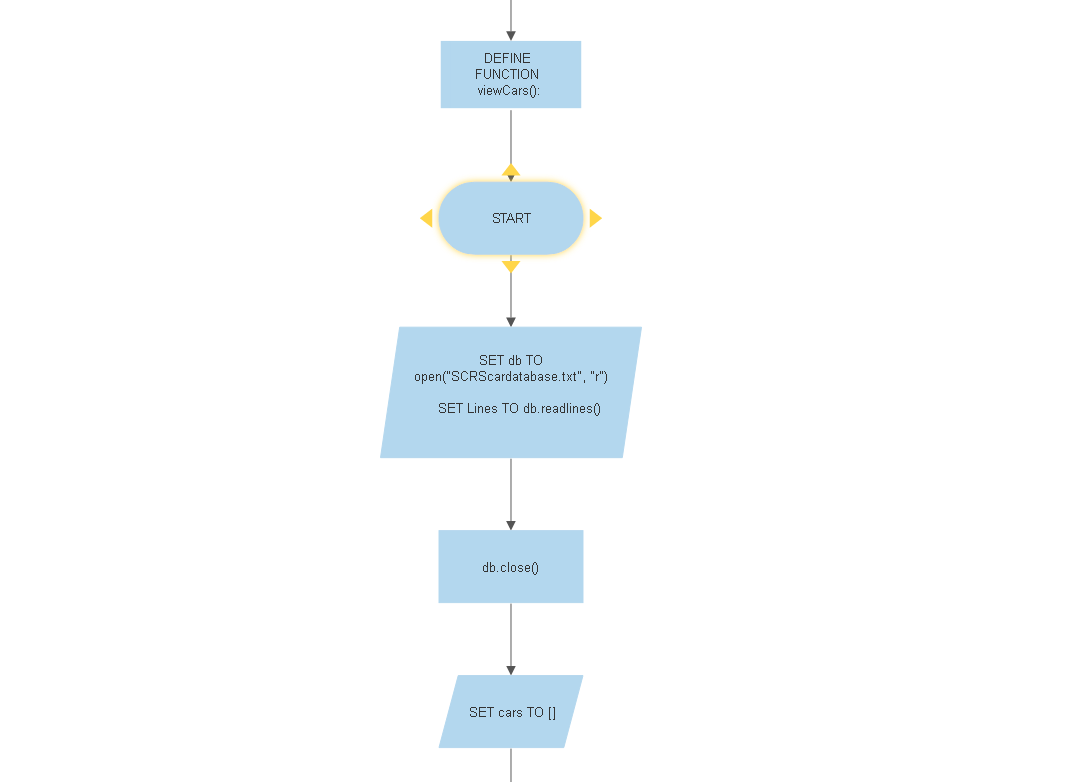
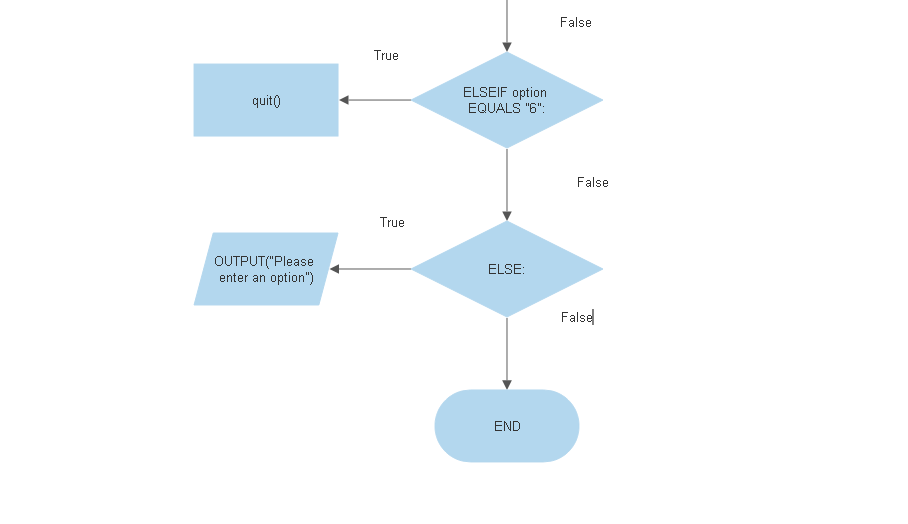
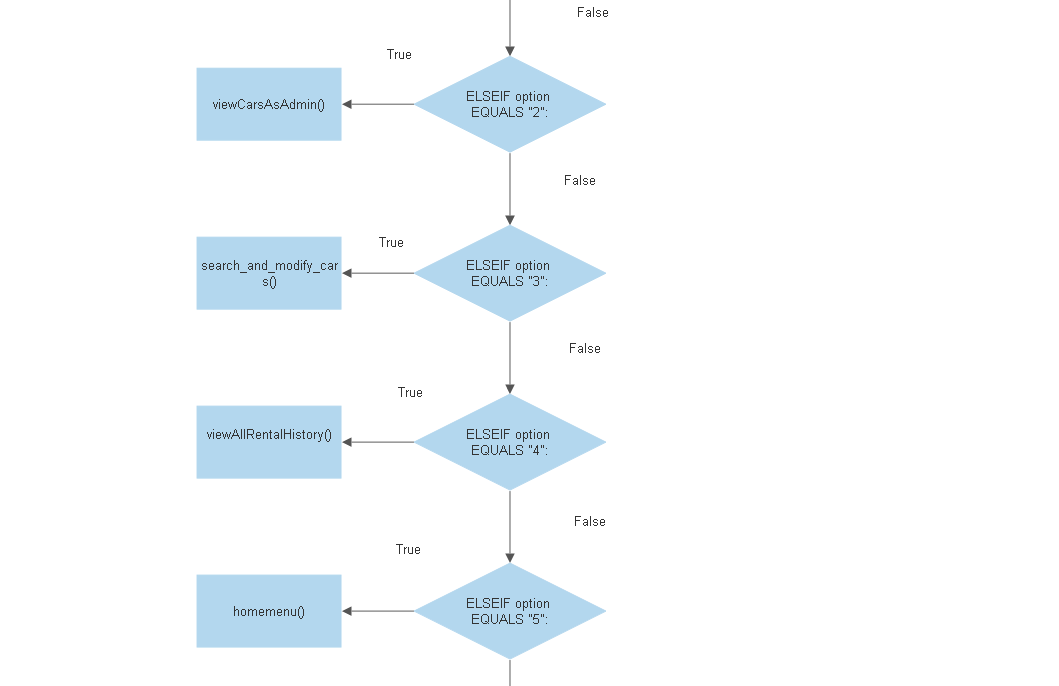
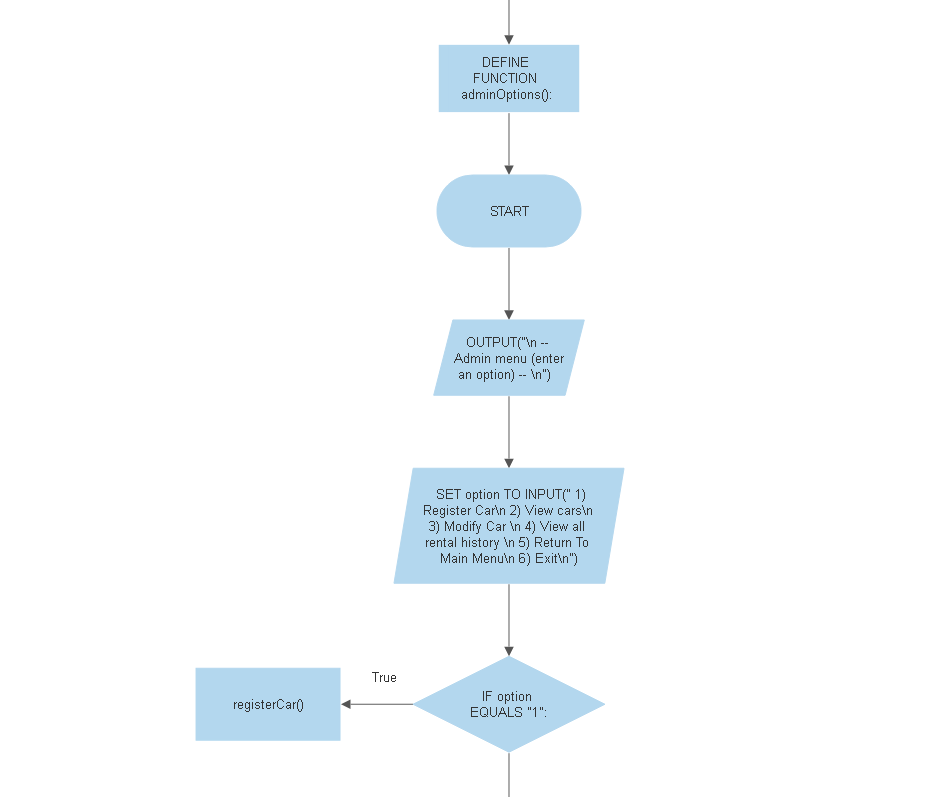
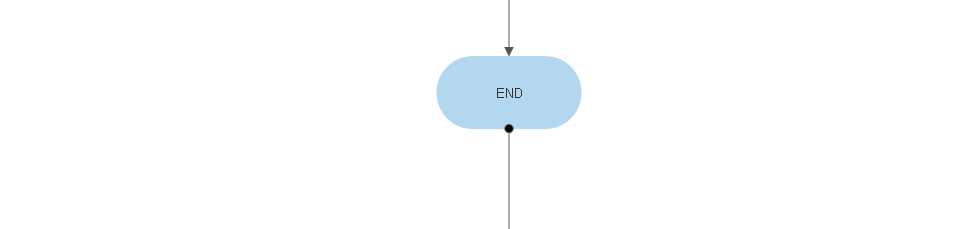
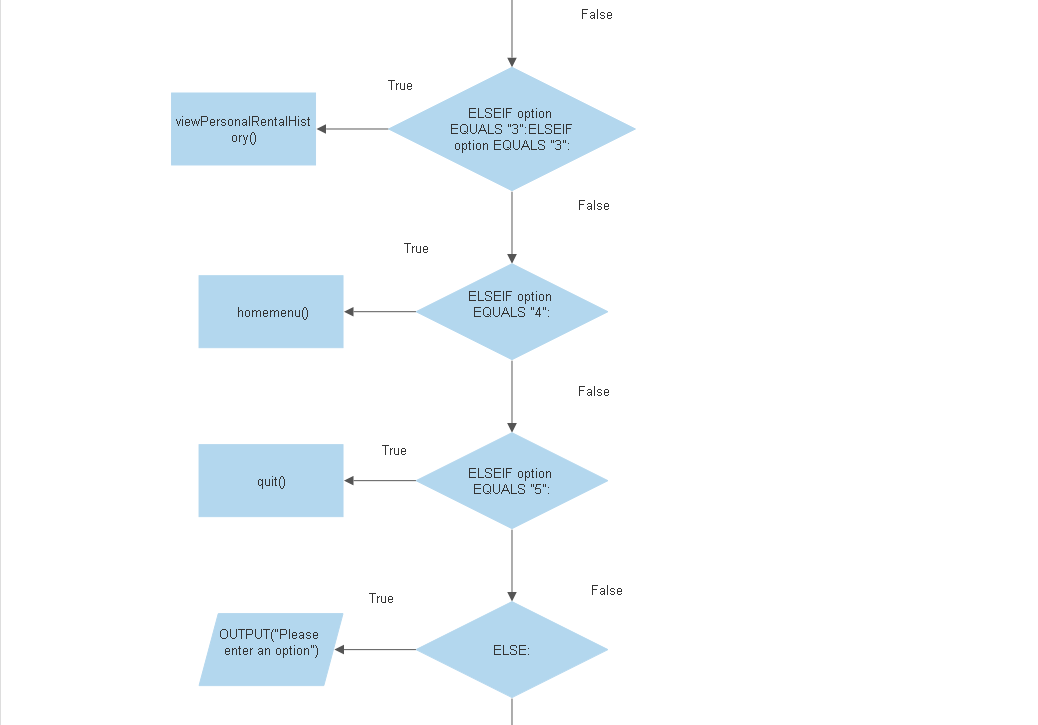
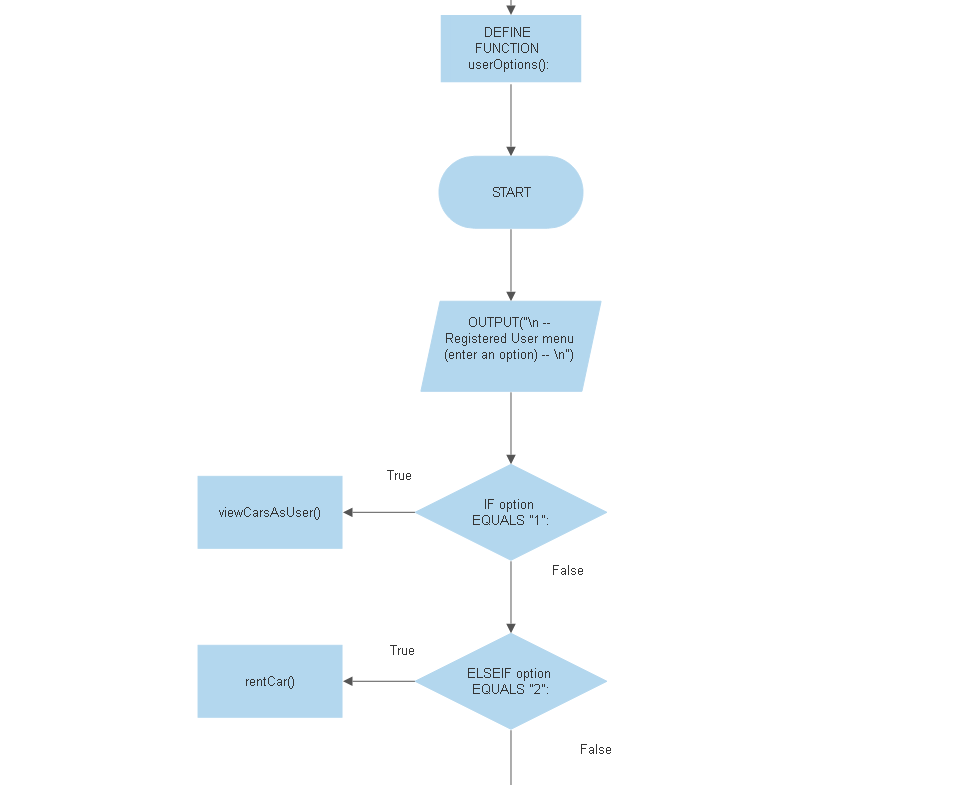
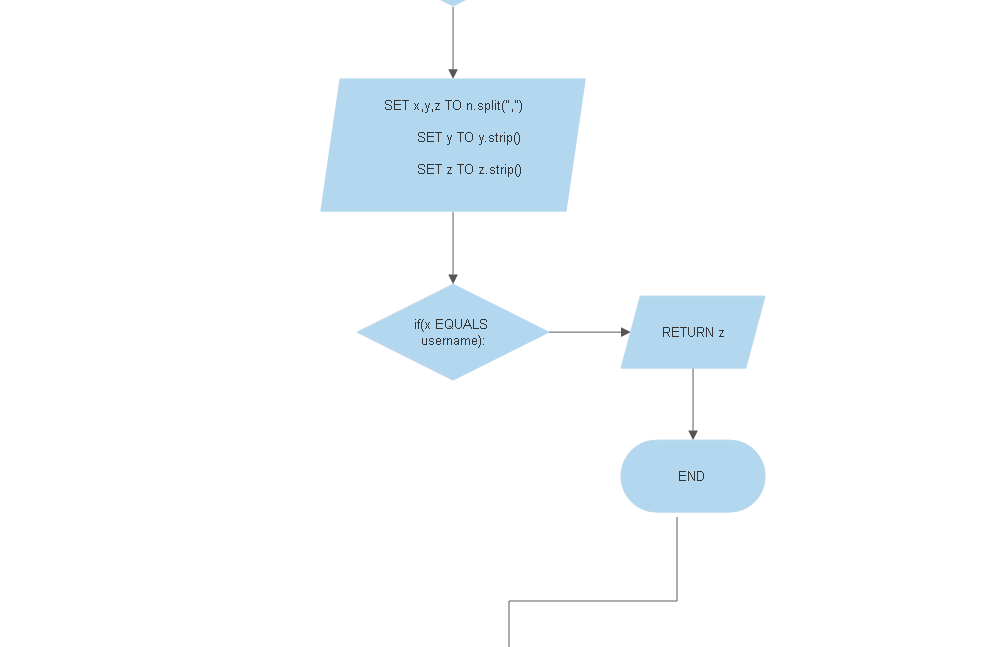
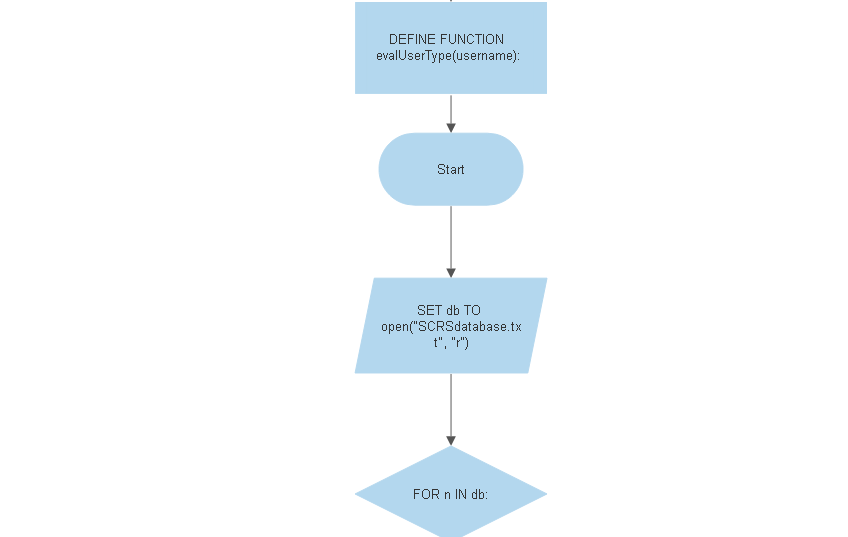
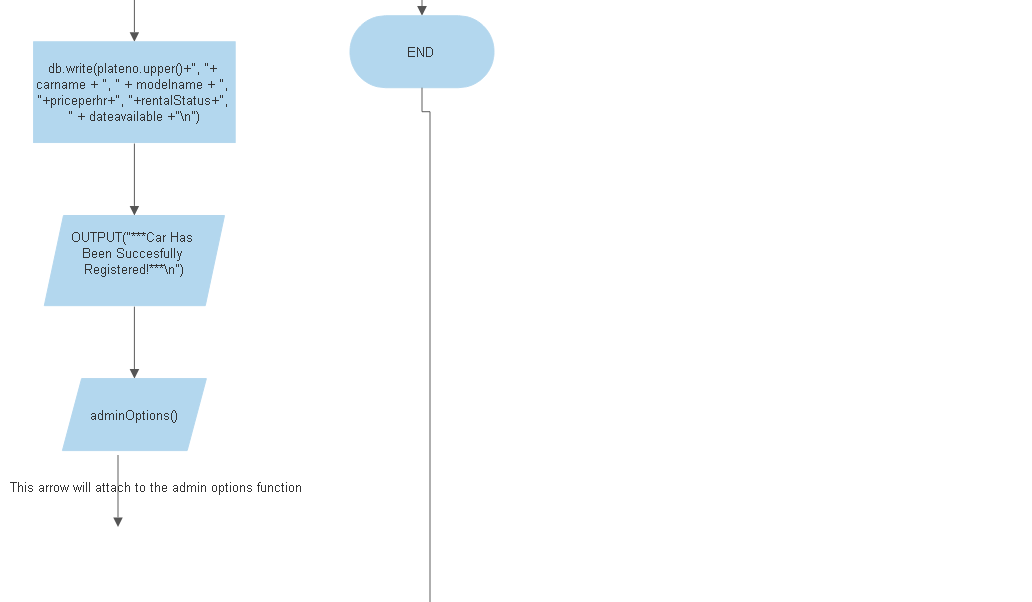
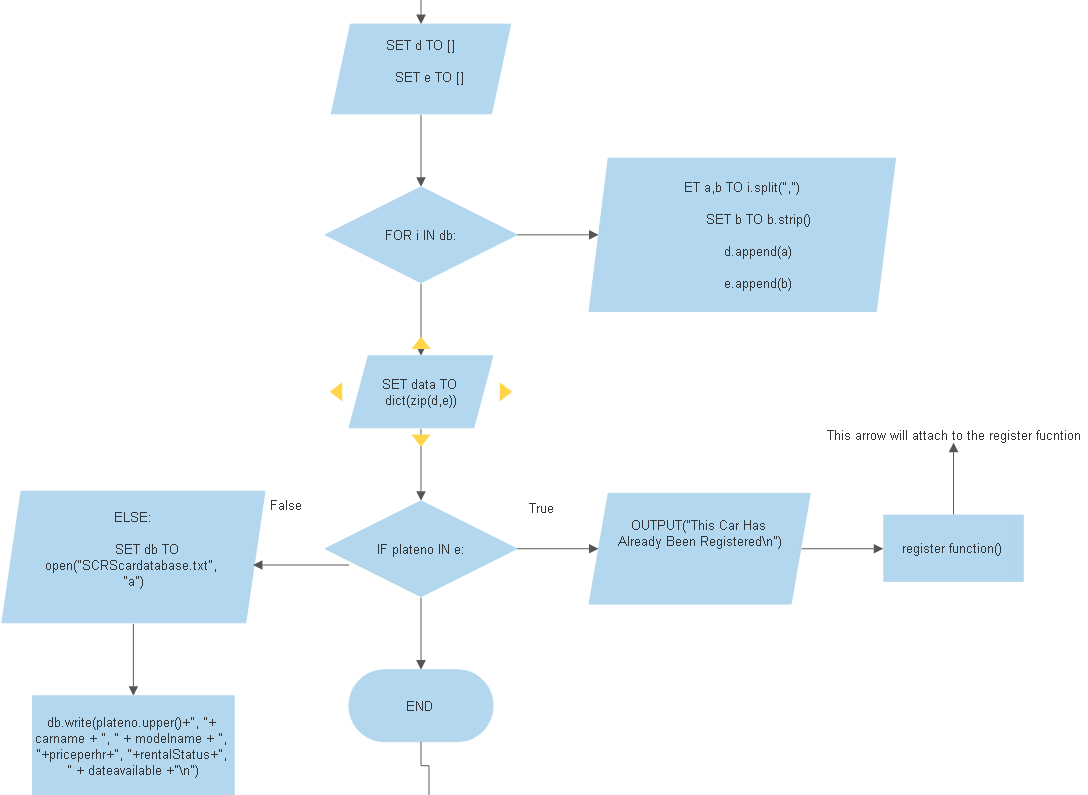
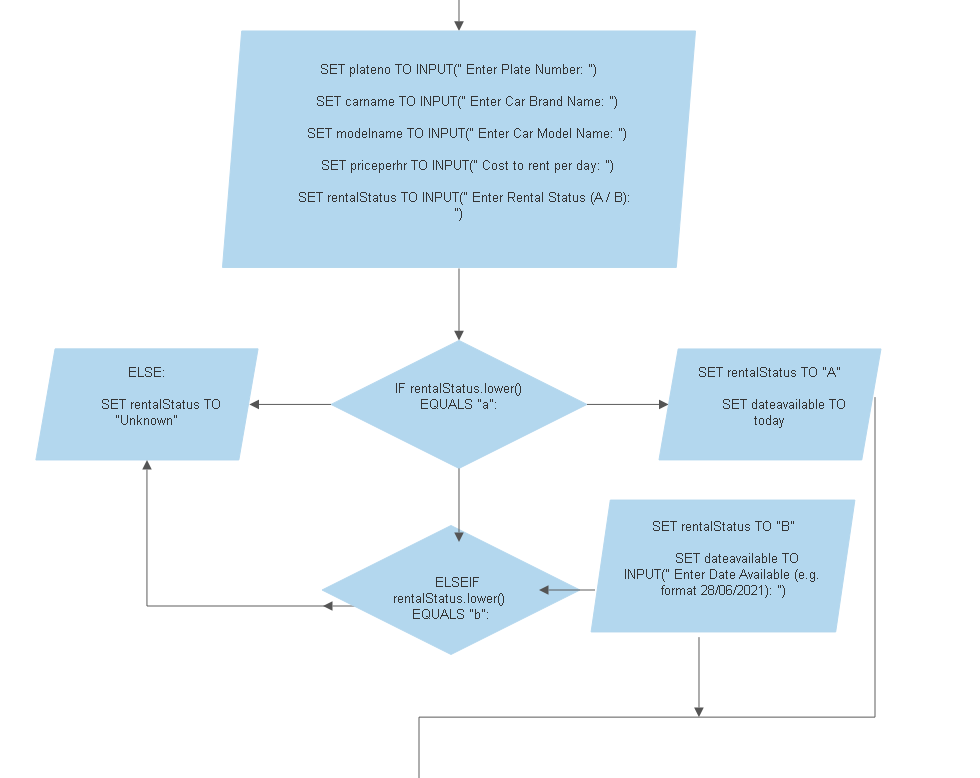
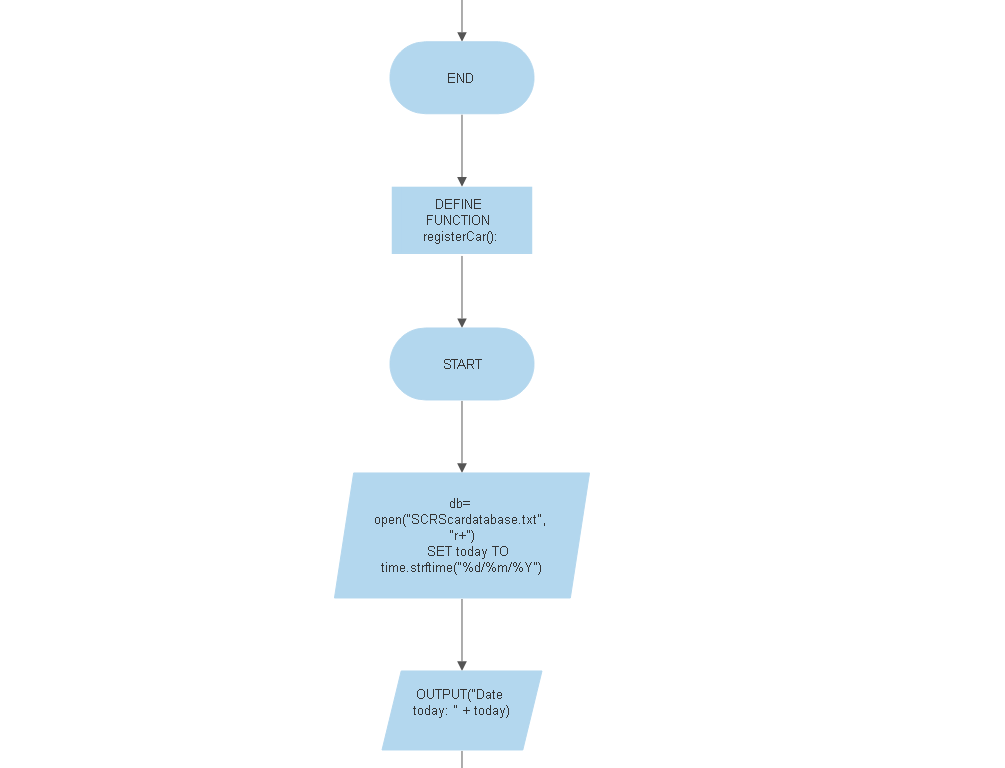
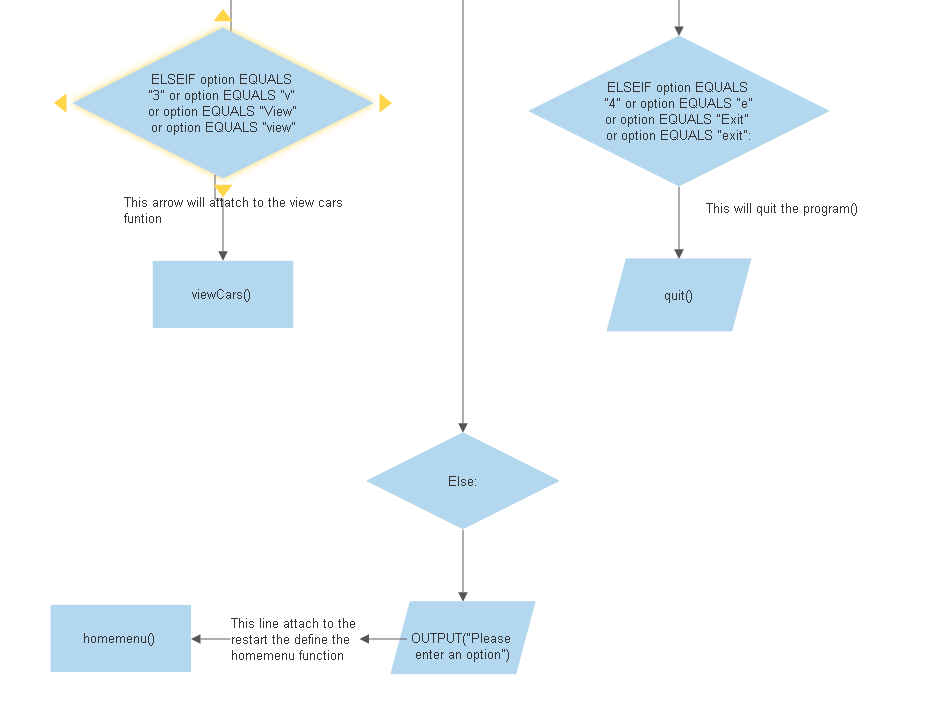
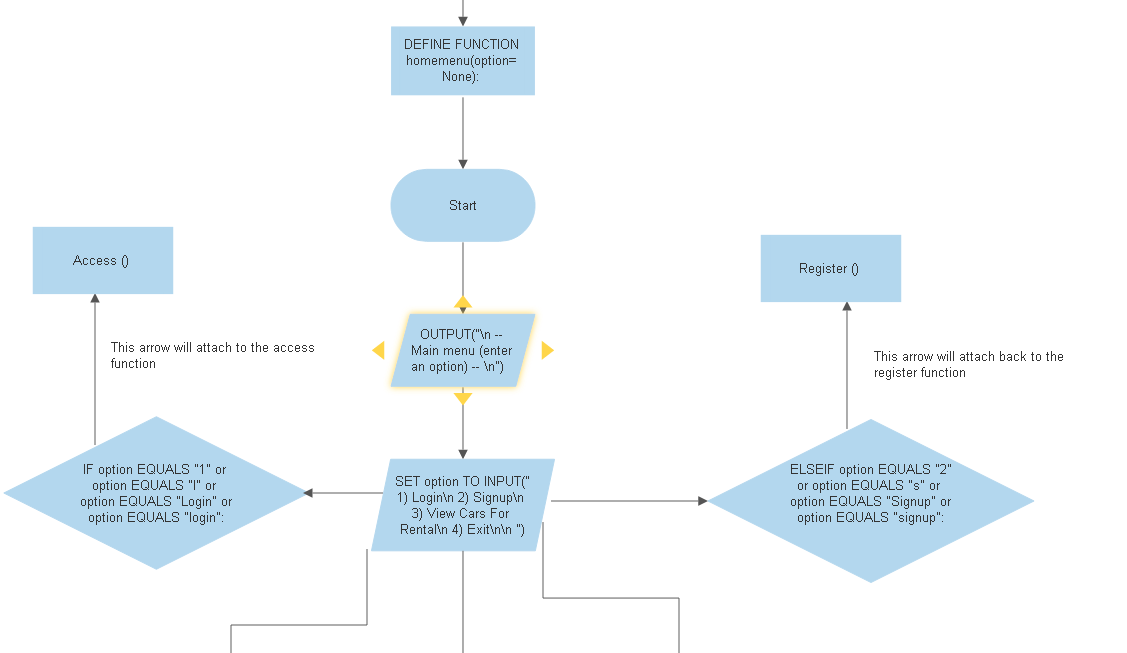
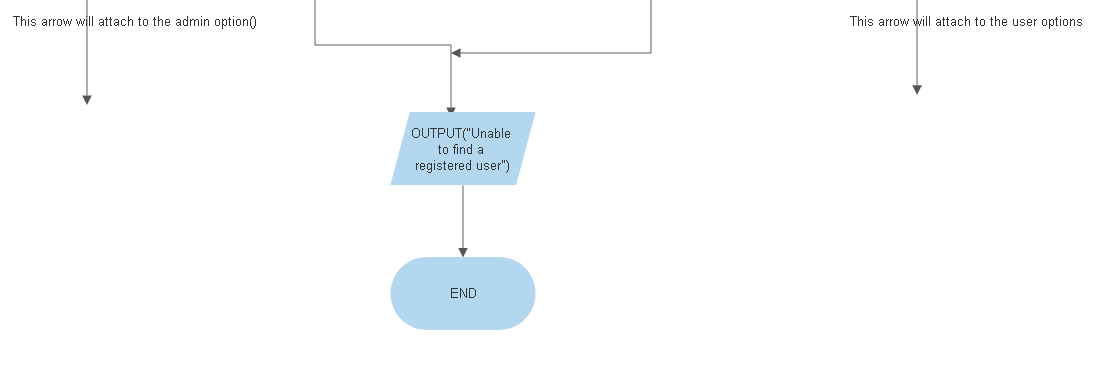
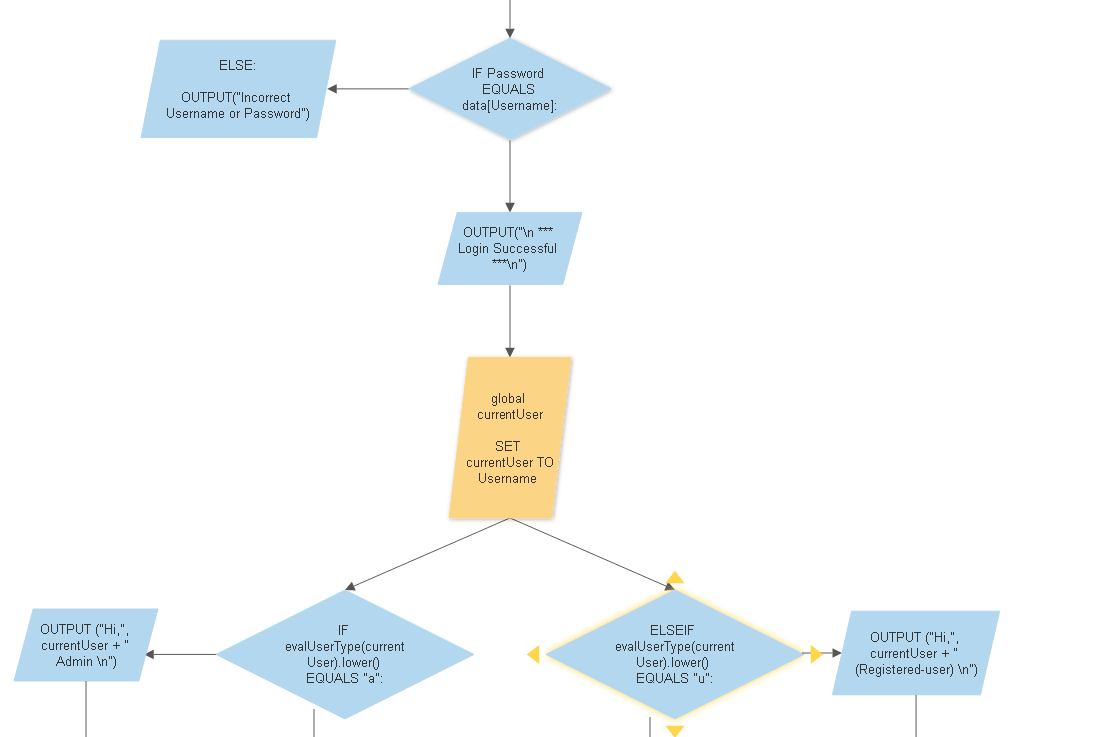
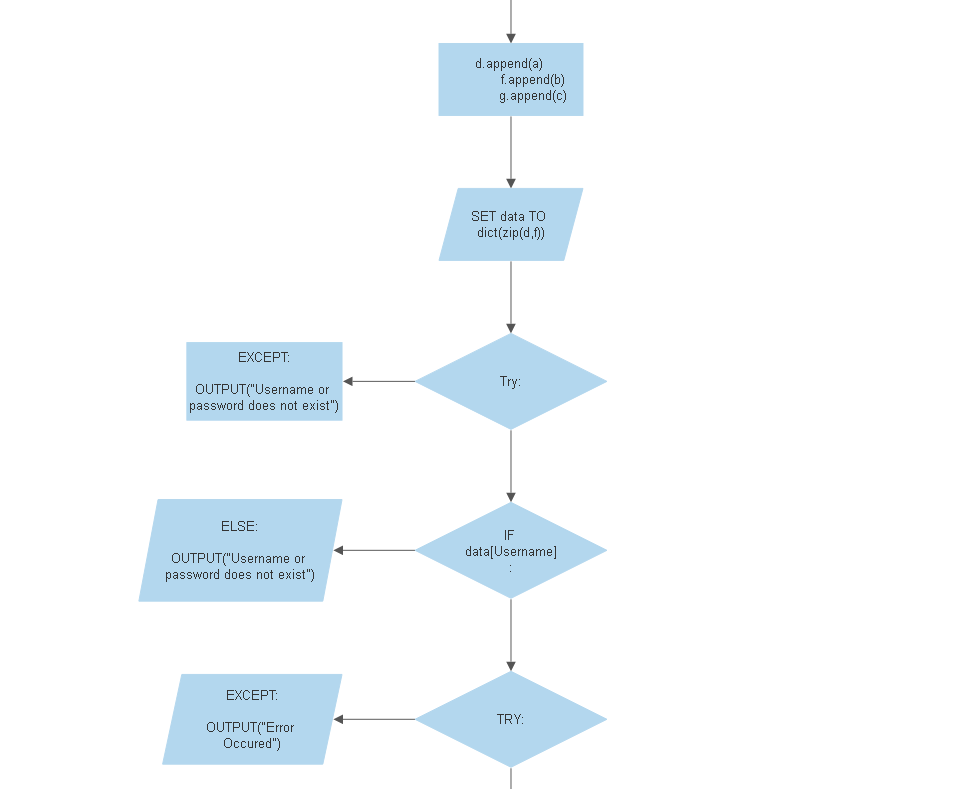
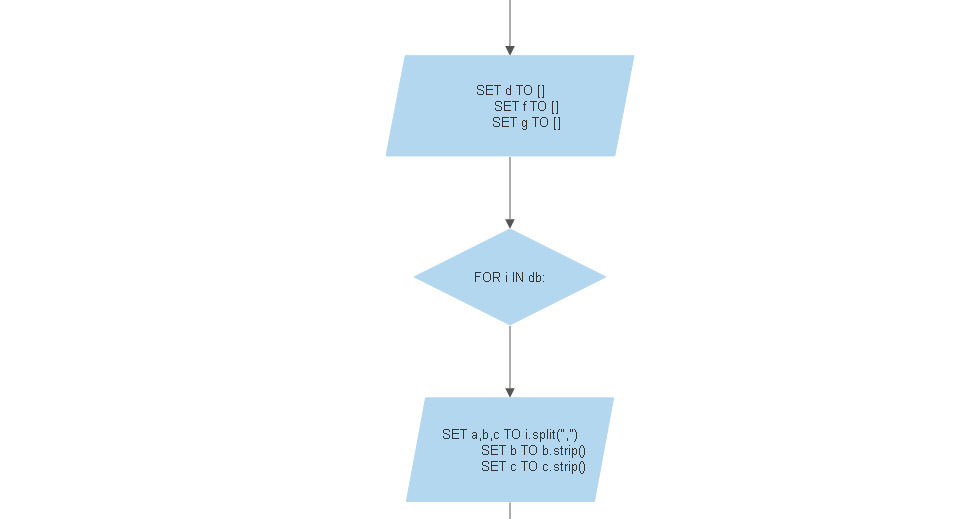
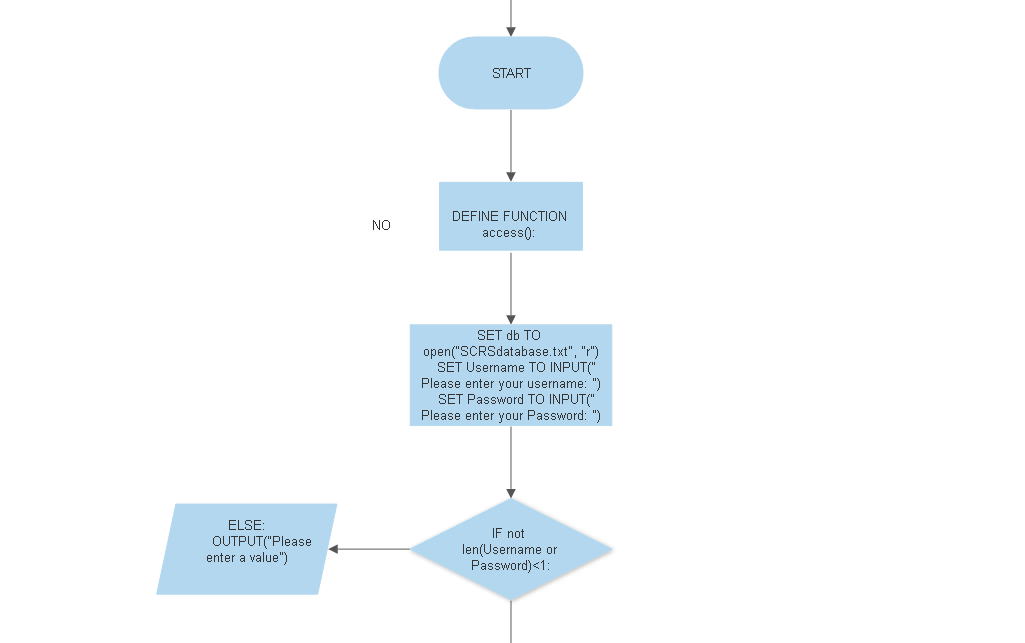
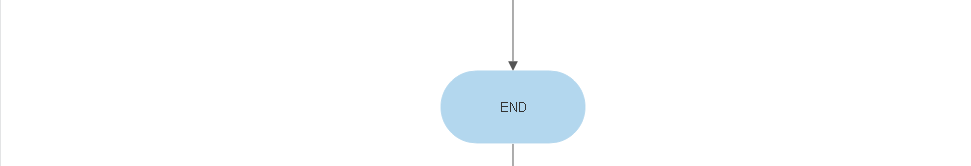
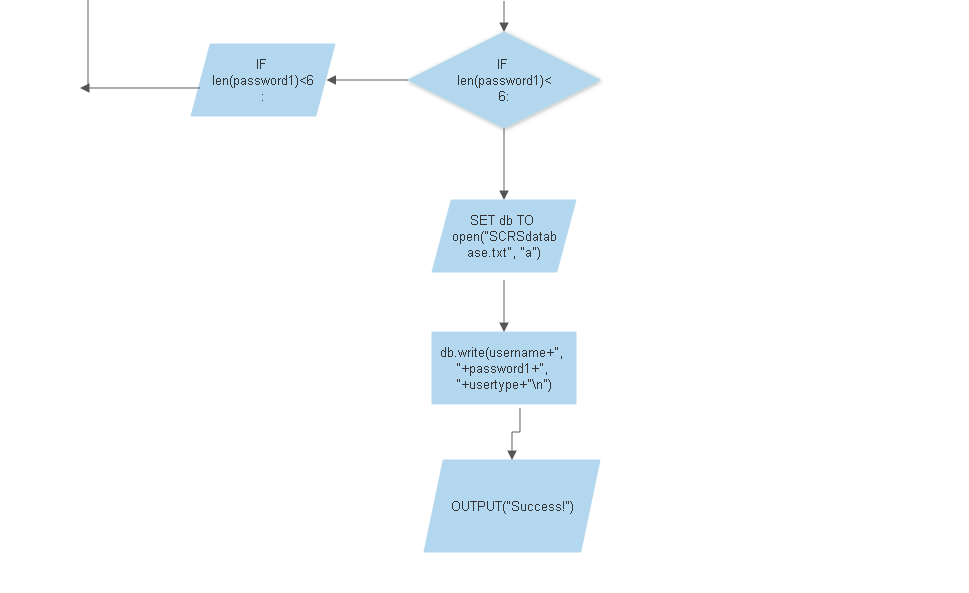
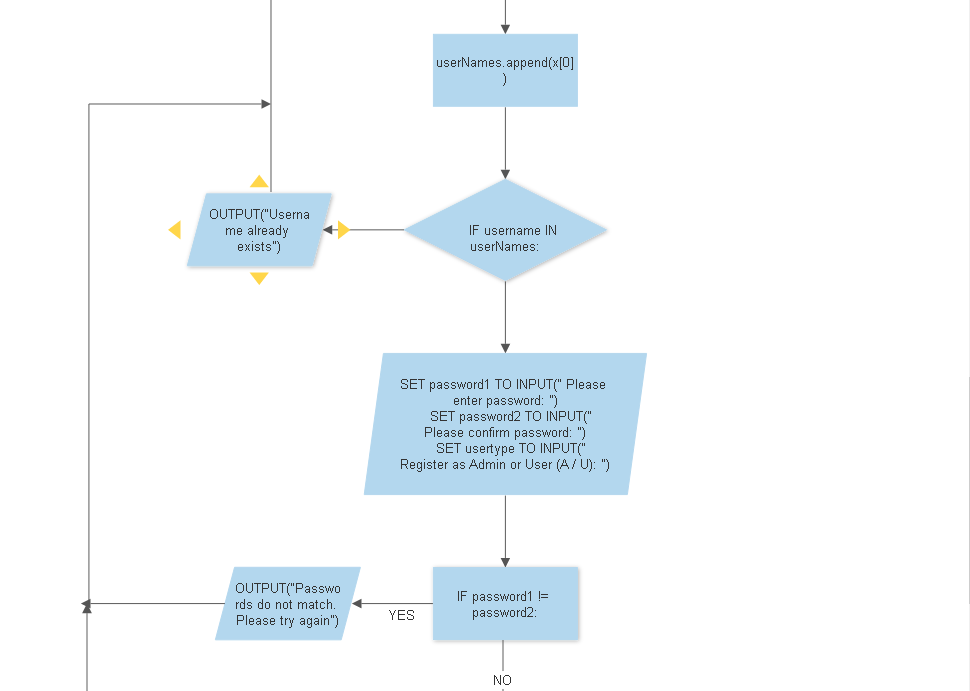
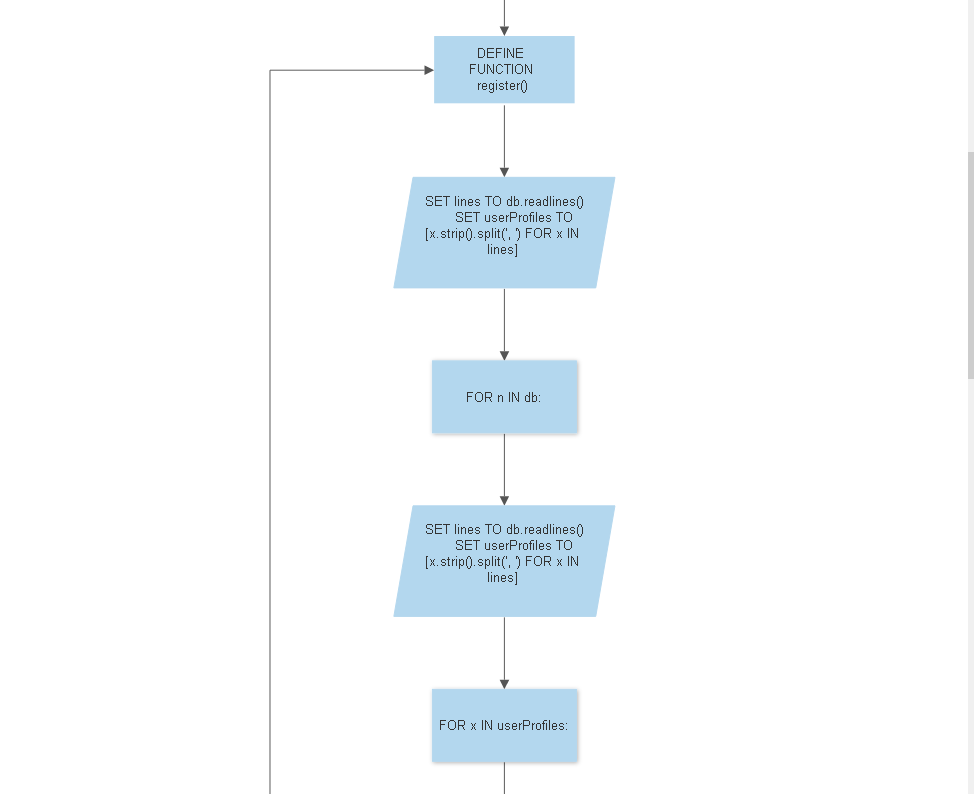
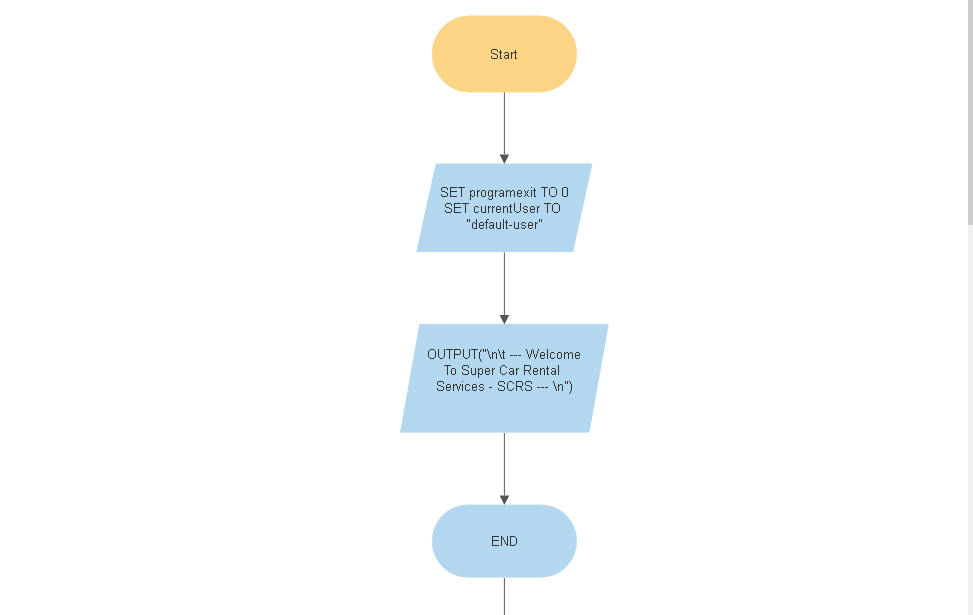
If the user selects option 3, “View Personal Rental History”, the program will read the SCRSrentaldatabase.txt text file and present the user with only the rental history of cars related with that specific user.

Finally, option 4, from the admin menu is selected, it will take user back to the main menu and selecting option 5 will kill the program.

**Design of the program**

The design of the program is broken down into 2 parts, flowchart and pseudocode. The first half of the program will be represented in flowcharts and the other half will be represented in pseudocode.

**Flowchart**

****

**Psuedocode**

**DEFINE FUNCTION search\_and\_modify\_cars():**

**SET fh\_r TO SCRScardatabase.txt**

**SET fh\_w TO temp\_modifycar.txt**

**OUTPUT "Enter the Plate number of the car:" AND SET plate\_no TO user INPUT**

**SET s TO ' '**

**while(s):**

**SET s TO line FROM fh\_r**

**SET L ARRAY TO FORMATTED s**

**IF LENGTH OF s > 0:**

**IF L[0] EQUALS TO plate\_no:**

**OUTPUT "\*\*\*Enter Update Values\*\*\*"**

**OUTPUT "Enter the brand of the car: " AND SET make TO user INPUT**

**OUTPUT "Enter the model of the car: " AND SET model to user INPUT**

**OUTPUT "Enter the price per day: " AND SET price to user INPUT**

**OUTPUT "Enter A IF available and B IF not: " AND rentalStatus price to user INPUT**

**SET today TO FORMATTED time**

**IF rentalStatus EQUALS "a":**

**SET rentalStatus TO "A"**

**SET dateavailable TO today**

**ELSEIF rentalStatus EQUALS "b":**

**SET rentalStatus TO "B"**

**OUTPUT "Enter Date Available (e.g. format DD/MM/YYYY): " AND SET dateavailable to user INPUT**

**IF "/" not IN dateavailable:**

**OUTPUT "Oops, Incorrect date format."**

**CALL search\_and\_modify\_cars METHOD**

**ELSE:**

**SET rentalStatus TO "Unknown"**

**INSERT plate\_no, make, model, price, rentalStatus, dateavailable TO SCRScardatabase.txt**

**ELSE:**

**INSERT s TO SCRScardatabase.txt**

**CALL close METHOD on fh\_r AND fh\_w**

**IF SCRScardatabase.txt EXISTS USING os.path.exists METHOD**

**CALL os.remove METHOD and remove SCRScardatabase.txt**

**CALL os.rename METHOD and rename temp\_modifycar.ttx.txt TO SCRScardatabase**

**ELSE:**

**OUTPUT "The file does not exist"**

**CALL adminOptions METHOD**

**DEFINE FUNCTION rentCar():**

**SET db TO SCRScardatabase.txt**

**SET db2 TO SCRSrentaldatabase.txt**

**SET dbcars TO line FROM db**

**SET carRecords TO []**

**FOR index, line IN dbcars USING enumerate METHOD**

**SET acar TO FORMATTED line**

**ADD acar TO carRecords**

**SET availableCars TO []**

**FOR x IN carRecords**

**if(x[4] EQUALS "A")**

**ADD x TO availableCars**

**IF availableCars LENGHT EQUALS TO 0**

**OUTPUT "\t --- Oops, No Car is available to rent. tray again later. --- \n"**

**IF availableCars LENGHT > 0**

**SET availableCarscount TO availableCars LENGHT**

**OUTPUT "\n\t ---- Cars Available to rent " + str(availableCarscount) + " ----\n"**

**FOR index, car IN availableCars USING enumerate METHOD**

**SET item TO 1 + index**

**OUTPUT (str(item) + ") " + car[1] + " " + car[2] + " | Plate# " + car[0] + " | " + "Cost to Rent per day: " + str(car[3]) + "\n")**

**OUTPUT "\t --- End of list --- \n"**

**SET cost TO 0**

**OUTPUT "\t !!! Rental form !!! \n"**

**OUTPUT "Booking person username: " + currentUser + "\n"**

**SET plate\_no TO INPUT("Enter the Plate number of the car: "**

**SET plate\_no TO FORMATTED plate\_no**

**SET carfound TO True**

**if availableCars LENGTH > 0**

**FOR x IN availableCars:**

**if(x[0] EQUALS TO plate\_no and x[4] EQUALS TO "A"):**

**SET carfound TO True**

**OUTPUT "Enter rent duration (in days): " AND SET rent\_duration to user INPUT**

**SET car TO x**

**SET car\_costperday TO x[3]**

**SET total\_cost TO car\_costperday \* rent\_duration**

**OUTPUT "To book " + car[0] + " " + car[1] + " - " + car[2] + ", it's going to cost RM " + str(total\_cost) + ". \n"**

**OUTPUT "Enter Y to book and pay or N to cancel booking: " AND SET confirm to user INPUT**

**IF (confirm DOES NOT EQUAL "y")**

**CALL userOptions METHOD**

**SET bookingdate TO FORMATTED time**

**COMPUTE RETURNdate USING timedelta METHOD**

**SET RETURNdate TO FORMATTED RETURNdate**

**INSERT plate\_no, currentUser, rent\_duration, total\_cost, bookingdate, RETURNdate TO db2**

**OUTPUT "\n\t !!! Thank you FOR your purchase !!!" + "\n"**

**OUTPUT "Booking person username: " + currentUser + "\n"**

**OUTPUT "Booking Date: " + bookingdate + "\n"**

**OUTPUT "Car: "+ plate\_no + " " + car[1] + " " + car[2]+ "\n"**

**OUTPUT "Return Date: " + RETURNdate + "\n"**

**OUTPUT "Cost per day: RM " + str(car\_costperday) + "\n"**

**OUTPUT "Total payable: RM " + str(total\_cost)+ "\n"**

**CALL changeCarRentalStatus METHOD**

**BREAK**

**ELSE:**

**SET carfound TO False**

**IF (carfound EQUALS TO FALSE):**

**OUTPUT "Sorry, no car is available with Plate# " + plate\_no**

**CALL close METHOD on db AND db2**

**call userOptions() METHOD**

**DEFINE FUNCTION changeCarRentalStatus(plate\_no,newstatus,RETURNdate):**

**SET fh\_r TO "SCRScardatabase.txt"**

**SET fh\_w TO "tempcardb.txt"**

**SET s TO ' '**

**while(s):**

**SET s TO LINE FROM fh\_r**

**SET acar TO FORMATTED s**

**IF s LENGTH > 0:**

**IF acar[0] EQUALS TO plate\_no:**

**INSERT acar[0]+", "+acar[1]+", "+acar[2]+", "+acar[3]+", "+newstatus+", "+RETURNdate+"\n" TO fh\_w**

**ELSE:**

**INSERTs TO fh\_w**

**CALL Close METHOD on fh\_r and fh\_w**

**IF CRScardatabase.txt EXISTS USING os.path.exists METHOD:**

**TRY:**

**CALL os.unlink METHOD on SCRScardatabase.txt**

**CALL os.rename METHOD tempcardb.txt**

**except OSError as e:**

**OUTPUT "Failed with:", e.strerror**

**ELSE:**

**OUTPUT "The file does not exist"**

**DEFINE FUNCTION viewPersonalRentalHistory():**

**SET rentaldb TO "SCRSrentaldatabase.txt"**

**OUTPUT "\t Rental History of user " + currentUser + "\n"**

**SET dbrecords TO LINE FROM rentaldb**

**CALL close METHOD on rentaldb**

**SET personalRecords TO []**

**FOR index, line IN dbrecords USING enumerate METHOD:**

**SET record TO FORMATTED line**

**IF record[1] EQUALS currentUser:**

**ADD record TO personalRecords**

**if LENGTH personalRecords > 0**

**FOR y IN personalRecords:**

**OUTPUT "Plate#: " + y[0] + " | Total Paid : " + y[3] + " | Booking: " + y[4] + " - " + y[5] + "\n"**

**CALL userOptions() METHOD**

**DEFINE FUNCTION viewAllRentalHistory():**

**SET rentaldb TO SCRScardatabase.txt**

**OUTPUT "1) Search by a plate# \n 2) Search by customer username:\n 3) See all rental records\n" AND SET option to user INPUT**

**SET plate\_no TO ''**

**IF option EQUALS TO 1:**

**OUTPUT "Enter the Plate#: " AND SET plate\_no TO user INPUT**

**SET plate\_no TO FORMATTED plate\_no**

**SET uname TO ''**

**IF option EQUALS 2:**

**SET uname TO INPUT ("Enter the customer username: ")**

**SET dbrecords TO rentaldb.readlines()**

**CALL the CLOSE METHOD on rentaldb**

**SET allRecords TO []**

**SET filteredRecords TO []**

**SET filteredRecordsbyuser TO []**

**FOR index, line IN dbrecords USING enumerate METHOD:**

**SET record TO FORMATTED line**

**ADD record TO allRecords**

**IF int(option) EQUALS 1 and record[0] EQUALS plate\_no:**

**ADD record TO filteredRecords**

**IF int(option) EQUALS 2 and record[1] EQUALS name:**

**ADD record TO filteredRecordsbyuser**

**IF option EQUALS TO 1:**

**if LENGTH filteredRecords > 0:**

**OUTPUT("\n\t Rental History records for: " + plate\_no + "\n")**

**FOR y IN filteredRecords:**

**OUTPUT "Plate#: " + y[0] + " | Booked By: " + y[1] + " | Total Paid : " + y[3] + " | Booking: " + y[4] + " - " + y[5] + "\n"**

**ELSE:**

**OUTPUT "--- No records --- "**

**IF option EQUALS 2:**

**if LENGTH filteredRecordsbyuser > 0:**

**OUTPUT "\n\t Rental History records for: " + uname + "\n"**

**FOR y IN filteredRecordsbyuser:**

**OUTPUT "Plate#: " + y[0] + " | Booked By: " + y[1] + " | Total Paid : " + y[3] + " | Booking: " + y[4] + " - " + y[5] + "\n"**

**ELSE:**

**OUTPUT "--- No records --- "**

**IF option EQUALS 3:**

**if LENGTH allRecords > 0**

**OUTPUT("\n\t All Rental History records \n")**

**FOR y IN allRecords:**

**OUTPUT("Plate#: " + y[0] + " | Booked By: " + y[1] + " | Total Paid : " + y[3] + " | Booking: " + y[4] + " - " + y[5] + "\n")**

**ELSE:**

**OUTPUT "--- No records --- "**

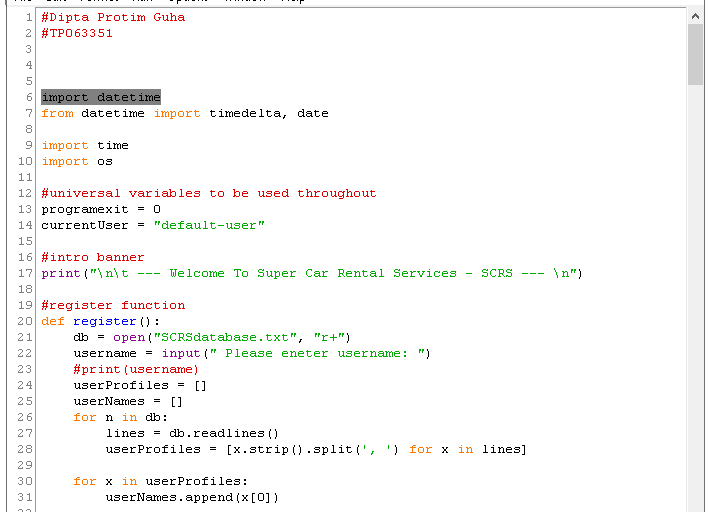
**CALL adminOptions METHOD**

**while(programexit!=1):**

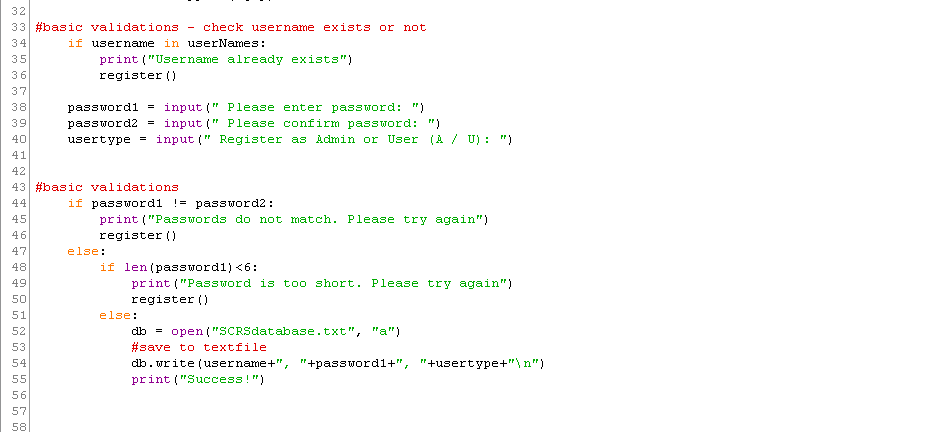
**homemenu()**

**Program Source code with explanation**

**Intro Banner and Register Function**

****

|  |  |
| --- | --- |
| Line Number | Explanation |
| 7-10 | Importing Python Library to be used for specific functions |
| 13 - 14 | Universal variables to be used through the program |
| 17 | Printing a string that reads to user when using the program |
| 20 | Define a function called register |
| 21 | File Handler |
| 22 | Take input from the user and setting to the variable, username |
| 23-24 | Setting a variable to an empty array |
| 26-27 | For loop asking to read lines |
| 28 | Strip and split functions used for formatting |
| 30-31 | Putting x in a for loop and appending the variable, userNames, which was defined as an empty array |



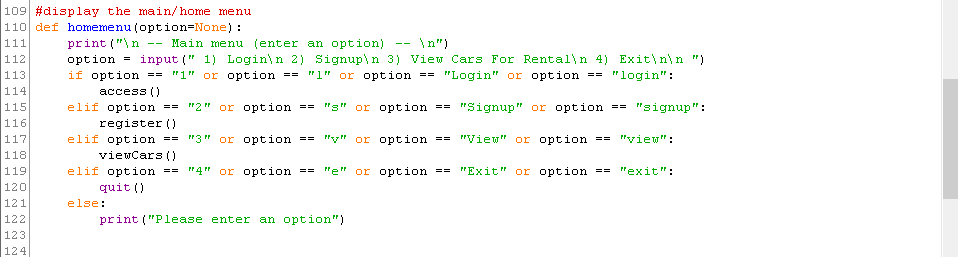
|  |  |
| --- | --- |
| Line Number | Explanation |
| 34-40 | If else statement validation to see if username typed already in the text file or not |
| 44-55 | If else statement to validate whether first password matches second password or if the length of the password is more than 6 character or not. The validation succeeds, it will print Success and append (not write as it overwrites existing data) the new username and password to the text file. |

**Access Function**



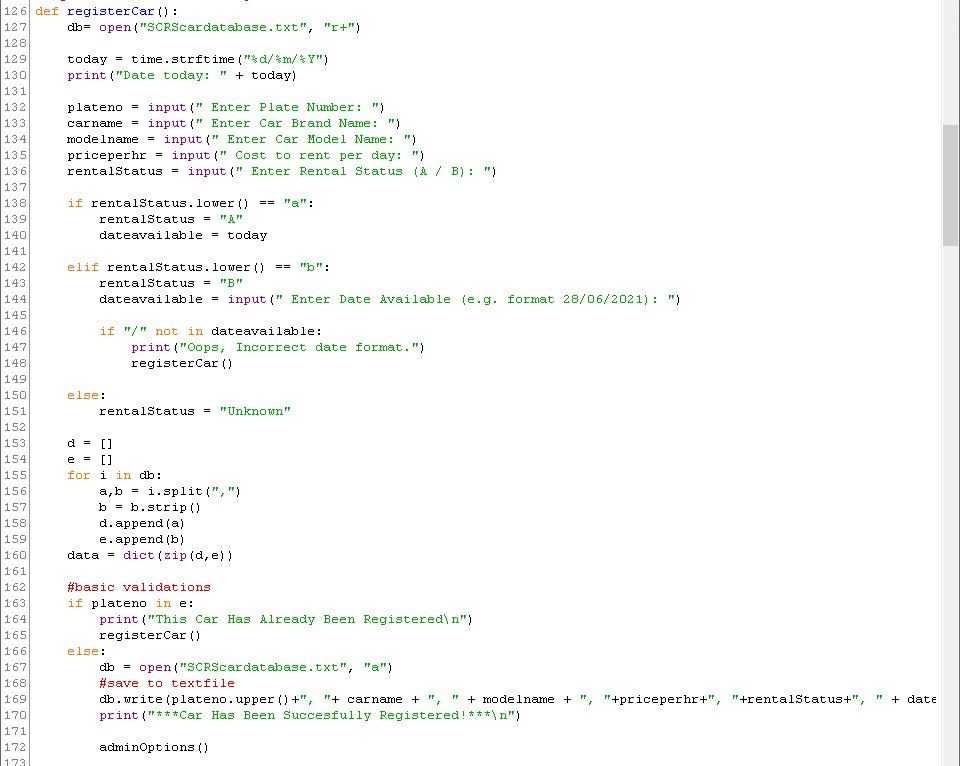
|  |  |
| --- | --- |
| Line Number | Explanation |
| 76 | Using the built in dict fuction which consist of a key with an associated value. |
| 78-104 | Using the try and except method to test and handle errors and indenting if else methods inside it. |

**Main menu function**



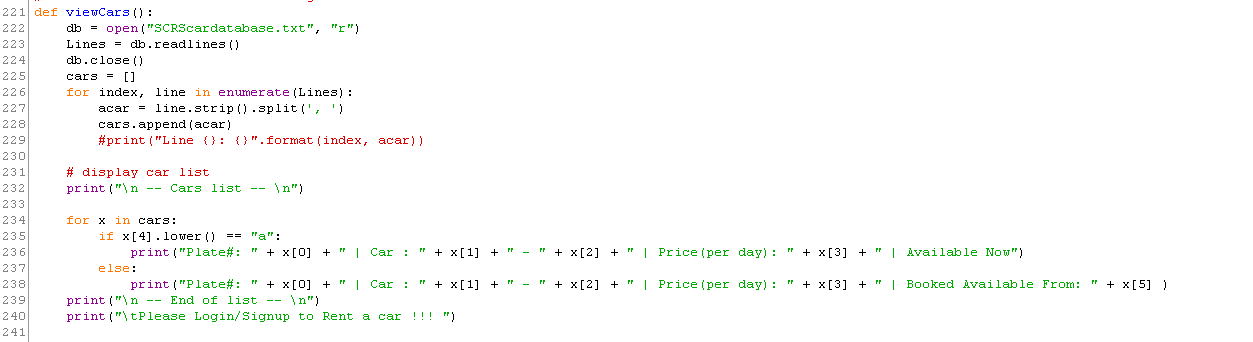
|  |  |
| --- | --- |
| Line Number | Explanation |
| 113-121 | Calling functions to run from user input, validating with if else statement method |

**Register car function**



|  |  |
| --- | --- |
| Line Number | Explanation |
| 127 | File handler |
| 129 | Using strftime python fuction to set a format for the date and assigning it to variable |
| 146-148 | For loop, printing a statement and rerunning the function if the format for the date is wrong |
| 169 | Writing to text file in append mode |
| 172 | Calling admin function to run after the completion of the ongoing function |

**View Cars Function**



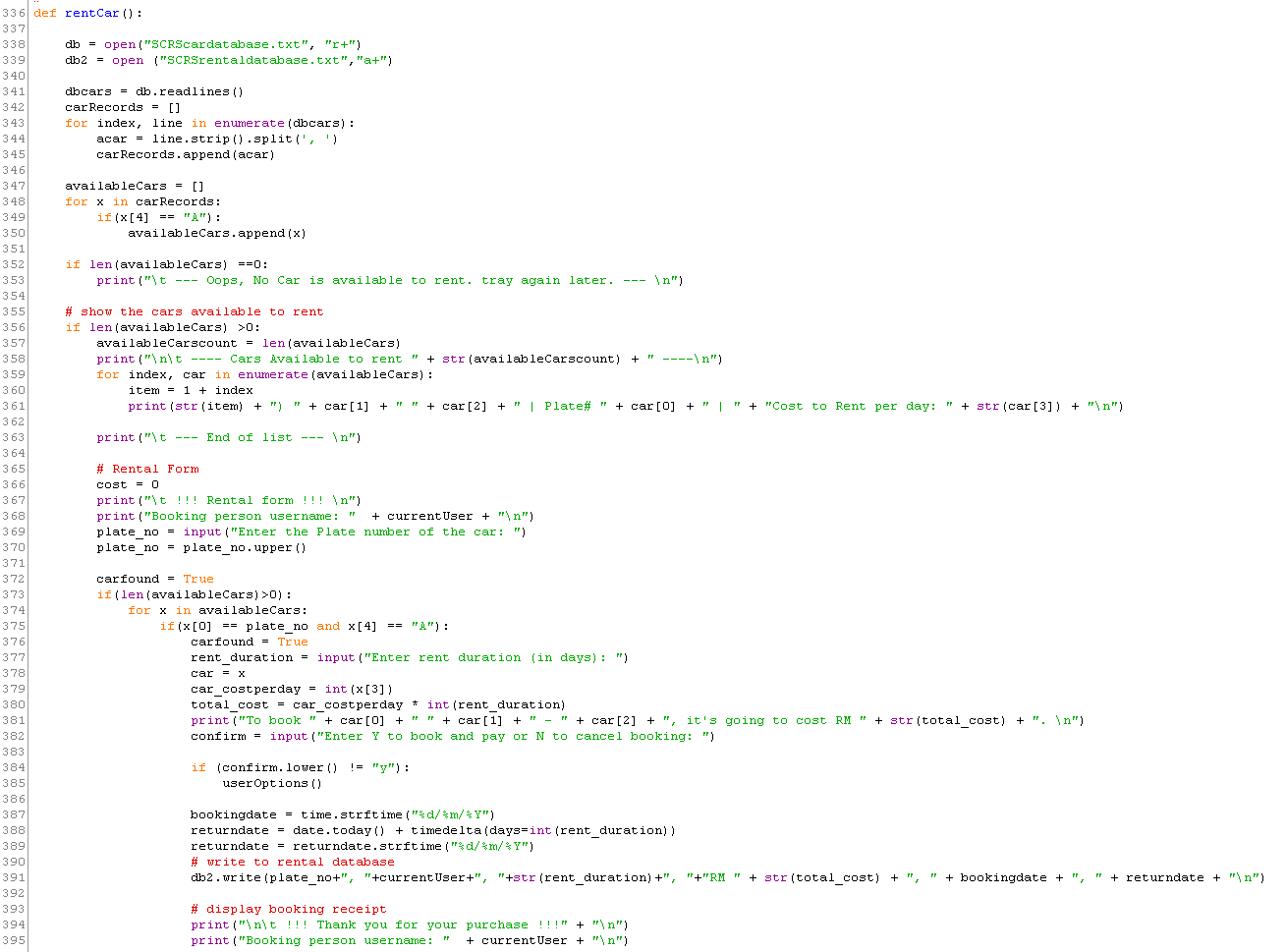
|  |  |
| --- | --- |
| Line Number | Explanation |
| 222 | Assigning variable to open text file in read mode |
| 226-228 | Using enumerate function to loop through the lines from the text file, format using strip and split and append each car to cars array |
| 232 – 239 | Printing out the list of cars available cars from the array |

**Search and modify car function**

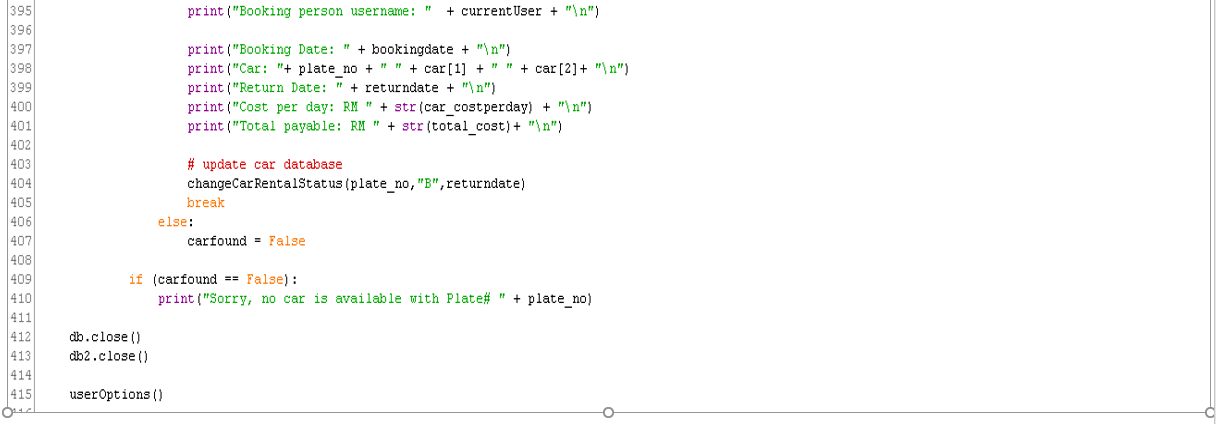


|  |  |
| --- | --- |
| Line Number | Explanation |
| 287-288 | Assigning file handler to variable  Opening car database text file in read mode  Creating a temp text file and open in write mode |
| 292 | Assigning S to an empty array |
| 293 to 297 | While loop used to search through the text file (being used in read mode) to match what the user input |
| 297-332 | If user input is found in the text file it will copy everything in the text file in read mode and edit all the other fields based on user input and write everything to tempt file opened in write mode |
| 324-325 | Closing the file handlers opened at the start of the function |
| 326-327 | Using the import os python method stated at the start of the code, remove the text file opened in read mode |
| 328 | Rename the tempt file opened in write mode to match the car database text file |

**Rent car function**

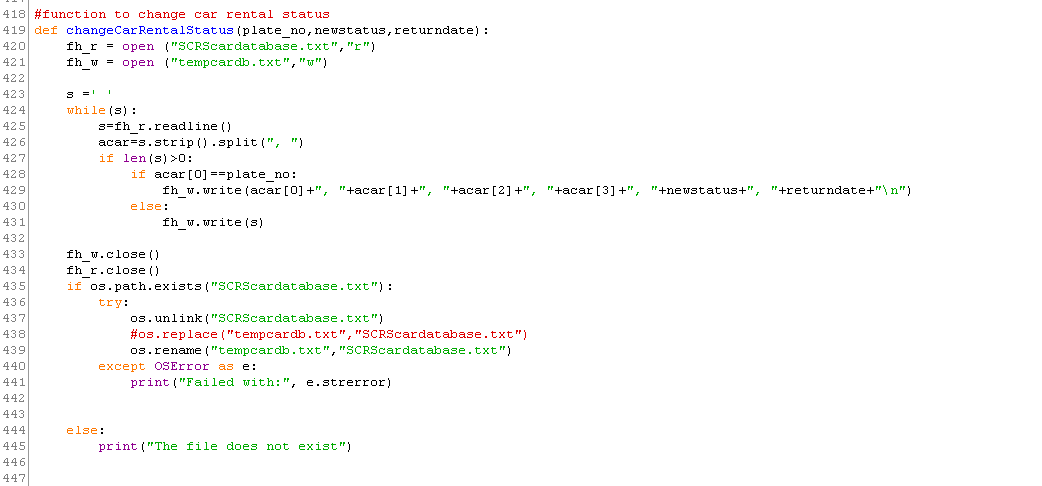


|  |  |
| --- | --- |
| Line Number | Explanation |
| 375 | Search available cars array for user requested car |
| 380 | Calculate total cost (cost per day\*rent duration in days) |
| 391 | Writing to rental database |



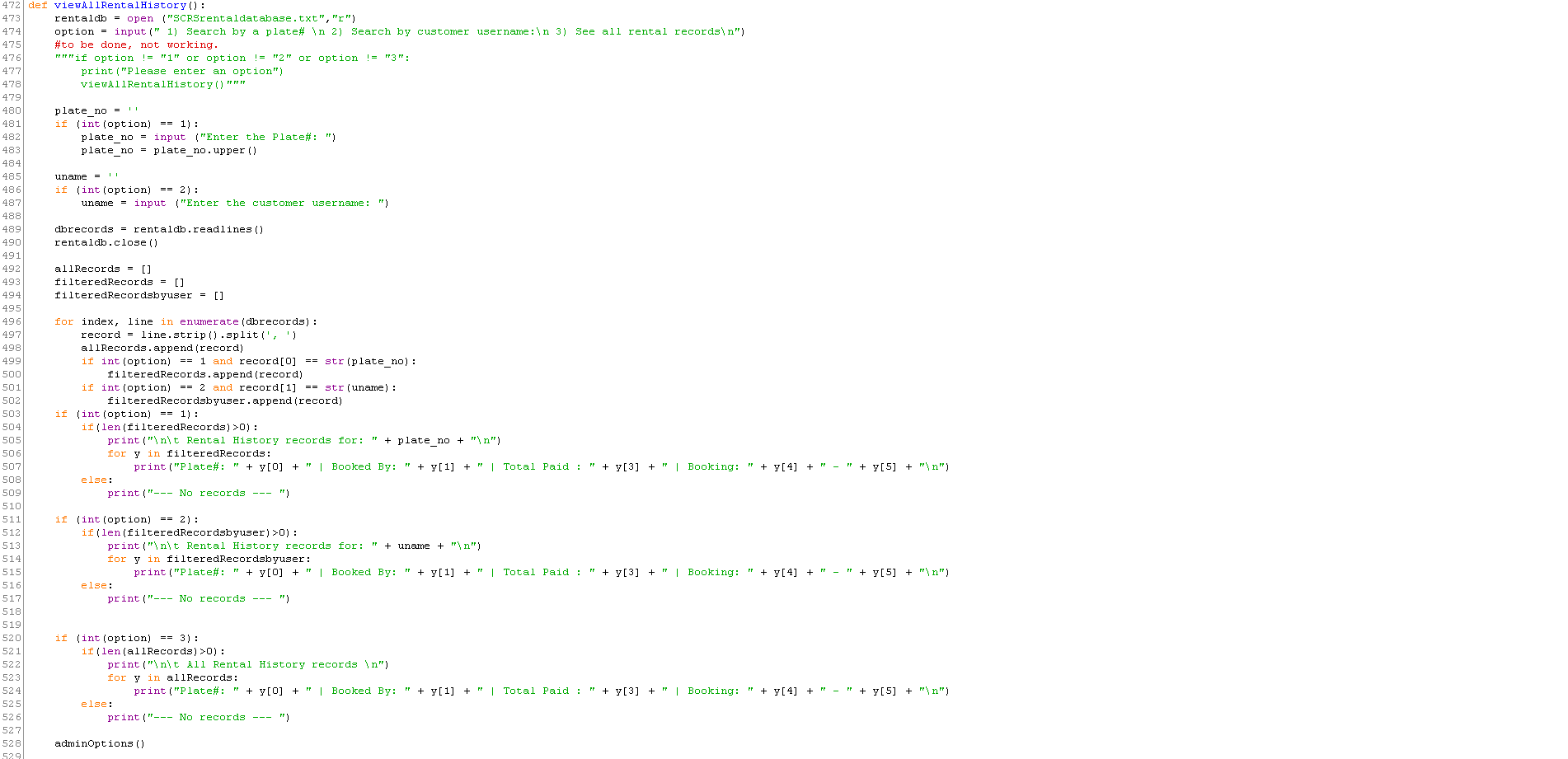
|  |  |
| --- | --- |
| Line Number | Explanation |
| 404 | Updating the car rental status in the text file |

**Change rental car status function**



|  |  |
| --- | --- |
| Line Number | Explanation |
| 337 | Alternative to OS.remove used to remove the text file |
| 440 | Returns system related errors e.g as file not found |

**View all rental history function**

****

|  |  |
| --- | --- |
| Line Number | Explanation |
| 480-483 | Taking user selection for viewing rental history based on plate number search |
| 485-487 | Taking user selection for viewing rental history based on username |
| 492-517 | Iterating and filtering line item based on user selection |
| 520-526 | Taking user selection for viewing all rental history |

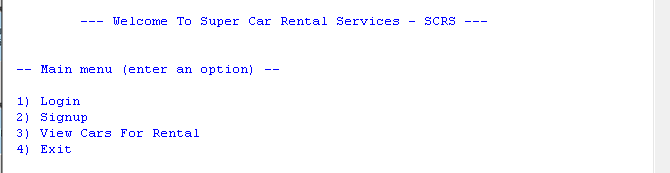
Rest of the code is repetitive and explained in either of the functions mentioned above.

**Sample Input/Output**

**Main Menu/Non-registered user view**

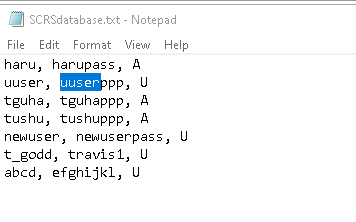
The program has a main menu which provides the following options which users can key in to move forward with the program. The programs are:

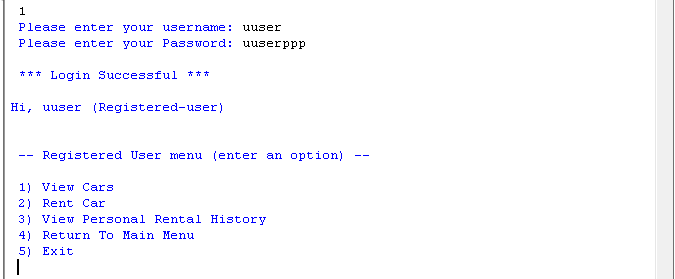
1. Login
2. Signup
3. View All Cars
4. Exit

****

**Login**

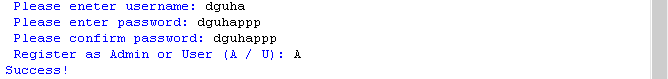
When users want to login they will be required to key in their username and password which the program will try to read from a text file, SCRSdatabase.txt.

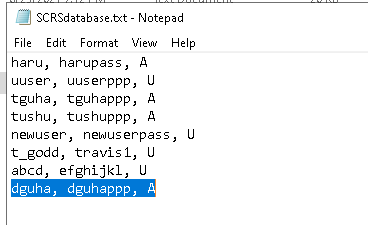
****

****

**Signup**

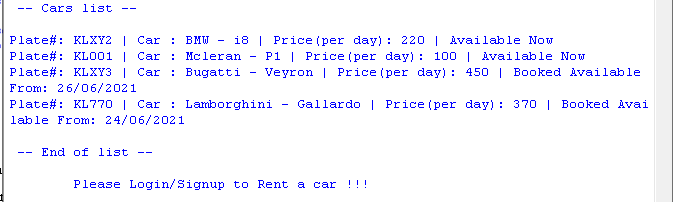
The text file can be updated when users select the register function. Here they can register with new username and password and select whether they wish to register as a user or an admin. The text file has a 3rd column which distinguishes between the types of users. If the username and password belong to an admin, they will be presented with the admin menu and if the username and password belongs to a user, they will be presented with the user menu.

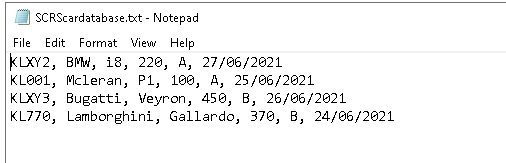
****

****

**View All Cars**

If user press 3 and select the option to view all cars, data from another text file known as SCRScardatabase.txt, will be read by the program and presented to the user to see.

****

****

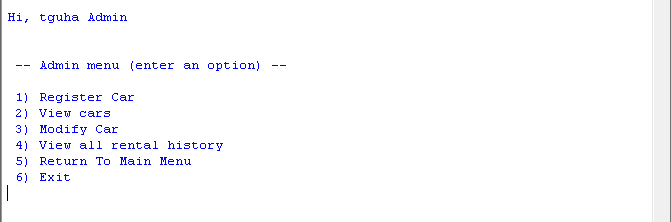
**Exit**

Finally, selecting option 4 will kill the program.



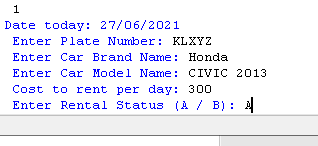
**Admin View**

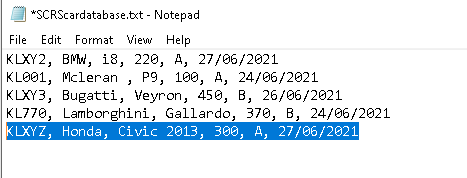
In the Program, the admin view looks like following screenshot:



**Register Car**

If the user keys in 1, it will let the user add a new car to the car database, SCRScardatabase.txt, text file. The program will ask user to key in the car plate number, the brand, the model and the availability. Once all these inputs are keyed in, the program will print “\*\*\*Car Has Been Succesfully Registered!\*\*\* ”. The text file will also be update with the new car.

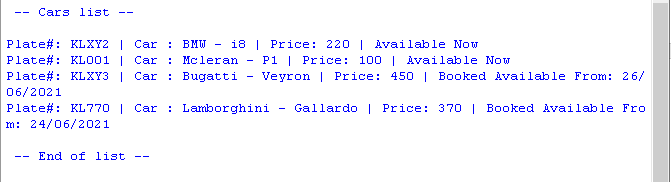




New car registered has been added to the SCRScardatabase.txt txt file.

**View Cars**

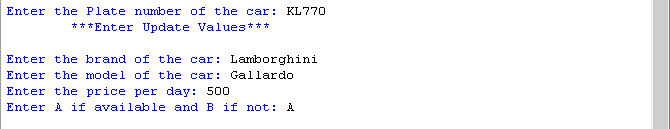
If the user selects the option number 2, the program will do a similar thing the option 3 in the main menu. It will read from the text file, SCRScardatabase.txt, and present all the car information to the admin.

****

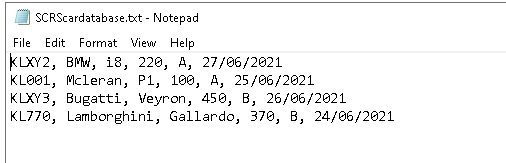
**Modify Cars**

Option 3 lets admin to modify car specifics in the SCRScardatabse.txt file. The text file is broken down in 6 columns. Plate no. of the car, the brand of the car, the model of the car, rent price per day of the car, the availability of the car which is represented by either A or B in the text file and lastly the next available date. User can search the specific car they want to modify by searching a car with the plate number. Once found, they can modify the car name, rent price and availability of the car. If the car is selected as available, the next available date will always show the date of the current day.

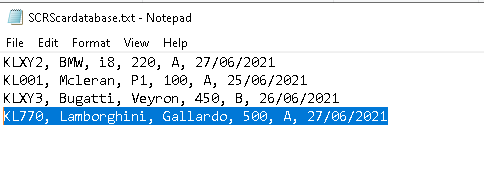
Admin can also return a car when the renting period is over by simply searching for the car and modifying the availability of the car from B (booked out) to A (available) and keeping the rest of the car details the same.

****

Before

****

After

****

**View All Rental History**

Selecting option 4, “View all rental history” presents the user with 3 options:

1) Search by a plate#

2) Search by customer username:

3) See all rental records

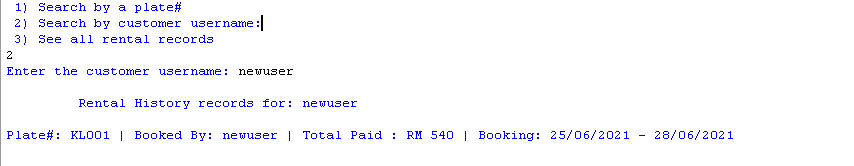


When either of these options are selected, information from another text file, SCRSrentaldatabase.txt, will be pulled by program and displayed to the user based on the options they select.

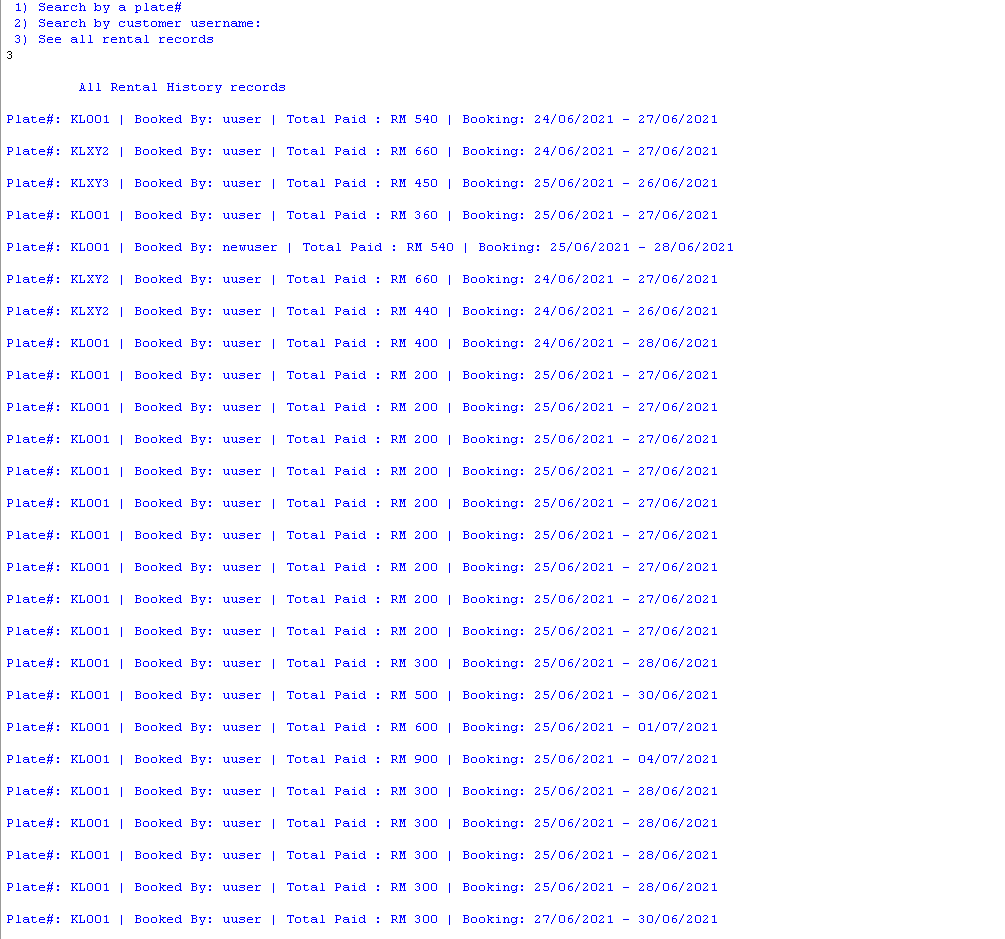
If the user selects option 1, “Search by a plate#”, admin can key in the plate number of a specific car and see all rental history of that car.



If the user selects option 2, “Search by customer username”, user can all the rental details of a specific customer.

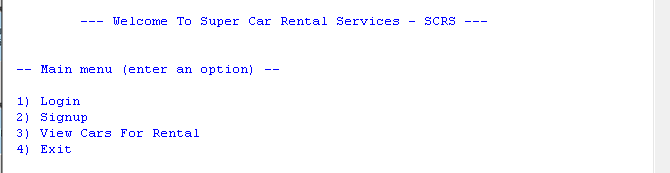


If option 3,” See all rental records”, is selected, user will be presented with all rental details of all cars and customers without any filtration.



Lastly, selecting option 5 from the admin menu, will take user back to the main menu and selecting option 6 will kill the program.

Option 5

****

Option 6



**Registered-User View**

Registered-User view presents the user with the following view:

“

Hi, (Name of the user or username) (Registered-user)

-- Registered User menu (enter an option) –

1) View Cars

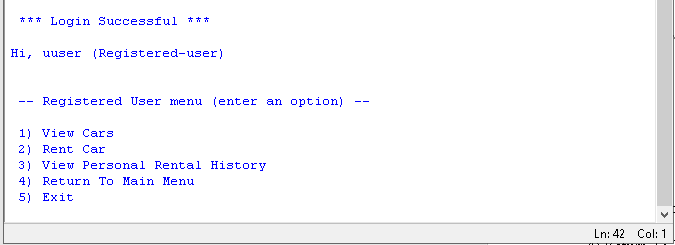
2) Rent Car

3) View Personal Rental History

4) Return To Main Menu

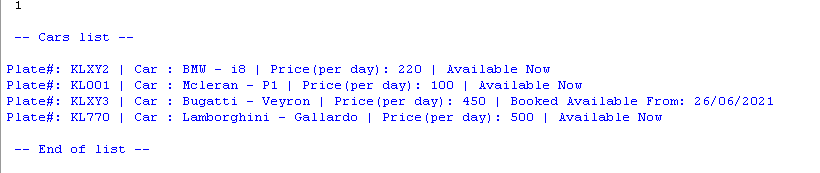
5) Exit

”



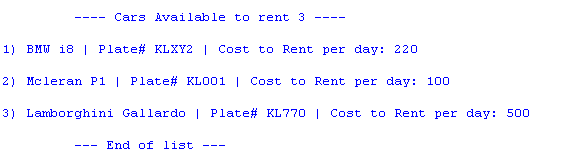
**View Cars**

When user selects option 1, “View Cars”, the same process of the program pulling data from SCRScardatabase.txt text file and representing all the information of the cars to the user, will be executed.

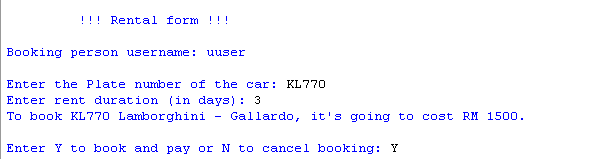
****

**Rent Car**

If user selects option 2, “Rent Car”, the program will read from the SCRScardatabase.txt text file and filter it to show user with only the cars available for rent.



Then the program will ask the customer to key in the plate number of the car they want to book. Once the plate number is keyed in, the program will ask user to select the number of days, the user wants to rent the car for.



Once program takes the number of days input from the user, it will do a quick calculation to provide the customer with the total amount it will take to rent the car for the specific duration. Then the program will ask if the user wants to continue with the payment (Y for yes and N for no). If the user selects Y, the program will print a ticket with information:

“!!! Thank you for your purchase !!!

Booking person username:

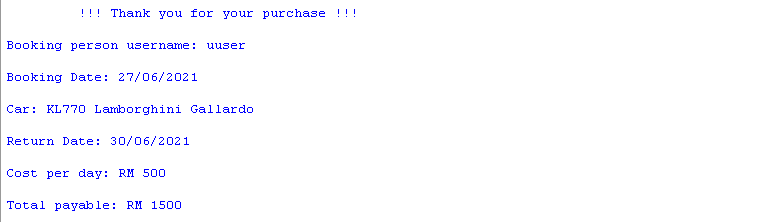
Booking Date:

Car: KL001

Return Date:

Cost per day:

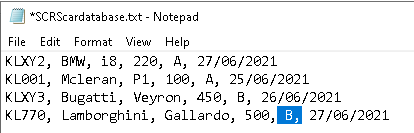
Total payable”



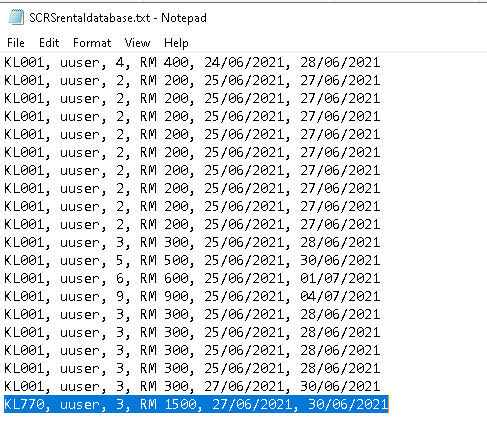
Simultaneously, the program will also modify, SCRScardatbase.txt text file to change the availability of the car from A (available) to B (booked-out) and also add a new record to the SCRSrentaldatabase.txt text file. The SCRSrentaldatabse.txt text file is the following format:

Car plate number, user who rented the car, number of days the car has been rented for, the total amount to rent the car depending on the amount of days the car had been rented for, the date it was rented and next available date of the car.

Car availability changed from A to B as user rented this car:

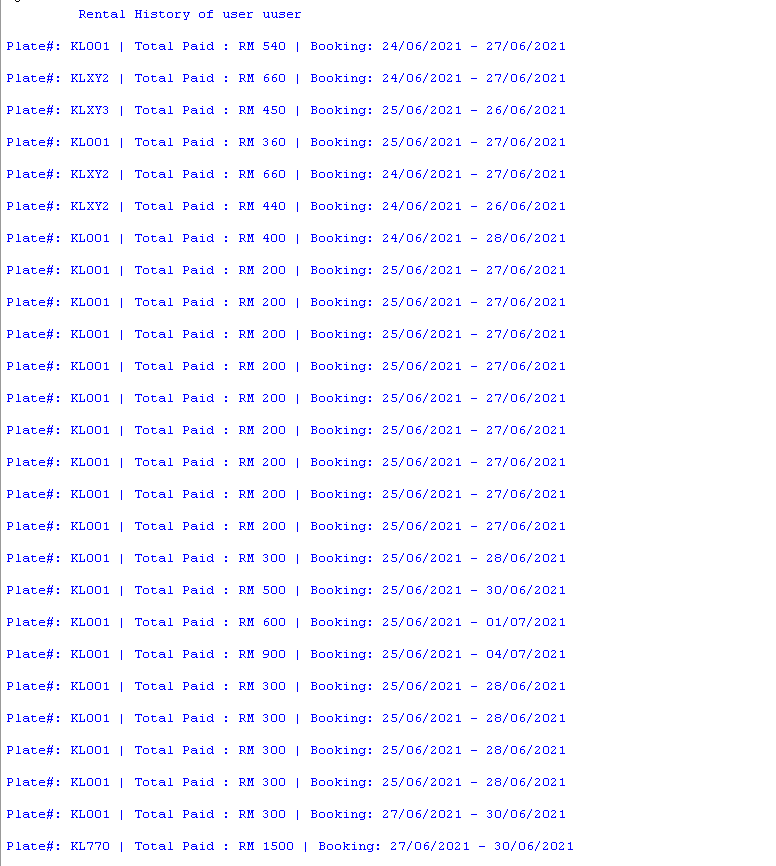


New rental information added to SCRSrentaldatabase.txt text file:



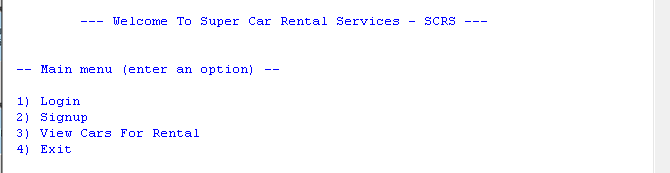
**View Personal Rental History**

If the user selects option 3, “View Personal Rental History”, the program will read the SCRSrentaldatabase.txt text file and present the user with only the rental history of cars related with that specific user.

****

Lastly, selecting option 4 from the admin menu, will take user back to the main menu and selecting option 5 will kill the program.

Option 4

****

Option 5



**Conclusion**

In Conclusion, the program runs smoothly with little or no hiccups. It covers all the functionalities required for users to quickly rent a car. This eliminates the need for customers to go to a designated place, wait in a queue and physically book a car. They just need to book the car and have admin figure out a way to deliver the car to the customer for the number of days he/she has booked the car for.

There are quite a few things that could have been done differently to provide a better ease of access for the customers. A graphical user interface could be added to make the program look nicer. Also, .csv files could be used to generate lists or information much better compared to text files. Although these were not required by the assignment question, these implementations could help come up with a more real-world program that customers could use on a daily basis.

**References**

AskPython. n.d. *How to convert a list to a dictionary in Python? - AskPython*. [online] Available at: <https://www.askpython.com/python/list/convert-list-to-a-dictionary> [Accessed 6 June 2021].

AskPython. n.d. *How to Rename a File/Directory in Python? - AskPython*. [online] Available at: <https://www.askpython.com/python/examples/rename-a-file-directory-python> [Accessed 5 June 2021].

Stack Overflow. n.d. *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Available at: <https://stackoverflow.com/> [Accessed 3 June 2021].