GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

AAPP010-4-2-PWP

PROGRAMMING WITH PYTHON

UCDF2005-ICT(DI)

HAND OUT DATE: **08TH APRIL 2021**

HAND IN DATE: **18TH JUNE 2021**

WEIGHTAGE: **100%**

Name	Tin Eugene	Chia Wen Xuen
Student TP Number	TP061195	TP061184

INSTRUCTIONS TO CANDIDATES:

1. Submit your assignment online in MS Teams unless advised otherwise
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

1.0 Introduction 3
2.0 Assumption 4
3.0 Flowchart 5
4.0 Pseudocode 19
5.0 Source code 43
5.1 Utilities Functions 43
5.1.1 Read file 43
5.1.2 Write file 43
5.1.3 Rental expiration 44
5.2 User Functions 45
5.2.1 User Register 45
5.2.2 User Login 46
5.2.3 Modify user details (User/Admin) 47
5.2.4 Add funds into wallet (User) 49
5.2.5 View rental history (User) 50
5.3 Vehicle Functions 51
5.2.1 Rent Car (User) 51
5.3.2 Add Car (Admin) 52
5.3.3 Modify Car (Admin) 53
5.3.4 Rented out car's detail (Admin) 54
5.3.5 Customer query (Admin) 55
6.0 Additional Features 56
7.0 Sample input/output with explanation 56
7.1 Login as admin 59
7.2 Register 62
7.3 View Cars 63
8.0 Conclusion 64
9.0 Limitations 64
References 65

1.0 Introduction

In this era of modernization, it is inevitable that business models are having a drastic move into online business due to the advancement of Internet and digital gadget such as smartphone and laptop. Super Car Rental Service must follow the trend to remain competitive in the car rental business field. Online Car Rental System, OCRS software has been developed to override the problems prevailing in the practicing of offline business model. This software is built using python programming language as it has extensive support libraries, and user-friendly data structures which may benefits the freshly started online business software. Python is open source and has a extensive community to improve development process as the program can be available in the digital market as early as possible compared to other competitors. This system is designed to be user-friendly and ease the process for customer to place rental orders online without physical presence on the rental workshop. Integrated with in app wallet that supports wide varieties of payment method to simplify the payment. OCRS, as described above, can lead to non-error, secure, reliable and fast management system. It can assist the user to concentrate on their trip and activities other than making a detour to pick up a rental vehicle on the workshop on a distance. Thus, it will help the growth of Super Car Rental Service organization.

2.0 Assumption

Below is a list of **assumptions** that are taken into consideration when creating the program:

- Program will be developed using Python programming language.
- Database will be implemented using text file.
- There are two main phases of main user interface for security purposes:
  - UI without user logged in will only display vehicles to be rented along with login and register function.
  - UI that are logged in will determine if.
    - ✓ User is an admin: display admin functionalities.
    - ✓ User is not an admin: display car rental services and wallet top up feature.
- There are two entities included in the program, including **user** and **vehicle**.
- The system will not allow non administrator user to access user personal information and vehicle modification.
- Administrators can add, modify, and update vehicle details and status manually.
- User personal information can be updated within the program.
- User are allowed to rent car that are not rented by other users only, rental history and details can be access personally and administrator.
- User credentials are securely encrypted with SHA256 asynchronous encryption algorithm.

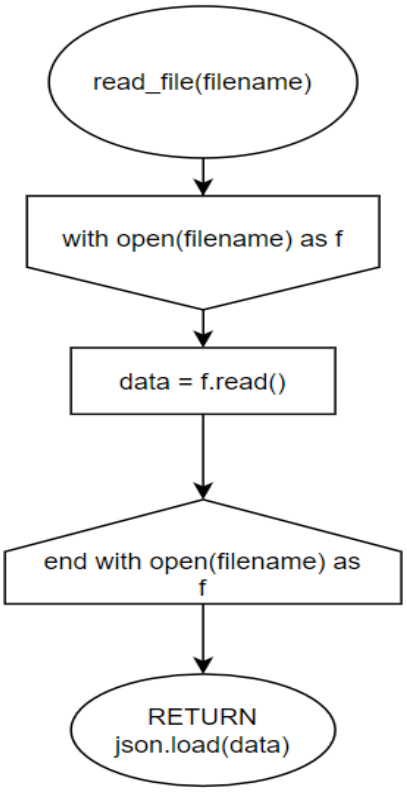
Below is the list of **convention** that are obeyed throughout the program:

- Return multiple statements using list.
- Double quotes for string data type.
- Snake casing variables.
- Store values in the form of JSON data type and parse into file.
- Tabs are preferred, default indentation = 2.
- Four classes of function: utilities, users, vehicles, and user interface.
- Two entities, which is userlist.txt and carlist.txt.

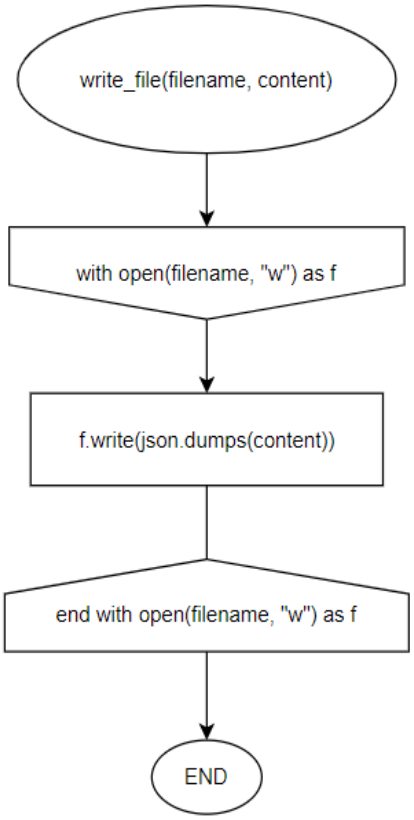
Admin credentials: username = “admin”, password = “admin”

Customer credentials: username = “wenxuen”, password = “wenxuen”

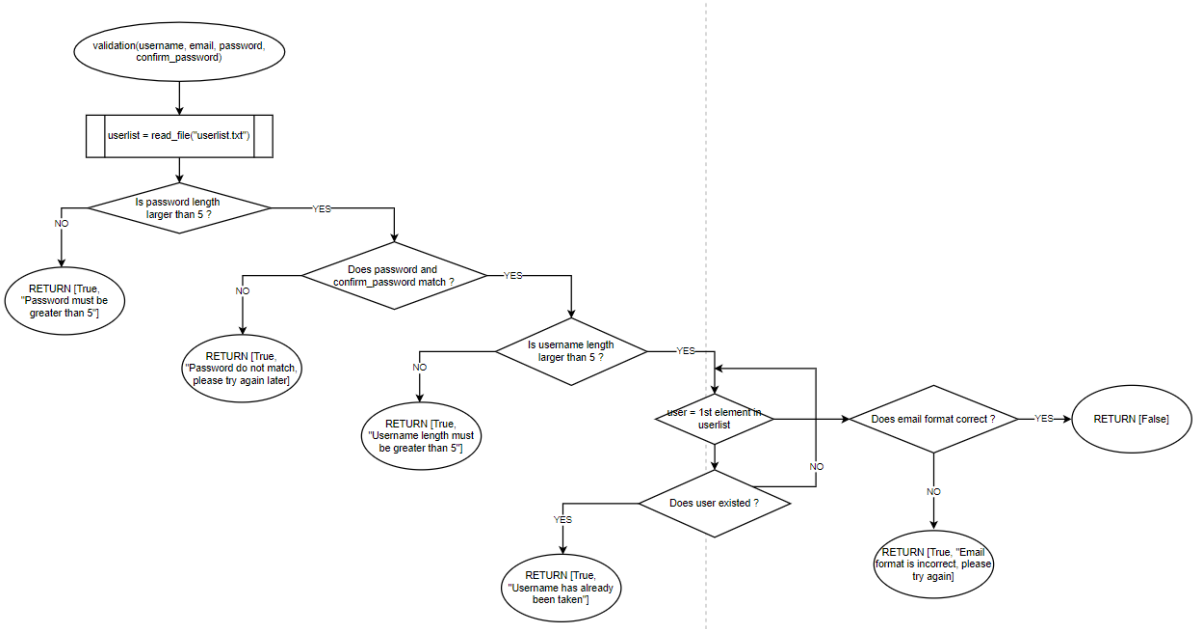
3.0 Flowchart



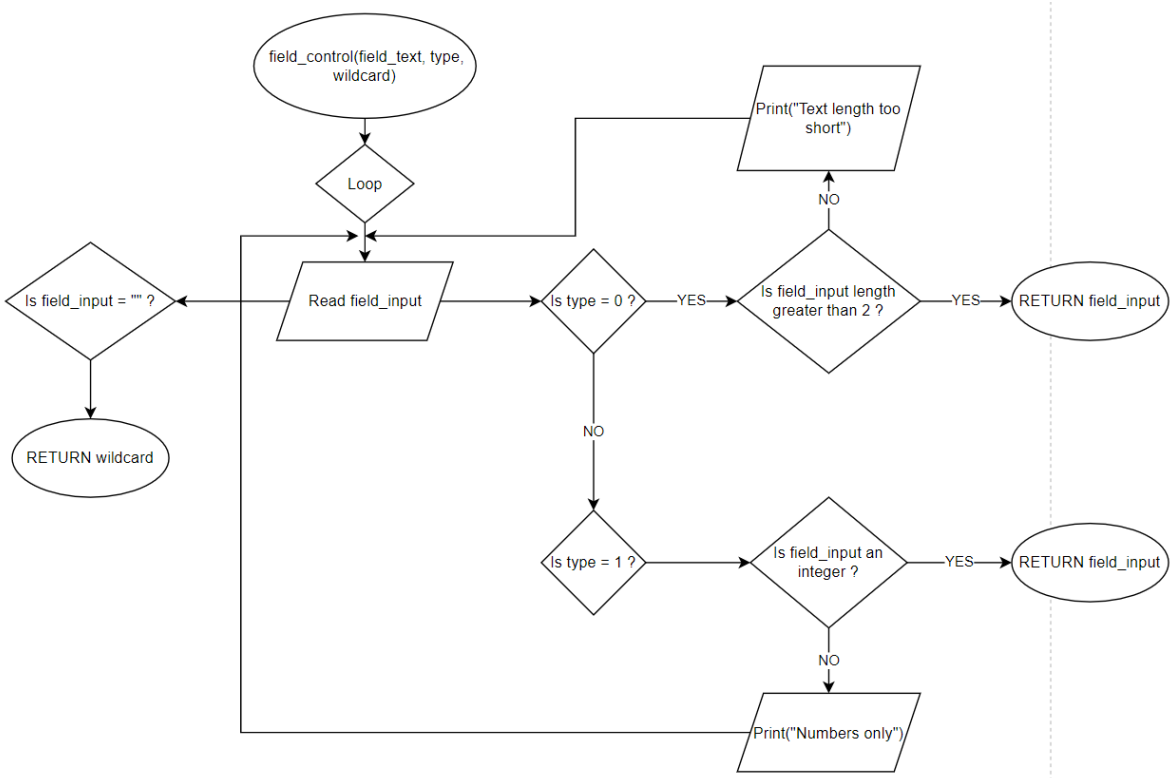
Flowchart 2 read_file



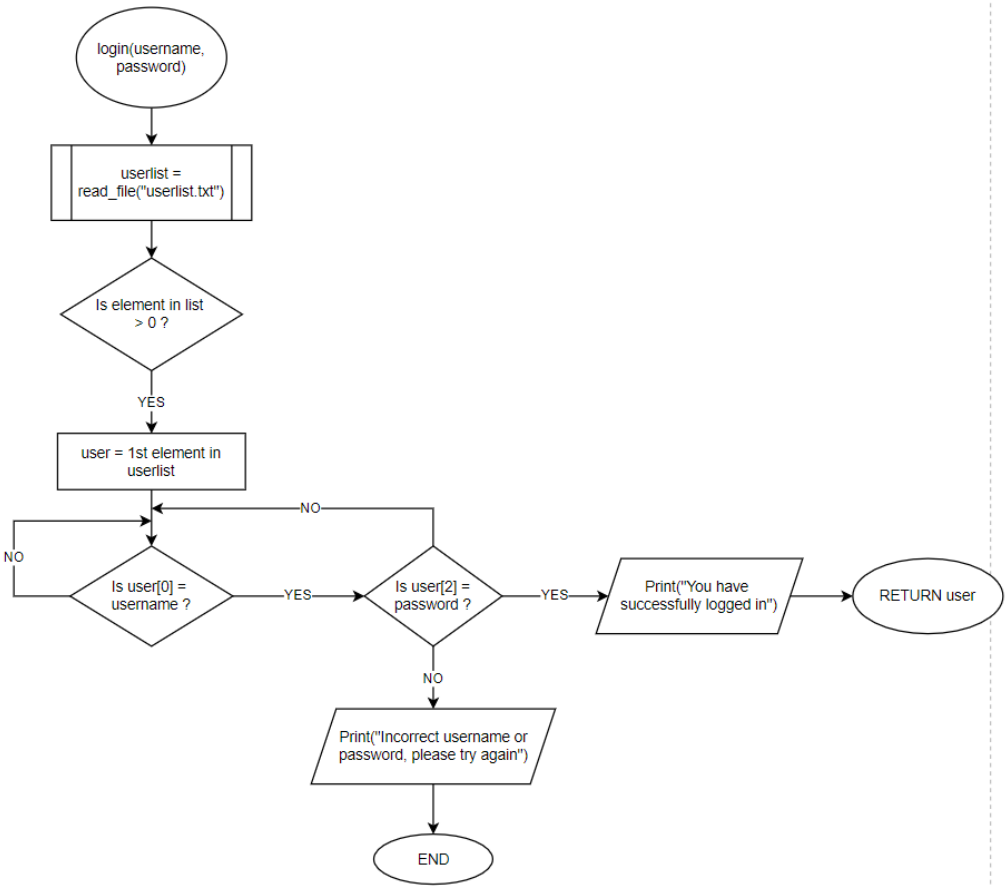
Flowchart 1 write_file



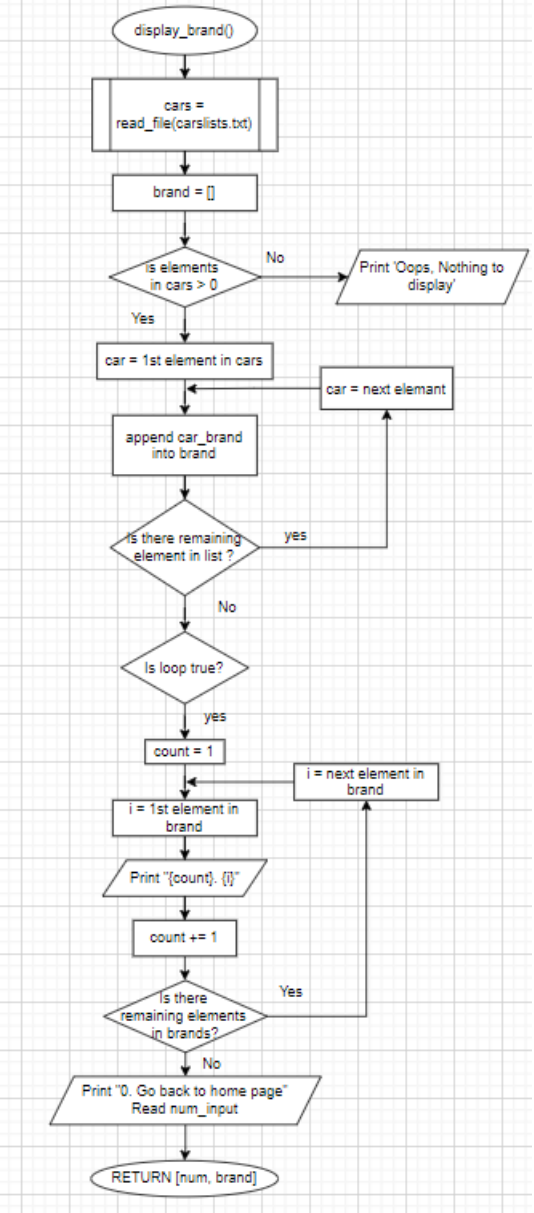
Flowchart 3 data_validation



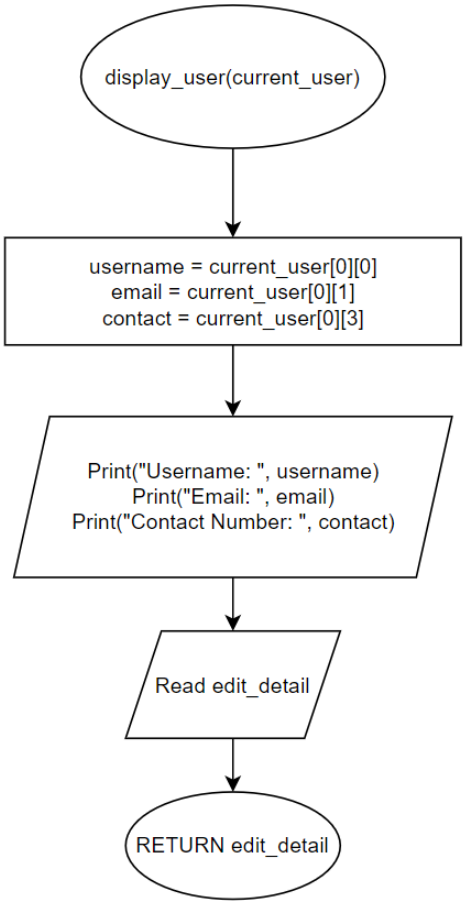
Flowchart 4 Field_control



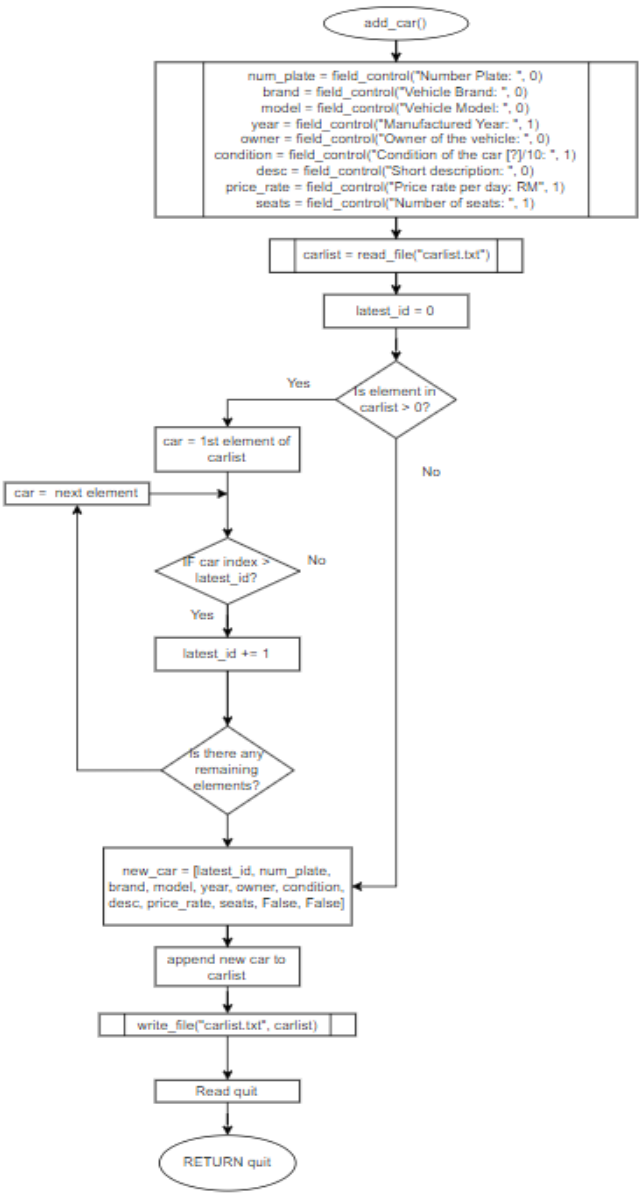
Flowchart 5 login



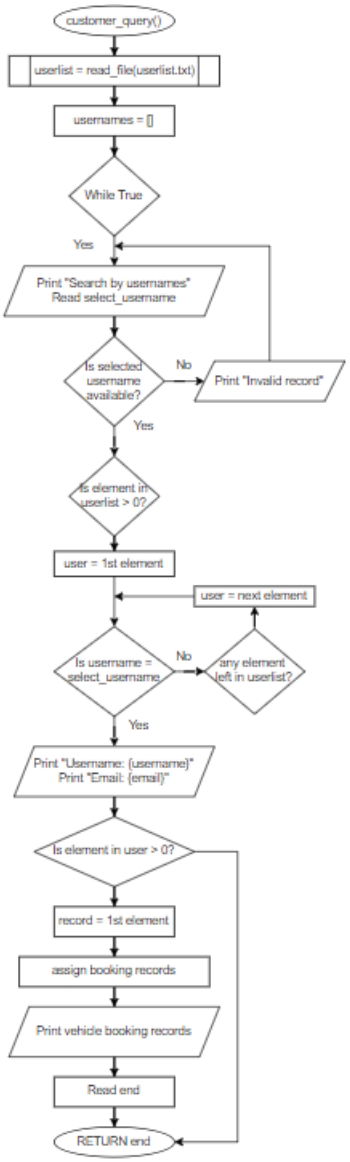
Flowchart 6 display brand



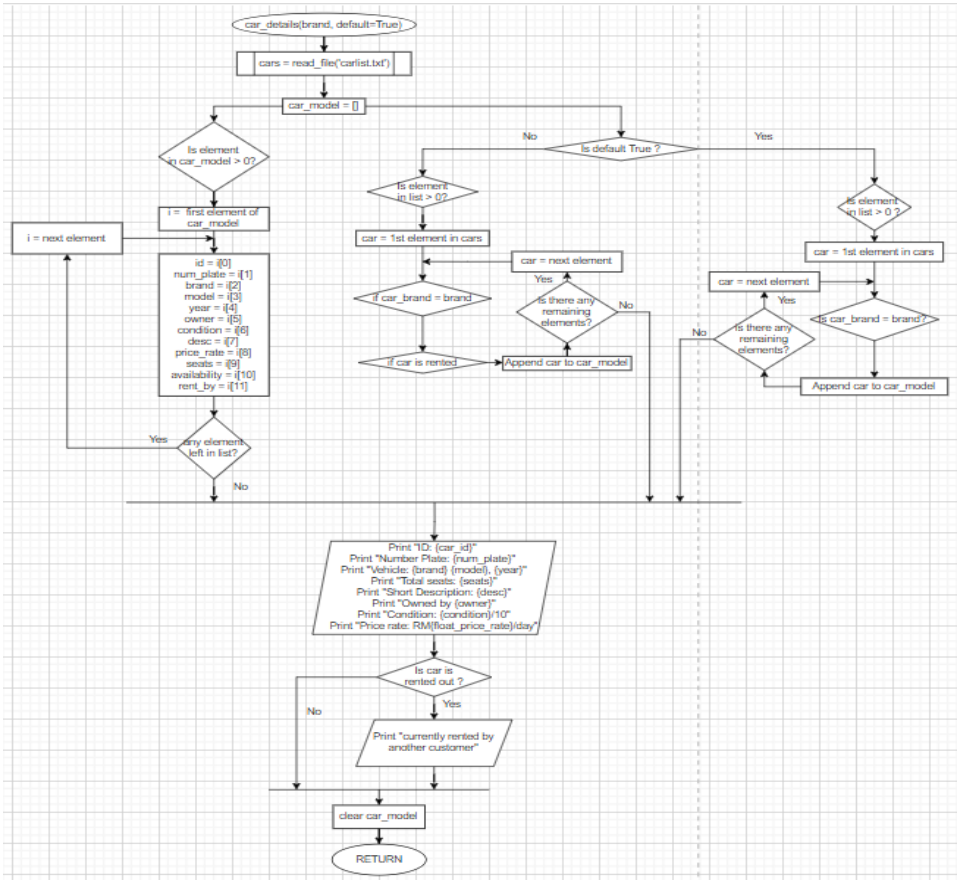
Flowchart 7 display_user



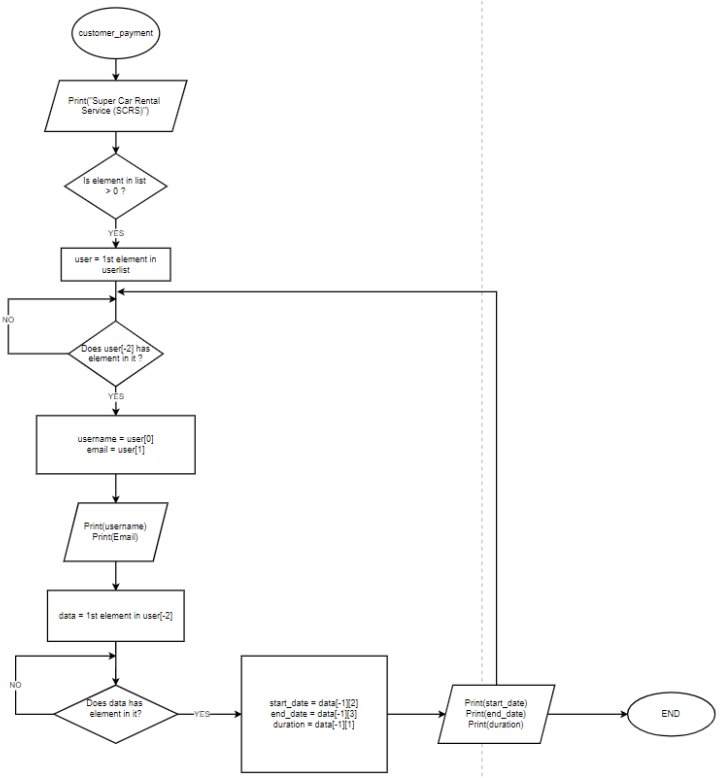
Flowchart 8 add_car



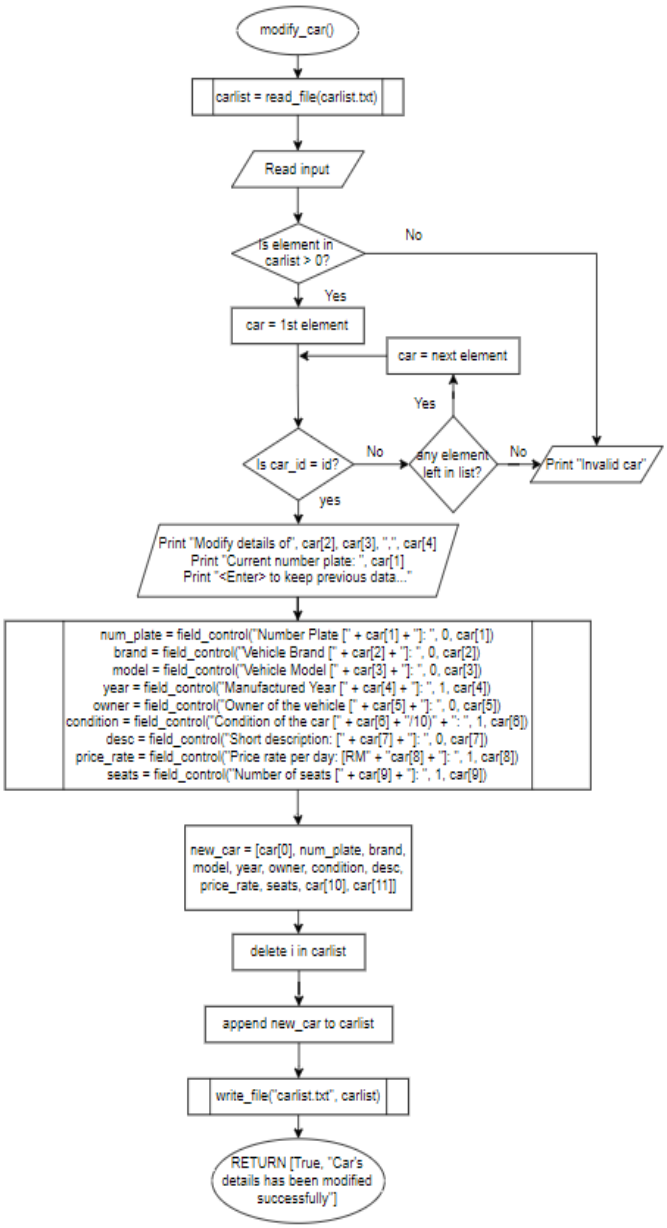
Flowchart 9 customer_query



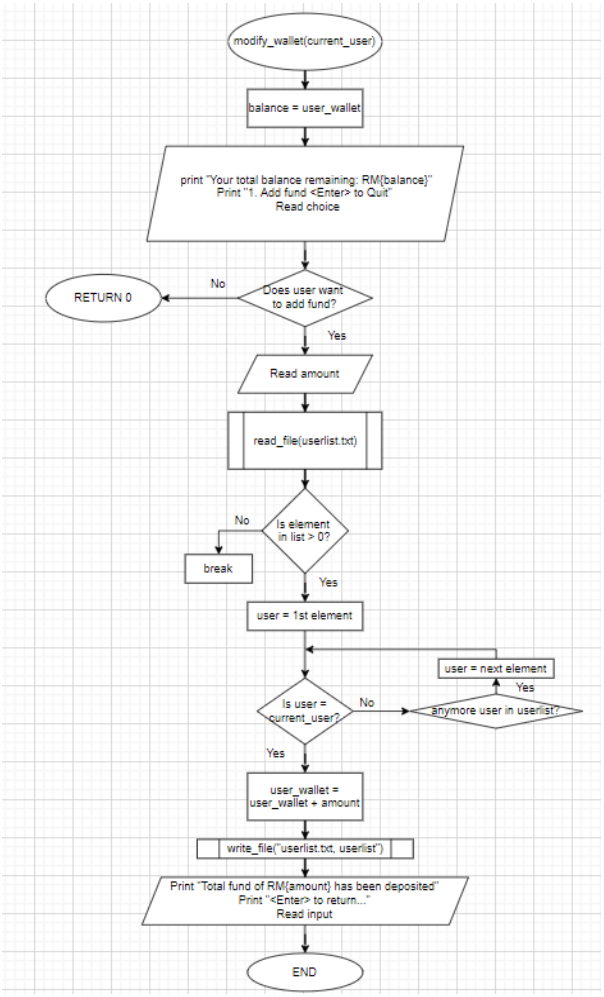
Flowchart 10 car details



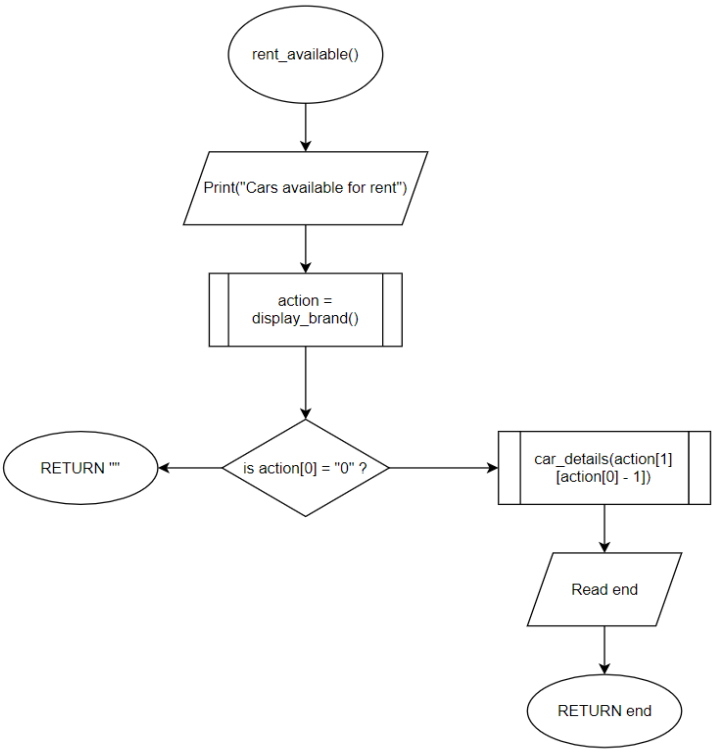
Flowchart 11 customer_payment



Flowchart 12 modify_car

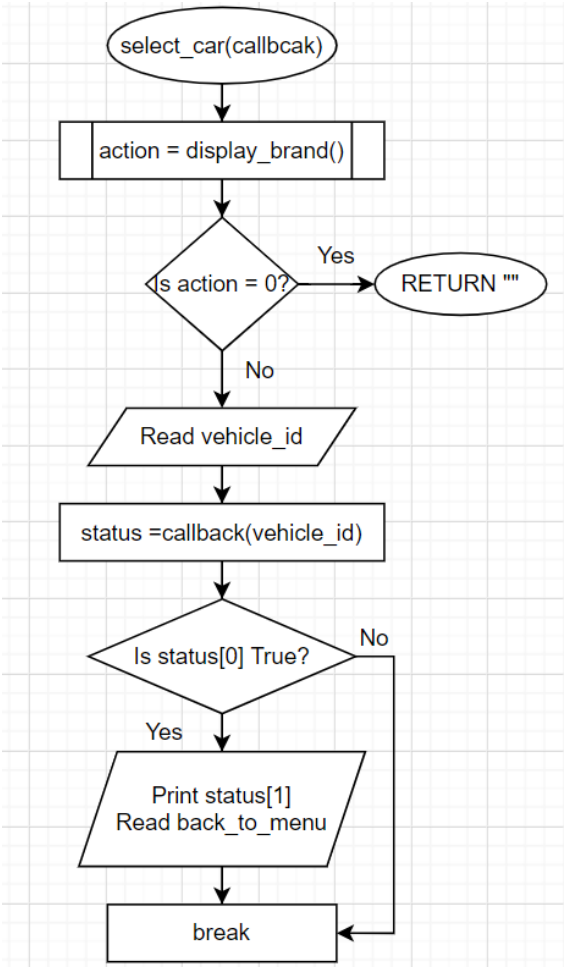


Flowchart 13 modify_wallet

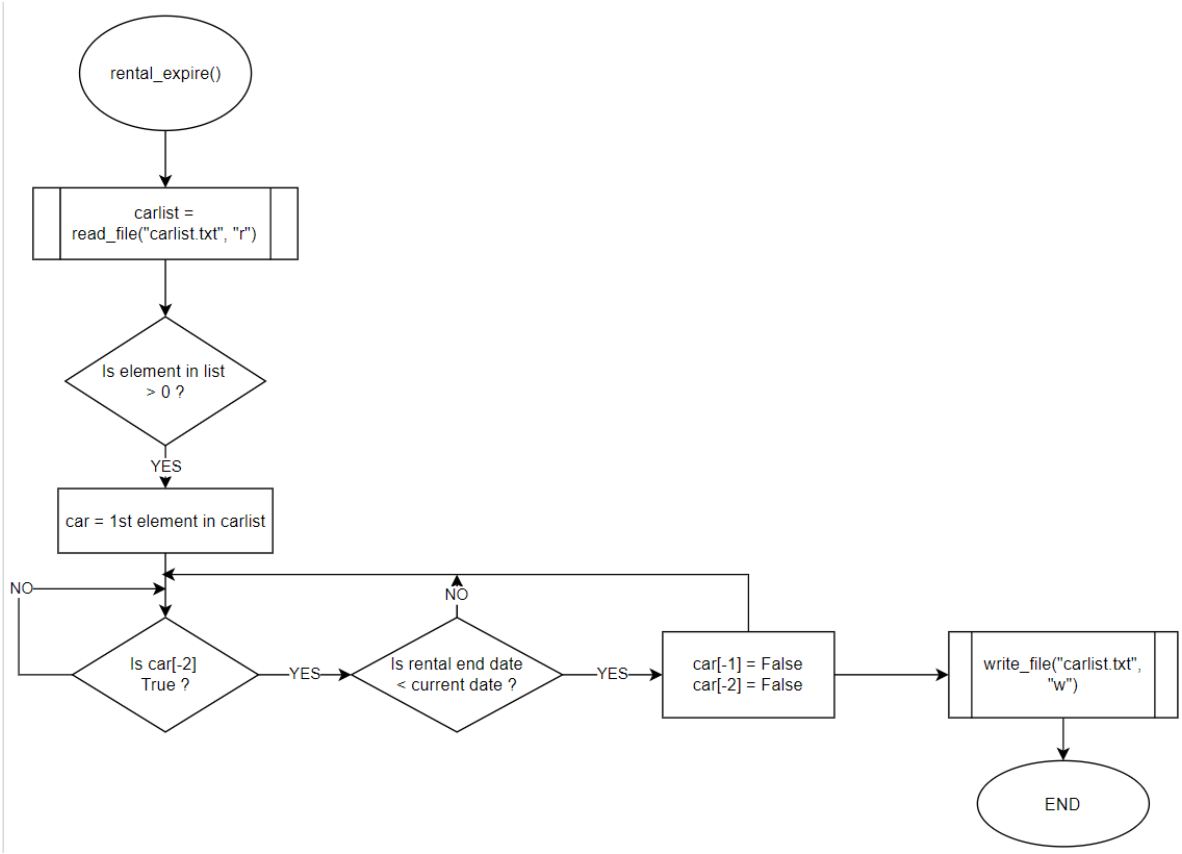


Flowchart 14 rent_available

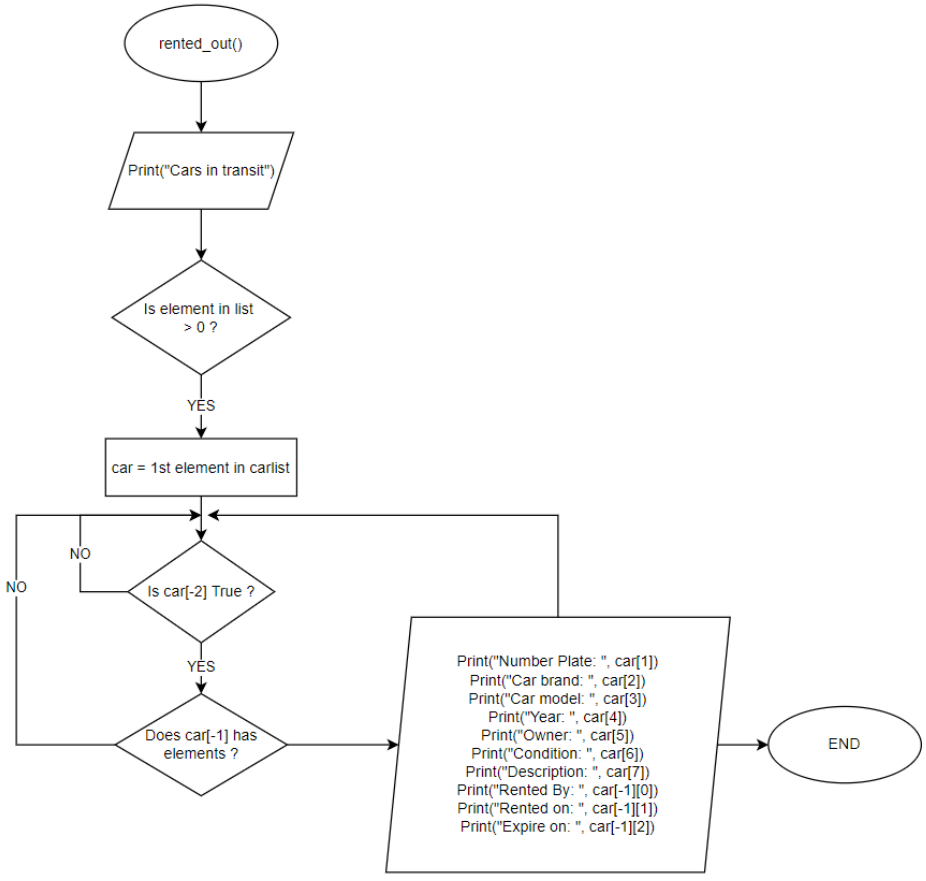
Flowchart 15 rent_car



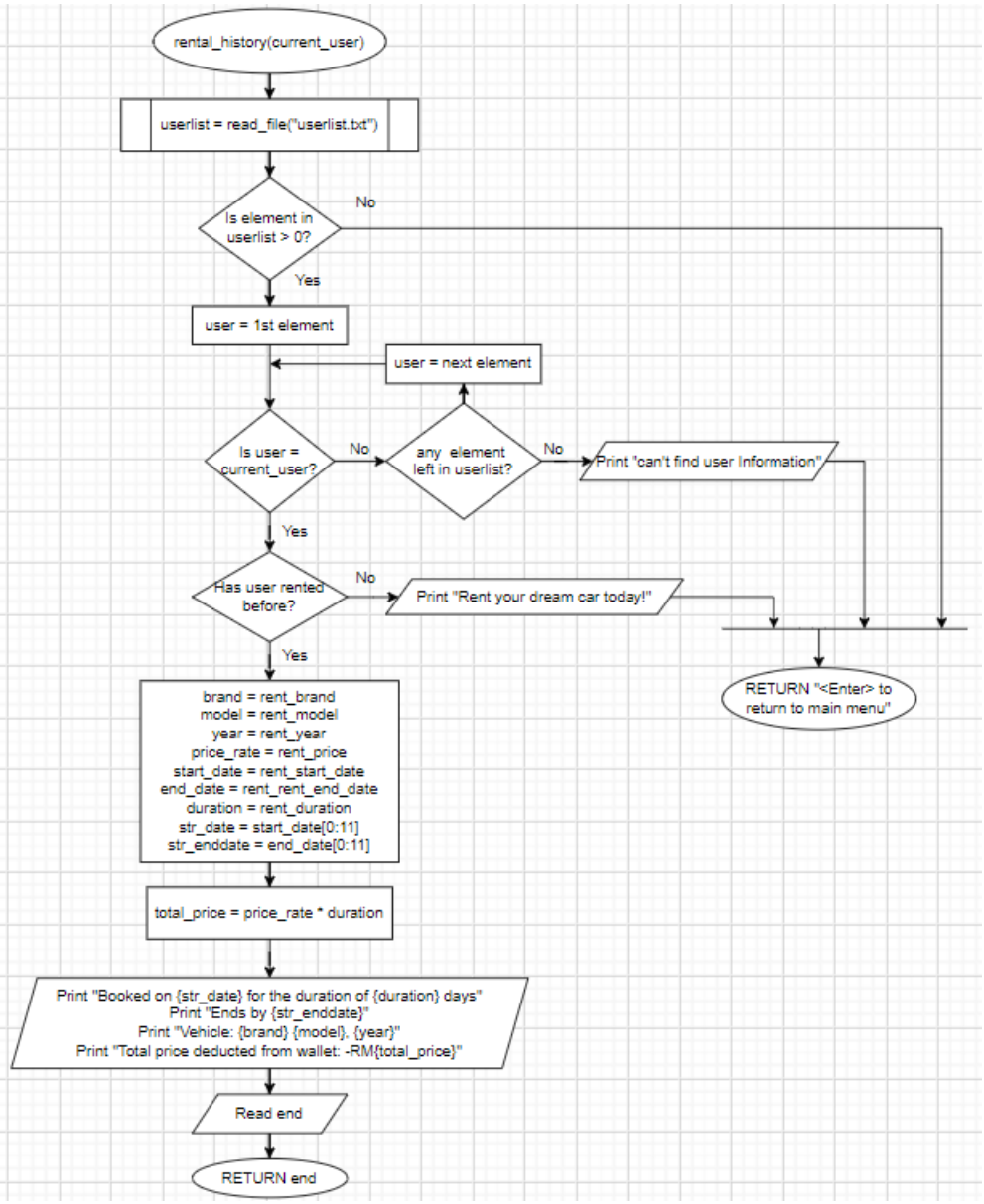
Flowchart 16 select_car



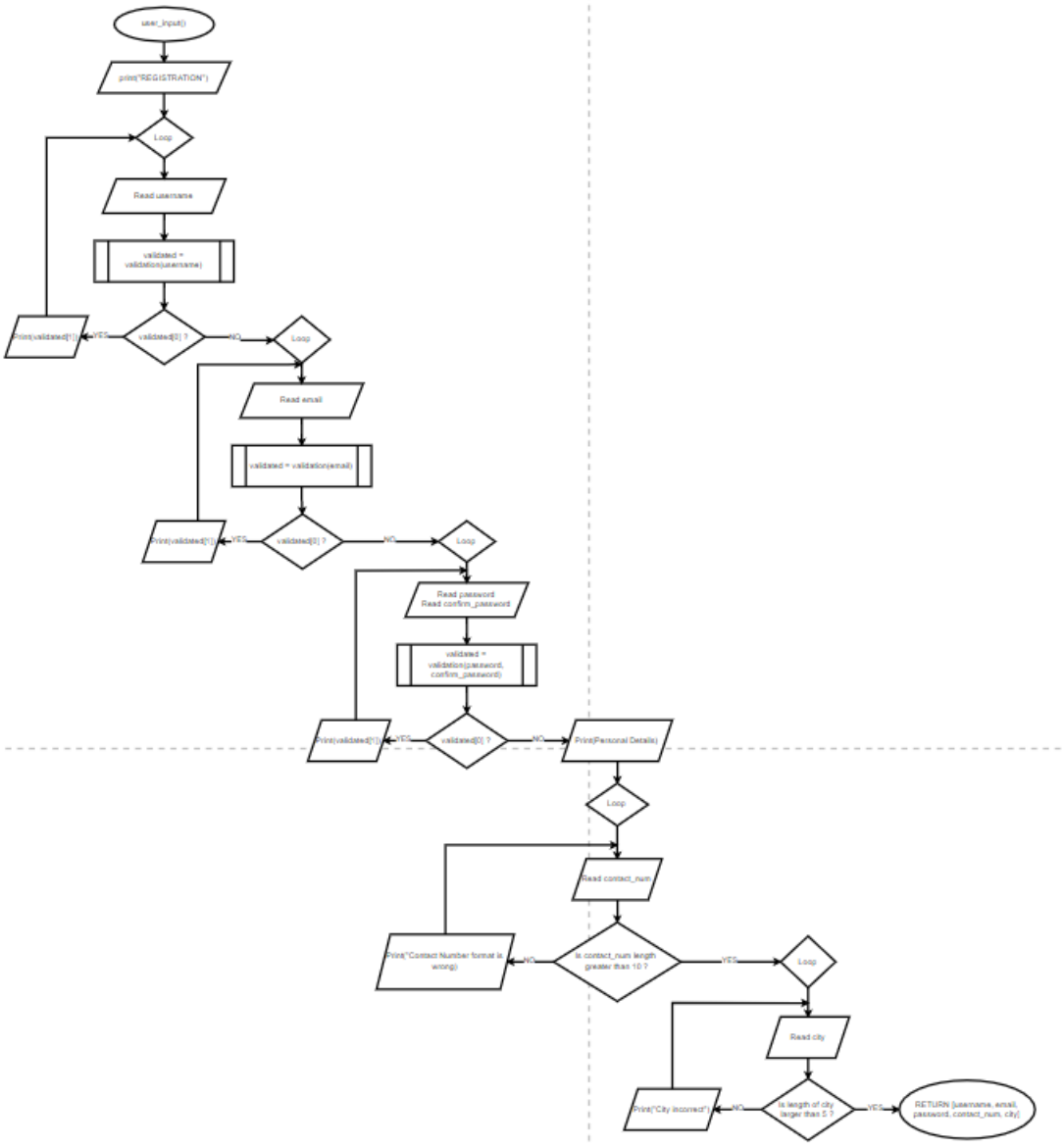
Flowchart 17 rental_expire



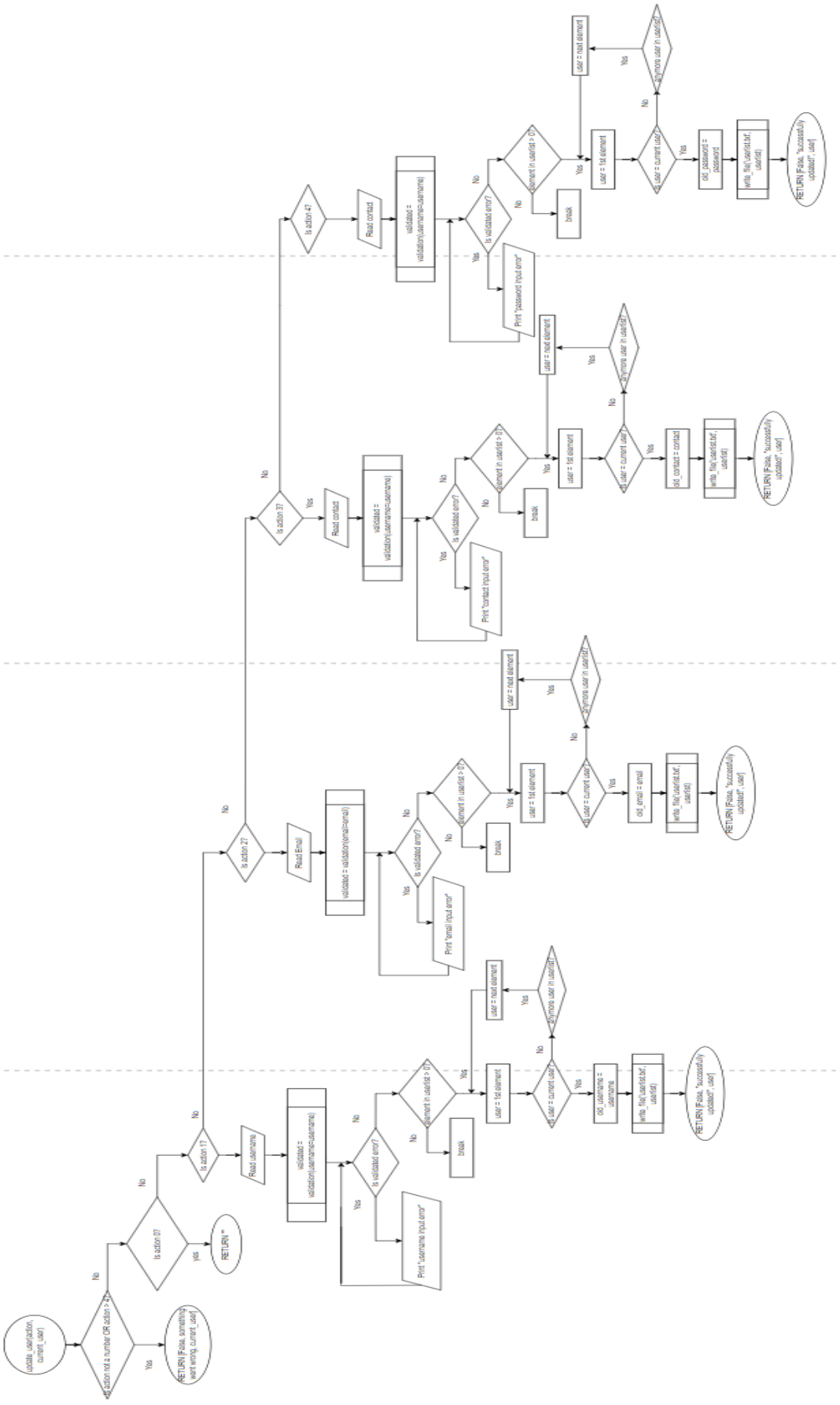
Flowchart 18 rented_out



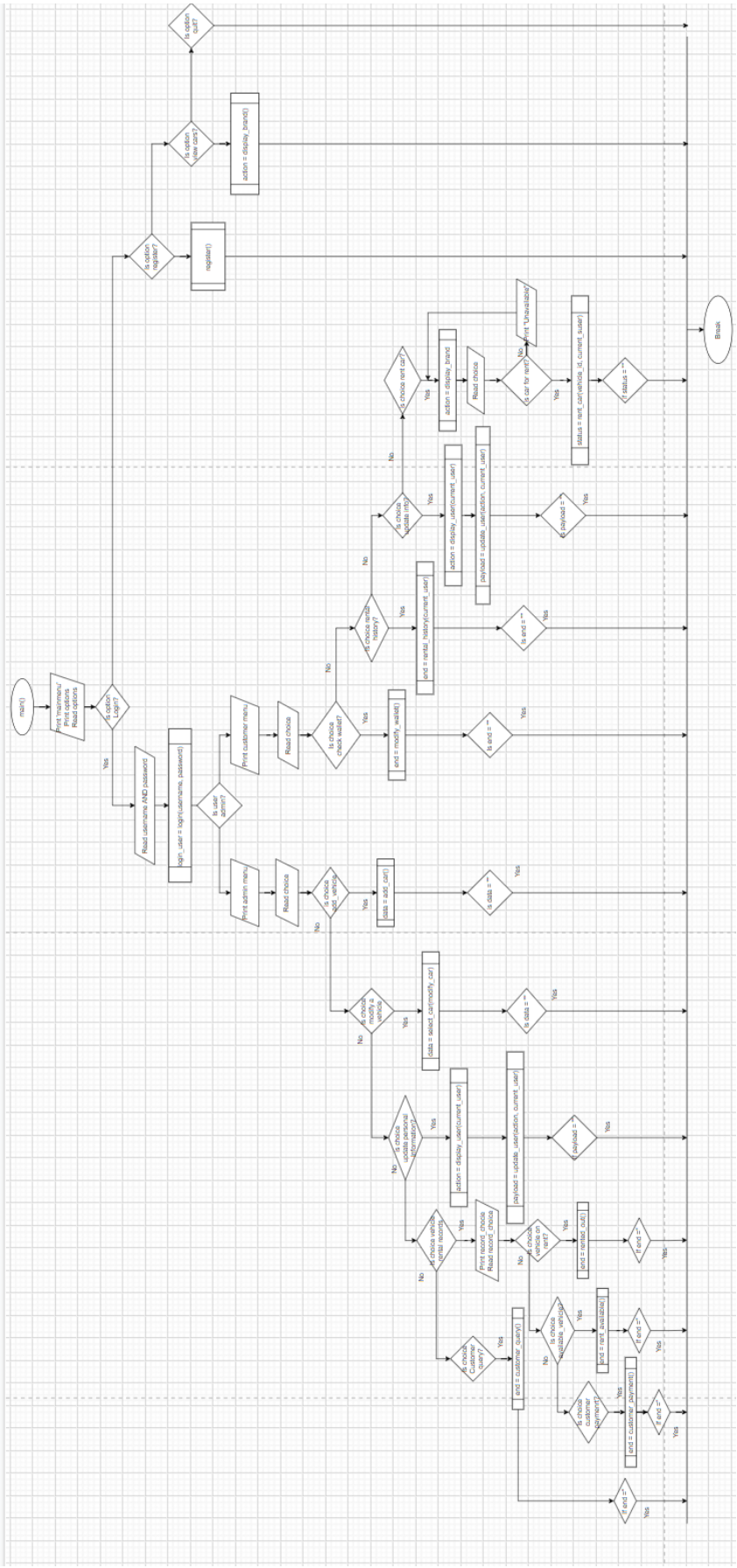
Flowchart 19 rental_history



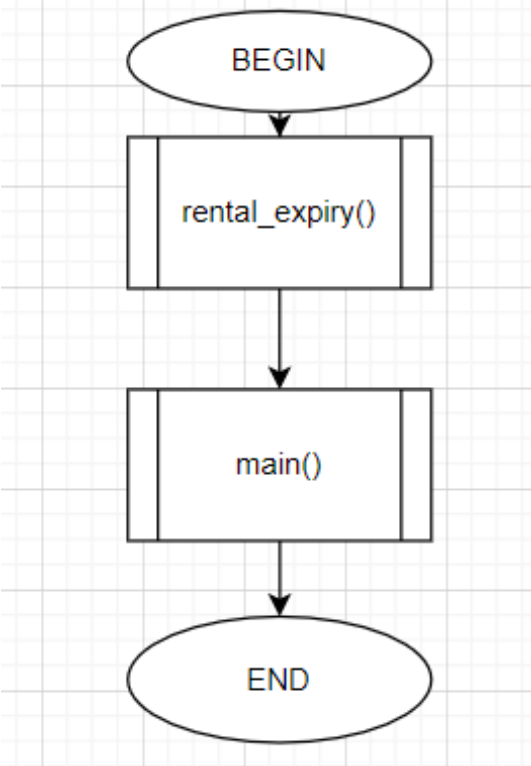
Flowchart 20 user_input



Flowchart 21 update_user



Flowchart 22 main



Flowchart 23 program_run

4.0 Pseudocode

*Remark: Pseudocode is done with 1.15 spacing to reduce eye strain

```
### READ FILE ###
FUNCTION read_file(filename)
  OPENFILE ‘filename’ for READ
  RETURN
  CLOSEFILE
ENDFUNCTION
### WRITE FILE ###
FUNCTION write_file(filename,content)
  OPENFILE ‘filename’ for WRITE
ENDFUNCTION

### VALIDATION ###

FUNCTION validation
  userlist = call read_file(‘userlist.txt’)
----PASSWORD LENGTH---
  IF (length of password < 5) THEN
    RETURN [True, “Password length must be greater than 5.”]
  ENDIF
  -----SAME PASSWORD---
  IF (password != confirm_password) THEN
    RETURN [True, “Password entered is incorrect, please try again.”]
  ENDIF
-----USERNAME LENGTH---
  IF (length of username < 5) THEN
    RETURN [TRUE, “Username must be greater than 5.”]
  ENDIF

  FOR EACH user IN userlist
    IF (username entered = user) THEN
      RETURN [True, 'Username taken,please try again.']
    ENDIF
  ENDFOR
  -----EMAIL FORMAT---
  REGEX = re.compile(r'[^@]+@[^@]+\.[^@]+')
  IF (not REGEX(email)) THEN
    RETURN [True, 'Email format is incorrect', please try again.']
  ENDIF

  RETURN False
ENDFUNCTION
```

```
###USER INPUT###
FUNCTION user_input
  PRINT 'REGISTRATION'
  PRINT '------------'

  ---USERNAME---
  DOWHILE True
    Read username
    validated_info = call validation(username)
    IF (validated_info[0]) THEN
      Print (validated_info[1])
      continue
    ELSE
      break
    ENDIF
  ENDDO

  ---EMAIL---
  DOWHILE True:
    Read email
    validated_info = call validation(email)
    IF (validated_info[0]) THEN
      PRINT (validated_info[1])
      continue
    ELSE
      break
    ENDIF
  ENDDO

    ---PASSWORD---
  DOWHILE True
    Read password
    Read confirm_password
    validated_info = call validation(password, confirm_password)

    IF (validated_info[0]) THEN
      Print validated_info[1]
      continue
    ELSE
      break
    ENDIF
  ENDDO
```

```
  ---contact---
  DOWHILE True
    Read contact number
    IF (contact is not numeric) THEN
      Print “Contact number must only contain numbers”
      continue
    ELSE
      break
    ENDIF
  ENDDO

  ---city---
  DOWHILE True
    Read state
    IF (Length of state < 4) THEN
      Print “State not found, please try again”
      continue
    ELSE
      break
    ENDIF
  ENDDO

  ---deposit---
  DOWHILE True
    Print “Would you like to make an initial deposit?”
    Print “<Enter> to skip the deposit”
    Read deposit amount
    IF wallet = “” THEN
      wallet = 0
      break
    ELSE
      IF wallet not numeric THEN
        Print 'Invalid input'
      ENDIF
    ELSE
      break
    ENDIF
  RETURN [username, email, password, "0" + contact, city, int(wallet), [], ""]
  ENDDO
ENDFUNCTION

###Field control###
```

```
FUNCTION field_control(field_text, type, wildcard="")
  DOWHILE True
    Read field_input
    IF (field_input = "") THEN
      RETURN wildcard
    ENDIF

    IF type = 0 THEN
      IF (length of field_input < 2) THEN
        Print "Text Unknown, please try again"
        continue
      ELSE
        Break
      ENDIF
    ENDIF

    IF (type = 1) THEN
      IF (field_input = "" or not numeric) THEN
        Print "Please fill in with numbers only..."
        continue
      ELSE
        break
      ENDIF
    ENDIF
  RETURN field_input
  ENDDO
ENDFUCNTION


### RENTAL_EXPIRE###

FUNCTION rental_expire
  carlist = call read_file(carlist.txt)
  FOR EACH i IN (length(carlist)-1)
  car = carlist[i]
    IF car[-2] THEN
      IF rented out date < current date THEN
        car[-1] = False
        car[-2] = False
        call write_file("carlist.txt", carlist)
        RETURN
      ENDIF
    ENDIF
  ENDFOR
ENDFUNCTION
```

```
###USER Functions###
FUNCTION register
  userlist = call read_file(“userlist.txt”)
  user_detail = call user_input()
  append userlist(user detail)
  call write_file('userlist.txt',userlist)
  IF user_detail[3] != 0 THEN
    float_price = user_detail[3]
    Print “Total amount in your account”
  Print “You have registered successfully, please login now.”
  ENDIF
ENDFUNCTION

### Login Function ###
FUNCTION login(username, password)
  userlist = call read_file(userlist.txt)
  err = True
  FOR EACH user IN userlist
    IF user[0] = username THEN
      IF user[2] = password THEN
        err = False
        Print “You've logged in successfully”
        RETURN user
      ENDIF
    ENDIF

    IF err THEN
      Print “Username or password incorrect, please try again.”
      RETURN “”
    ENDIF
  ENDFOR
ENDFUNCTION
###DISPLAY USER###
FUNCTION display_user(current_user)
  username = current_user[0][0]
  email = current_user[0][1]
  contact = '+6' + current_user[0][3]

  Print “Update Personal Information”
  Print "1. Username: [{username}]\n2. Email: [{email}]\n3. Contact Number: [{contact}]\n4.
Password\n\n0. Go Back\n"
  detail = Read 'Which detail do u wish to update'
  RETURN detail
ENDFUNCTION
```

```
###UPDATE USER###

FUNCTION update_user(action, current_user)
  IF action is not numeric or action > 4 THEN
    RETURN [False, “Something went wrong”, current_user[0]]
  ENDIF
  ---back---
  IF action = '0' THEN
    RETURN ''
  ENDIF
  userlist = call read_file('userlist.txt')

   --- updates username ---
  DOWHILE action = 1
    Read new username
    validated = call validation(username=username)
    IF validated[0] THEN
      Print validated[1]
    ENDIF

    IF not validated[0] THEN
      FOR EACH user in userlist
        IF user[0] = current_user[0][0] THEN
          user[0] = username
          break
        ENDIF
      ENDFOR
    ENDIF
    call write_file('userlist.txt', userlist)
    RETURN [False, "User info has been successfully updated!", user]
  ENDDO

  --- updates email ---
  DOWHILE action = 2
    Read new_email
    validated = call validation(email=email)

    IF validated[0] THEN
      Print validated[1]
    ENDIF

    IF NOT validated[0] THEN
      FOR EACH USER IN userlist
        IF user in list = current_user[0][0] THEN
```

```
        user[1] = new_email
        break
       ENDIF
     ENDFOR
     Call write_file('userlist.txt', userlist)
     RETURN [False, 'User info has been successfully updated!', user]
   ENDIF
  ENDDO

  --- update contact ---
  DOWHILE action = 3
   Read new_contact

   IF contact is not numeric THEN
    Print 'Please insert correct information'
    continue
   ENDIF

   FOR EACH user in userlist
    IF user[0] = current_user[0][0] THEN
     user[3] = new_contact
     break
    ENDIF
   call write_file('userlist.txt', userlist)
   RETURN [False, 'User info has been successfuly updated!', user]
   ENDFOR
  ENDDO

  ---update password---
  DOWHILE action = 4
   error = False
   Read old_password
   Print new line
   Read new_password
   Read confirmed_new_password
   Validated = validation(password = new_password, confirm_password =
confirmed_new_password)
   IF validated[0] THEN
    Print validated[2]
    continue
   ENDIF

   FOR EACH user IN userlist
    IF user[0] = current_user[0][0] THEN
```

```
      IF user[2] != password(old)THEN
        err = True
        Print “Old password incorrect”
        Print “1. Retry”
        Print “0. Quit”
        Read choice
        IF choice = "0" THEN
          RETURN [True, “Please try again later...”]
        ENDIF
        IF choice EQUALS "1" THEN
          continue
        ENDIF
      ENDIF
      IF not validated[0] and not False THEN
        user[2] = password(new_password)
        break
      ENDIF
    ENDIF
  ENDFOR
  Call write_file('userlist.txt', userlist)
  RETURN [True, “User info has been successfully updated, please login again”]
 ENDDO
ENDFUNCTION

###Modify Wallet###
FUNCTION modify_wallet(current_user)
 balance = current_user[0][-3]
 Print “Total remaining balance: ”
 Print “1. Add fund, <Enter> to quit”
 Read choice
 DOWHILE True
  IF choice != 1 THEN
   RETURN 0
  ENDIF
  IF add_fund = 1 THEN
    Read amount_to_deposit
    userlist = call read_file(“userlist.txt”)
    FOR EACH user IN userlist
      IF user[0] = current_user[0][0] THEN
        user[5] = user[5] + amount_to_deposit
        updated_user = user
        break
      ENDIF
    ENDFOR
```

```
   ENDIF
  Call write_file("userlist.txt", userlist)
  SET current_user[0] TO updated_user
  Print "Total fund of RM{amount_to_deposit} has been deposited"
  Read "<Enter> to RETURN..."
  break
  ENDDO
ENDFUNCTION


### RENT CAR###
FUNCTION rent_car(id, current_user)
  carlist = call read_file('carlist.txt')
  userlist = call read_file('userlist.txt')
  FOR EACH car IN carlist
   IF CAR[0] = id THEN
    IF CAR[-2] THEN
     RETURN [True, 'Car is already rented out']
    ENDIF
    brand = car[2]
    model = car[3]
    year = car[4]
    price = car[8]
    Print “You have selected {brand} {model}, {year}”
    Print “Rental price FOR this product will be fixed at the rate of RM{price} per day”
    Read confirmation_input
    IF confirmation_input = 'no' THEN
     RETURN
    ENDIF
    Read duration
    DOWHILE confirmation = “yes”
     total_price = price * duration
     FOR EACH user IN userlist
      IF username = current_user[0][0] THEN
       IF wallet < total_price THEN
        RETURN [True, “Insufficient balance, you are broke!”]
       ENDIF
       username = current_user[0][0]
       car[-2] = True
       car[-1] = [username, duration, datetime.datetime.now(), datetime.datetime.now() +
timedelta(days=int(duration))]

       append car into user[6]
       user[5] -= total_price
```

```
        call write_file('carlist.txt', carlist)
        call write_file('userlist.txt', userlist)
        current_user[0] = user
        Print “Total payment made RM{total_price}”
        Print “Your booking order FOR {brand} {model}, {year} FOR the duration of {duration} days has been confirmed\nEnjoy your ride!”

        Read end_input
        RETURN end_input
        ENDIF
       ENDIF
     ENDFOR
   break
   ENDDO
ENDFUNCTION
### DISPLAY BRAND###
FUNCTION display_brand()
  cars = call read_file('carlist.txt')
  brand = []
  FOR EACH car in cars
    Append (car[2]) to brand
  ENDFOR
  DOWHILE True
    count = 1
    FOR EACH i in brand
      Print '{count}. {i}'
      count = count + 1
    ENDFOR
    Print “0. Go back to home page.”
    Read mocel_choice
    IF model_choice is numeric THEN
      break
    ENDIF
  RETURN [num, brand]
  ENDDO
ENDFUNCTION

###CAR DETAILS###
FUNCTION car_details(brand, default=True)
  cars = call read_file('carlist.txt')
  car_model = []
  IF default THEN
    FOR EACH car IN cars
```

```
    IF car[2] = brand THEN
      append car to car_model
    ENDIF
  ENDFOR
ENDIF
IF NOT default THEN
  FOR EACH car IN cars
    IF car[-2] == False THEN
      append(car) to car_model
    ENDIF
  ENDFOR
ENDIF
FOR EACH i IN car_model
  id = i[0]
  num_plate = i[1]
  brand = i[2]
  model = i[3]
  year = i[4]
  owner = i[5]
  condition = i[6]
  desc = i[7]
  price_rate = i[8]
  seats = i[9]
  availability = i[10]
  rent_by = i[11]
  float_price_rate = float(price_rate)
ENDFOR

Print "-" * 25
Print "ID: {id}"
Print "Number Plate: {num_plate}"
Print "Vehicle: {brand} {model}, {year}"
Print "Total seats: {seats}"
Print "Short Description: {desc}"
Print "Owned by {owner}"
Print "Condition: {condition}/10"
Print "Price rate: RM{float_price_rate}/day"
IF availability THEN
  Print "availability: No"
ENDIF
IF NOT availability THEN
  Print "availability: Yes"
ENDIF
IF rent_by THEN
```

```
    IF rent_by[0] THEN
      username = rent_by[0]
      start_date = rent_by[2]
      duration = rent_by[1]
      str_date = start_date[0:11]
      Print "currently rented by {username}\nRented since {str_date} for {duration} days"
    ENDIF
    Print "-" * 25
    Print newline
  ENDIF

  IF (length of car_model) == 0 THEN
    Print "Oops, nothing is here yet"
  ENDIF
ENDFUNCTION

### ADD CAR ###
FUNCTION add_car()
  Print “-“ * 20
  Print “SCRS Vehicle Management”
  Print “-” * 20
  num_plate = call field_control('Number Plate: ', 0)
  brand = call field_control('Vehicle Brand',0)
  model = call field_control('Vehicle Model: ', 0)
  year = call field_control('Manufactured Year: ', 1)
  owner = call field_control('Owner of the vehicle: ', 0)
  condition = call field_control('Condition of the car [?]/10: ', 1)
  desc = call field_control('Short description: ', 0)
  price_rate = call field_control('Price rate per day: RM', 1)
  seats = call field_control('Number of seats: ', 1)
  carlist = call read_file('carlist.txt')
  latest_id = 0

  FOR EACH car IN carlist
    IF car[0] > latest_id THEN
      Latest_id = car[0] + 1
  ENDFOR
  new_car = [latest_id, num_plate, brand, model, year, owner, condition, desc, price_rate, seats,
False, False]

  append new_car to carlist
  call write_file('carlist.txt', carlist)
  Read ('Car has been successfully added to the system... <Enter> to RETURN:')
  RETURN detail
```

```
ENDFUNCTION

FUNCTION modify_car(id)
  Print “Car model: ”
  carlist = call read_file('carlist.txt')
  FOR EACH car IN carlist
    IF car[0] == id
      Print “Modify details of ”, car[2], car[3], ',', car[4]
      Print “Current number plate: ”, car[1]
      Print “<Enter> to keep previous data...”

      num_plate = call field_control('Number Plate [" + car[1] + "]: ', 0, car[1])
      brand = call field_control('Vehicle Brand [" + car[2] + "]: ', 0, car[2])
      model = call field_control('Vehicle Model [" + car[3] + "]: ', 0, car[3])
      year = call field_control('Manufactured Year [" + str(car[4]) + "]: ', 1, car[4])
      owner = call field_control('Owner of the vehicle [" + car[5] + "]: ', 0, car[5])
      condition = call field_control('Condition of the car [" + str(car[6]) + "/10)" + ": ', 1, car[6])
      desc = call field_control('Short description: [" + car[7] + "]\n: ', 0, car[7])
      price_rate = call field_control('Price rate per day: [RM" + "{:.2f}".format(car[8]) + "]: ', 1,
car[8])
      seats = call field_control('Number of seats [" + str(car[9]) + "]: ', 1, car[9])
      new_car = [car[0], num_plate, brand, model, year, owner, condition, desc, price_rate, seats,
car[10], car[11]]

      break
    ENDIF
  Append new_car to carlist
  call write_file('carlist.txt;,carlist)
  RETURN [True, “Car details have been modified successfully.”]
  ENDFOR
ENDFUNCTION

###SELECT CAR###
FUNCTION select_car(callback)
  Print “-” * 20
  Print “SCRS Vehicle Management”
  Print “-” * 20
  action = call display_brand()
  IF action[0] == '0' THEN
    RETURN “”
  ENDIF
  DOWHILE action[0] != '0'
    payload = action[0] -1
    call car_details(brand=action[1][payload])
```

```
    Read vehicle_id
    DOWHILE length of vehicle_id > 0
      status = callback(vehicle_id)
      IF status[0] THEN
        Print status[1]
        Read “<Enter> to RETURN back to main menu...”
        break
      ENDIF
    ENDDO
    IF vehicle_id == “” THEN
      break
    ENDIF
  ENDDO
ENDFUNCTION
###RENTAL HISTORY###
FUNCTION rental_history(current_user)
  userlist = call read_file('userlist.txt')
  FOR EACH user IN userlist
    IF user[0] == current_user[0][0] THEN
      IF length of user[-2] == 0 THEN
        Print “Start placing order today for exclusive discounts!”
        RETURN Read "<Enter> to return back to home page..."
      ENDIF
      FOR EACH rent IN user[-2]
        brand = rent[2]
        model = rent[3]
        year = rent[4]
        price_rate = rent[8]
        start_date = rent[-1][2]
        end_date = rent[-1][3]
        duration = rent[-1][1]
        str_date = start_date[0:11]
        str_enddate = end_date[0:11]

        total_price = price*duration

        Print “-” * 20
        Print “Booked on {str_date} for the duration of {duration} days”
        Print “Ends by {str_enddate}”
        Print “Vehicle: {brand} {model}, {year}”
        Print “Total price deducted from wallet: -RM{total_price}”
        Print “-” * 20
      break
      ENDFOR
```

```
   ENDIF
  ENDFOR
  end = Read “<Enter> to return back to home page...”
  RETURN end
ENDFUNCTION

###RENTED OUT###
FUNCTION rented_out()
  Print '-' * 20
  Print 'CARS ON TRANSIT RECORDS'
  Print '-' * 25

  carlist =call read_file('carlist.txt')

  FOR EACH car IN carlist
    IF car[-2] THEN
      IF length of car[-1] > 0 THEN
        booking_details = car[-1]
        num_plate = car[1]
        brand = car[2]
        model = car[3]
        year = car[4]
        owner = car[5]
        condition = car[6]
        desc = car[7]
        price_rate = car[8]
        start_date = booking_details[2]
        str_date = start_date[0:11]
        end_date = booking_details[3]
        str_enddate = end_date[0:11]
        username = booking_details[0]
        duration = booking_details[1]

        total_price = duration * price_rate

        Print “-” * 20
        Print “Booked on {str_date} for the duration of {duration} days”
        Print “Ends by {str_enddate}”
        Print “Vehicle: {brand} {model}, {year}”
        Print “Plate number: {num_plate}, owned by {owner}”
        Print “Condition: {condition}/10”
        Print “Description: {desc}”
        Print “Total price deducted from wallet: -RM{total_price}”
        Print “Rented by {username}”
```

```
      Print “-” * 20
    ENDIF
  ENDIF
 ENDFOR
 end = Read “<Enter> to go back...”
 RETURN end
ENDFUNCTION

###AVAILABLE RENTS###
FUNCTION rent_available()
 Print “-” * 20
 Print “SCRS Vehicle Management”
 Print “-” * 20
 Print “Cars available for rent: ”
 action = call display_brand()
 IF action[0] == '0'
  RETURN ''
 ENDIF
 DOWHILE action[0] != '0'
  payload = action[0] -1
  call car_details(action[1][payload], False)
  Read "Press <Enter> to quit: "
  break
 ENDDO
ENDFUNCTION

###CUSTOMER PAYMENT###
FUNCTION customer_payment()
 Print “-” * 20
 Print “SCRS Customer Order Record”
 Print “-” * 20
 userlist = call read_file('userlist.txt')
 FOR EACH user IN userlist
  IF length of user[-2] > 0 THEN
   username = user[0]
   email = user[1]
   total_spent = 0
   Print “-” * 20
   Print “Username: {username}”
   Print “Email: {email}”
   Print “-” * 15
   FOR EACH data IN user[-2]
    start_date = data[-1][2][0:11]
    end_date = data[-1][3][0:11]
```

```
      duration = data[-1][1]
      vehicle = '{data[2]} {data[3]}, {data[4]}'
      price_per_order = data[8] * duration
      Print “{vehicle}”
      Print “Order booked on {start_date}for {duration} days”
      Print “Total spent: RM{price_per_order}”
      Print “Expire on {end_date}”
      total_spent += price_per_order
    ENDFOR
    str_spent = total_spent
    Print “Total amount earned: {str_spent}”
  ENDIF
 end = Read “<Enter> to go back...”
 RETURN end
 ENDFOR
ENDFUNCTION

### CUSTOMER QUERY###
FUNCTION customer_query()
 userlist = call read_file('userlist.txt')
 username = []
 FOR EACH user IN userlist
  Append user[0] into usernames
 ENDFOR
 Print “Search by names: ”
 count = 0
 FOR EACH names IN  usernames
  Print “{count}. {names}”
  count += 1
 ENDFOR
 Print “<Enter> to return”
 Read selected_username
 IF selected_username == “” THEN
  RETURN selected_username
 ENDIF

---error handling for invalid index---
 IF selected_username is numeric THEN
  IF selected_username >= lists of usernames OR selected_username < 0 THEN
   Print “Username does not exist...”
   end = Read '<Enter> to return'
   RETURN end
  ENDIF
 ENDIF
```

```
---error handling for non-numeric---
 IF selected_username NOT numeric AND NOT selected_username IN usernames THEN
  Print “Username does not exist...”
   end = Read '<Enter> to return'
   RETURN end
 ENDIF
 DOWHILE LOOP
  IF selected_username is numeric THEN
   FOR EACH user IN userlist
    IF user[0] == usernames[selected_username] THEN
     username = user[0]
     email = user[1]
     Print “-” * 20
     Print “Username: {username}”
     Print “Email: {email}”
     Print “-” * 15
     IF length of user[2] > 0 THEN
      FOR EACH record IN user[-2]
       start_date = record[-1][2][0:11]
       end_date = record[-1][3][0:11]
       duration = record[-1][1]
       vehicle = {record[2]} {record[3]}, {record[4]}
       num_plate = record[-1][0]
       price_per_order = record[8] * duration
       Print “Number plate: {num_plate}”
       Print “{vehicle}”
       Print “Order booked on {start_date}for {duration} days”
       Print “Total spent: RM{price_per_order}”
       Print “Expire on {end_date}”
      ENDFOR
     ENDIF
    ENDIF
   ENDFOR
  ENDIF
  IF length of selected_username > 1 THEN
   FOR EACH user IN userlist
    IF user[0] == selected_username THEN
     username = user[0]
     email = user[1]
     Print “-” * 15
     Print “Username: {username}”
     Print “Email: {email}”
     Print “-” * 15
     IF length of user[-2] > 0 THEN
```

```
          FOR EACH record IN user[-2]
            start_date = record[-1][2][0:11]
            end_date = record[-1][3][0:11]
            duration = record[-1][1]
            vehicle = '{record[2]} {record[3]}, {record[4]}'
            num_plate = record[-1][0]
            price_per_order = record[8] * duration
            Print “Number plate: {num_plate}”
            Print “{vehicle}”
            Print “Order booked on {start_date}for {duration} days”
            Print 'Total spent: RM{price_per_order}'
            Print “Expire on {end_date}”
          ENDFOR
        ENDIF
      ENDIF
    ENDFOR
  ENDIF
  end = Read '<Enter> to return'
  RETURN end
 ENDDO
ENDFUNCTION

###MAIN FUNCTION###
FUNCTION main()
 current_user = []
 Print “-” * 20
 Print “Super Car Rental Service (SCRS)”
 Print “-” * 20

 ---Main Menu---
 DOWHILE length of current_user == 0
  Print “1. Login 2. Register 3. View Cars 0. Quit”
  Read Option
  FOR EACH i in option

 ---view car as guests---
    IF i == 3 THEN
      Print “Car List: ”
      action = call display_brand()
      IF action[0] == '0'
        break
      ENDIF
      DOWHILE action[0] != '0'
        payload = action[0] - 1
```

```
        call car_details(brand=action[1][payload])
        Read 'Press <Enter> to quit: '
        break
      ENDDO
    ELSE
      IF i == '2' THEN
        call register()
      ENDIF
    ELSE
      IF i == '1' THEN
        Print 'LOGIN'
        Read username
        Read password
        login_user = call login(username, password)
        IF login_user == '' THEN
          call main()
        Append login_user to current_user
        ENDIF
      ENDIF
    ELSE
      IF i == '0' THEN
        break
      ELSE
        Print “Invalid input, please enter value accorrding to menu...”
      ENDIF
  ENDFOR
 ENDDO

 -#- admin login -#-
 DOWHILE length of current_user > 0 AND current_user[0][-1] == 'admin'
  Print “Welcome current_user[0][0]”
  Print “Admin Menu(1. Add a vehicle 2. Modify vehicle Detail 3. Update Personal Information 4. Vehicle Rental Records 5. Query Customer Record 0. Logout)”
  Read admin_option

 ---customer query---
  DOWHILE admin_option == '5'
    Print “-” * 20
    Print “SCRS Customer Records Management”
    Print “-” * 20,
    end =call customer_query()
    IF end == ''
      break
    ENDIF
```

```
   ENDDO

  --- rental records---
   DOWHILE admin_option == '4'
    Print “SCRS Vehicle Management”
    Print “1. Vehicles in transit 2. Vehicles available for Rent 3. Customer Payments for a specific time duration 0.Back”
    Read record_option
  ## returns ##
    IF record_option == '0' THEN
     break
    ENDIF
     ----cars booked ---
    DOWHILE record_option == '1'
     end = call rented_out()
     IF end == '' THEN
      break
     ENDIF
    ENDDO
    DOWHILE record_option == '3' THEN
     end = call customer_payment()
     IF end == '' THEN
      break
     ENDIF
    ENDDO
   ENDDO

  ---update peronal info---
   DOWHILE admin_option == '3'
    action = call display_user(current_user)
    payload = call update_user(action, current_user)
    IF payload == '' THEN
     break
    ENDIF
    IF payload[0] THEN
     Print “payload[1]”
     Read '<Enter> to continue'
     current_user[0] = []
     call main()
    ENDIF
    IF NOT payload[0] THEN
     Print “payload[1]”
     current_user[0] = payload[2]
     Read choice
```

```
      break
    ENDIF
  ENDDO

 --- MODIFY VEHICLE---
  DOWHILE admin_option == '2'
    data = call select_car(modify_car)
    IF data == '' THEN
      break
    ENDIF
  ENDDO

 --- ADD VEHICLE---
  DOWHILE admin_option == '1'
     data = call add_car()
     IF data == '' THEN
       break
     ENDIF
  ENDDO
 --- QUIT ---
  IF admin_option == '0' THEN
    clear current_user
    call main()
    break
  ENDIF
 ENDDO

 -#- CUSTOMER INTERFACE -#-
 DOWHILE length of current_user > 0 AND current_user != 'admin'
  Print “Welcome, current_user[0][0]”
  Print “1. Rent a Car 2. Update Personal Information 3. Rental History 4. Check Wallet 0. Logout”
  Read user_option
 --- CHECK WALLET ---
  DOWHILE user_option == '4'
    end = call modify_wallet(current_user)
    IF end == 0 THEN
      break
    ENDIF
  ENDDO
 --- RENTAL HISTORY ---
  DOWHILE user_option == '3'
    end = call rental_history(current_user)
    IF end == '' THEN
```

```
      break
    ENDIF
  ENDDO
--- UPDATE PERSONAL INFO ---
  DOWHILE user_option == '2'
    action = call display_user(current_user)
    payload = call update_user(adtion, current_user)
    IF payload == '' THEN
      break
    ENDIF
    IF payload[0] THEN
      Print “payload[1]”
      current_user[0] = payload[2]
      choice = Read "<Enter> to continue..."
      break
    ENDIF
    IF choice THEN
      break
    ENDIF
  ENDDO

--- RENT CAR ---
  DOWHILE user_option == '1'
    action = call display_brand()
    IF action[0] == '0'
      break
    ENDIF
    DOWHILE action[0] != '0'
      payload = action[0] -1
      call car_details(brand=action[1][payload])
      Read vehicle_id
      DOWHILE length of vehicle_id > 0
        status = call rent_car(vehicle_id, current_user)
        IF status == '' THEN
          break
        ENDIF
        IF status[0] THEN
          Print “status[1]”
          retry = Read 'Please select other car available for rent. <Enter> to continue'
          IF retry == '' THEN
            break
          ENDIF
        ENDIF
      ENDDO
```

```
        IF vehicle_id == '' THEN
          break
        ENDIF
      ENDDO
    ENDDO

  --- Log Out ---
  IF user_option == '0' THEN
    clear current_user
    call main()
    break
  ENDIF
ENDFUNCTION


PROGRAM SCRS_ETWX
BEGIN
  call rental_expire()
  call main()
END
```

5.0 Source code

5.1 Utilities Functions

5.1.1 Read file

```
def read_file(filename):
    # ------------------------
    # read txt files
    # ------------------------
    try:
        with open(filename) as f:
            data = f.read()
        return json.loads(data)
    except:
        return []
```

Figure 1 Read file function source code

By importing json module into the program, it allows us to "stringify" JSON data into python readable list format. By calling the function, it must insert the filename extension parameter to open the selected file and read the JSON formatted data within it. Try except block are used as error handling to prevent fetching unidentified file, if the file inserted as parameter is unidentified, it will return an empty list to escape further errors.

5.1.2 Write file

```
def write_file(filename, content):
    # ------------------------
    # write txt files
    # ------------------------
    with open(filename, "w") as f:
        f.write(json.dumps(content, indent=2, sort_keys=True, default=str))
        f.close()
    return
```

Figure 2 Write file function source code

To write a python formatted list data types into the selected file, it requires two parameter including filename and data to be inserted into file. By calling the function, it will parse content into JSON formatted data types followed by a few arguments such as indentation, sorting to insert data in a readable manner for backend engineers. Default arguments are passed in to convert any unknown JSON data types into string format.

5.1.3 Rental expiration

```
def rental_expire():
  # ------------------------
  # reset car availability status when expired
  # ------------------------
  carlist = read_file("carlist.txt")

  for i in range(len(carlist) - 1):
    car = carlist[i]
    if car[-2]:
      if datetime.datetime.strptime(car[-1][3], "%Y-%m-%d %H:%M:%S.%f") < datetime.datetime.now():
        car[-1] = False
        car[-2] = False
        write_file("carlist.txt", carlist)
        return
```

This function will automate the process of returning car from currently rented customer when the rental duration ends. When the carlist has been retrieved from the text file, it will start looping through the cars that are currently rented by checking on the rental status. If the rental status is True, it will start comparing the expiration date with the current date, if the current date has passed the expiration rental date, it will automatically switch the rental details into False until every car is validated. It will rewrite and update the new carlist.

5.2 User Functions

5.2.1 User Register

```
def register():
  # -------------------------
  # Register
  # access: anyone
  # -------------------------
  clear()
  userlist = read_file("./list_solution/userlist.txt")

  user_detail = user_input()
  userlist.append(user_detail)

  write_file("./list_solution/userlist.txt", userlist)

  clear()
  if user_detail[-3] != 0:
    float_price = "{:.2f}".format(user_detail[-3])
    print(f"Total amount of RM{float_price} deposited into your account")
  print("You have registered successfully, please login now...")
```

Figure 3 User registration function source code

```
def user_input():
  # -------------------------
  # user info
  # extend from register
  # -------------------------
  clear()
  print("REGISTRATION")
  print("------------")

  # username
  while True:
    username = input("Username: ")

    validated_info = validation(username=username)

    if validated_info[0]:

      print(validated_info[1])
      continue
    else:
      break

  # email
  while True:
    email = input("Email: ")

    validated_info = validation(email=email)

    if validated_info[0]:

      print(validated_info[1])
      continue
    else:
      break
```

Figure 4 User registration inputs function source code

By calling the register function, it will retrieve the user data from user list data file. Once the user list has been retrieved successfully, it will call the user input function to prompt

user with account registration process, while loop is utilized within the function to ensure input format are accurate for data validation and redundancy, including the uniqueness of the username, minimum password length, minimum username length, format of the email, format of the contact number in Malaysia and so on. Once the registration inputs are validated, it will return the user details in a list holds by validated info to be written into the existing user list file

5.2.2 User Login

```
def login(username, password):
  # -------------------------
  # Login
  # access: anyone
  # -------------------------
  userlist = read_file("./list_solution/userlist.txt")

  err = True
  for user in userlist:
    if user[0] == username:
      if user[2] == hash_password(password):
        err = False
        clear()
        print("You have login successfully")
        return user

  if err:
    clear()
    input("Username or password is incorrect, please try again...\n\n <Enter> to return back to main menu...")
    return ""
```

Figure 5 User login function source code

Before calling the function, user is prompted to input username and password for validation. By inserting username and password parameter into login function, it will retrieve the user data from user list data file and hold by a variable named “userlist”. A series of looping is running to match the username with the database, if it existed, it would then proceed matching with the encrypted password, and proceed by returning user’s list if validated. Or else, it will notify user’s username or password are incorrect, please try again.

5.2.3 Modify user details (User/Admin)

```python
def update_user(action, current_user):      You, 2 weeks ago • -complete: version 0.0
  # ------------------------
  # Update user information
  # access: logged in users
  # ------------------------
  if not action.isnumeric() or action > "4":
    return [False, "something went wrong", current_user[0]]

  if action == "0":
    return ""

  userlist = read_file("./list_solution/userlist.txt")

  # update username
  while action == "1":
    username = input("Enter new username: ")
    validated = validation(username=username)

    if validated[0]:
      clear()
      print(validated[1])

    if not validated[0]:
      for user in userlist:
        if user[0] == current_user[0][0]:
          user[0] = username
          write_file("./list_solution/userlist.txt", userlist)
          return [False, "User info has been successfully updated!", user]
```

Figure 6 Modify user's username function source code

Update user function requires two parameters, including action, as the option to select fields to modify and "current_user", as logged in user list. In figure 6, if the user selected action 1, it will prompt the user to modify the username, if validated, it will loop through the userlist file and modify the current user username entirely, and return a list with two elements, including error indicator which is False and messages to print onto the user interface. The same process would repeat if the action were selected other than 4.

```python
# update password
while action == "4":
  err = False
  clear()
  old = input("Enter old password: ")
  new_password = input("\nEnter new password: ")
  new_confirm = input("Confirm new password: ")

  validated = validation(password=new_password, confirm_password=new_confirm)

  if validated[0]:
    clear()
    print(validated[1])
    continue

  for user in userlist:
    if user[0] == current_user[0][0]:
      if user[2] != hash_password(old):
        err = True
        clear()
        print("Old password incorrect\n\n1. Retry\n0. Quit\n")
        choice = input("Choice: ")

        if choice == "0":
          clear()
          return [True, "Please try again later..."]

        if choice == "1":
          continue

      if not validated[0] and not err:
        user[2] = hash_password(new_password)
        break
```

Figure 7 Modify user's password function source code

Different but similar approach are implemented on the password changing action, as it is highly dependent on security features. It must be implemented without any error introducing. At first, the user's interface will be prompted to enter old password to validate with current password. If validated, the user will be prompted to input new password and confirmation password to prevent typing errors. The validated new password will immediately be hashed with the SHA256 algorithm before storing into the userlist file. Once the process is completed, it will be forced logout the user and request for relogging into the program for security purposes.

5.2.4 Add funds into wallet (User)

```
def modify_wallet(current_user):
  # ------------------------
  # Deposit money into wallet
  # access: anyone
  # ------------------------
  clear()
  balance = current_user[0][-3]
  decimal_balance = "{:.2f}".format(balance)
  print(f"Your total balance remaining: RM{decimal_balance}\n")
  print("1. Add fund\n<Enter> to Quit\n")
  add_fund = input("Do you wish to add fund into your account? ")

  while True:
    if add_fund != "1":
      return 0
        You, 2 weeks ago • -complete: version 0.0.1
    if add_fund == "1":
      amount = input("Enter the amount you wished to deposit: RM")
      userlist = read_file("./list_solution/userlist.txt")

      amount = "{:.2f}".format(int(amount))

      for user in userlist:
        if user[0] == current_user[0][0]:
          user[5] = float(user[5]) + float(amount)
          updated_user = user
          break

      write_file("./list_solution/userlist.txt", userlist)
      current_user[0] = updated_user
      clear()
      print(f"Total fund of RM{amount} has been deposited")
      input("<Enter> to return...")
      break
```

Figure 8 Modify wallet function source code

By calling this function, it requires the logged in user's credentials as parameter. At first glance, it will prompt the user's current balance onto the user interface, pending further instruction, including add fund and return to main page. If user is determined to add fund, it will request the user to input amount of funds to deposit. Once the amount has been confirmed, it will start looping through the userlist to match against the username within the userlist file for validation. Once validated, it will sum up the total deposited amount from previous wallet and saved into the database.

5.2.5 View rental history (User)

```
def rental_history(current_user):
  # -------------------------
  # View rental history
  # access: customer
  # -------------------------
  clear()
  userlist = read_file("userlist.txt")

  for user in userlist:
    if user[0] == current_user[0][0]:
      if len(user[-2]) == 0:
        print("\nStart placing order today for exclusive discounts!\n")
        return input("<Enter> to return back to home page...")

      for rent in user[-2]:       You, 2 weeks ago • -complete: version 0.0.1
        brand = rent[2].capitalize()
        model = rent[3].capitalize()
        year = rent[4]
        price_rate = rent[8]
        start_date = rent[-1][2]
        end_date = rent[-1][3]
        duration = rent[-1][1]
        str_date = start_date[0:11]
        str_enddate = end_date[0:11]

        total_price = "{:.2f}".format(float(price_rate) * int(duration))

        print("-"*20)
        print(f"\nBooked on {str_date} for the duration of {duration} days\n")
        print(f"Ends by {str_enddate}\n")
        print(f"Vehicle: {brand} {model}, {year}\n")
        print(f"Total price deducted from wallet: -RM{total_price}\n")
        print("-"*20, "\n")
      break

  end = input("<Enter> to return back to home page...")
  clear()
  return end
```

Figure 9 Rental history function source code

To call this function, it requires the user to login as a customer and pass in as a parameter. It will loop through the userlist to search for the user's username that match against the current logged in customer's username. If it matches, it will start to check if the customer has rental histories. If the rental history is empty, it will prompt the user to start order today with exclusive promotions. Otherwise, it will print out all the user's rental history. User can go back to the home page by clicking enter on the keyboard.

5.3 Vehicle Functions

5.2.1 Rent Car (User)

```python
def rent_car(id, current_user):
  # -------------------------
  # Book a car and payment
  # access: customer
  # -------------------------
  clear()
  carlist = read_file("./list_solution/carlist.txt")
  userlist = read_file("./list_solution/userlist.txt")

  for car in carlist:
    if car[0] == id:
      if car[-2]:
        return [True, "Car is already been taken by someone"]

      brand = car[2].capitalize()
      model = car[3].capitalize()
      year = str(car[4])
      price = "{:.2f}".format(car[8])

      print(f"You have selected {brand} {model}, {year}")
      print(f"Rental price for this product will be fixed at the rate of RM{price} per day\n")

      confirmation = input("Do you want to confirm order? [yes/No]: ")
      if confirmation.lower() == "no":
        return
      duration = input("How many days would you like to rent? ")

      while confirmation.lower() == "yes":
        total_price = float(price) * int(duration)

        for user in userlist:
          if user[0] == current_user[0][0]:
            if user[5] < total_price:
              return [True, "Insufficient balance, you are broke!"]

            username = current_user[0][0]

            # update car to rented
            car[-2] = True
            car[-1] = [username, duration, datetime.datetime.now(), datetime.datetime.now() + timedelta(days=int(duration))]

            # update user rental history
            user[6].append(car)
            user[5] -= total_price

            write_file("./list_solution/carlist.txt", carlist)
            write_file("./list_solution/userlist.txt", userlist)
            current_user[0] = user

            total_price = "{:.2f}".format(total_price)

            print(f"\nTotal payment made RM{total_price}")
            print(f"Your booking order for {brand} {model}, {year} for the duration of {duration} days has been confirmed\nEnjoy your ride!")

            end = input("Press Enter to return back to home page!")
            return end
      break
```

Figure 10 Rent a car function source code

In the rent a car function, it must be called with 2 parameters including, car's id and logged in user's details. It needs to read both userlist and carlist text files data to update car rental status and rental history synchronously. On first procedure, it will loop into carlist details to match against the selected car id's availability status, if its rental status is True, it will notify the customer that the car is currently rented by another user and prompt a retry page. Otherwise, it will proceed with the selected car's details and estimated price rate, a confirmation process and duration selection process will prompt respectively to confirm the rental purchase, if the user's wallet is insufficient, it will notify the user that they have to deposit to continue. On the other hand, user that are successfully purchased will be printed with an invoice statement and

the total amount spent and deducted from wallet. User's rental history will be recorded, and car status will be modified and saved into the database.

5.3.2 Add Car (Admin)

```
def add_car():
  # -------------------------
  # Add car into file
  # access: admin
  # -------------------------
  clear()
  print("-"*20)
  print("SCRS Vehicle Management")
  print("-"*20, "\n")

  num_plate = field_control("Number Plate: ", 0)
  brand = field_control("Vehicle Brand: ", 0)
  model = field_control("Vehicle Model: ", 0)
  year = field_control("Manufactured Year: ", 1)
  owner = field_control("Owner of the vehicle: ", 0)
  condition = field_control("Condition of the car [?]/10: ", 1)
  desc = field_control("Short description: ", 0)
  price_rate = field_control("Price rate per day: RM", 1)
  seats = field_control("Number of seats: ", 1)

  carlist = read_file("./list_solution/carlist.txt")

  latest_id = 0

  for car in carlist:
    if car[0] > latest_id:
      latest_id = car[0] + 1

  new_car = [int(latest_id), num_plate, brand.capitalize(), model.capitalize(), int(year), owner.capitalize(), float(condition), desc, float
  (price_rate), int(seats), False, False]

  carlist.append(new_car)

  write_file("./list_solution/carlist.txt", carlist)
  detail = input("Car has been successfully added to the system... <Enter> to return:")
  clear()
  return detail
```

Figure 11 Add car function source code

The function is restricted to only allow administrator to access. When the admin user calls the function, it will prompt a new vehicle management interface. The admin can fill in details regarding the vehicles through the field control functions which handles the input with validation. Once the details have been validated, it will begin to access the carlist file and discover the latest id from it. The new id will be assigned to the new car alongside with details into an array, which will be appended to the previous carlist. Finally, a new carlist with the latest data are stored successfully.

5.3.3 Modify Car (Admin)

```
def modify_car(id):
  # -------------------------
  # Update car details
  # extended from select car
  # access: admin
  # -------------------------
  clear()
  carlist = read_file("./list_solution/carlist.txt")

  for car in carlist:
    if car[0] == id:
      print("Modify details of", car[2], car[3], ",", car[4])
      print("Current number plate: ", car[1])
      print("\n<Enter> to keep previous data...\n")

      num_plate = field_control("Number Plate [" + car[1] + "]: ", 0, car[1])
      brand = field_control("Vehicle Brand [" + car[2] + "]: ", 0, car[2])
      model = field_control("Vehicle Model [" + car[3] + "]: ", 0, car[3])
      year = field_control("Manufactured Year [" + str(car[4]) + "]: ", 1, car[4])
      owner = field_control("Owner of the vehicle [" + car[5] + "]: ", 0, car[5])
      condition = field_control("Condition of the car [" + str(car[6]) + "/10)" + ": ", 1, car[6])
      desc = field_control("Short description: [" + car[7] + "]\n: ", 0, car[7])
      price_rate = field_control("Price rate per day: [RM" + "{:.2f}".format(car[8]) + "]: ", 1, car[8])
      seats = field_control("Number of seats [" + str(car[9]) + "]: ", 1, car[9])

      new_car = [car[0], num_plate, brand, model, int(year), owner, float(condition), desc, float(price_rate), int(seats), car[10], car[11]]
      break

  for i in range(len(carlist)):
    if carlist[i][0] == id:
      del carlist[i]
      break

  carlist.append(new_car)
  write_file("./list_solution/carlist.txt", carlist)

  return [True, "Car's details has been modified successfully"]
```

Figure 12 Modify car function source code

To modify a vehicle details, the admin must select a car's id from the menu section before proceeding. After selecting an id that are valid, it will read the carlist file and loop through the database to find the car that match against the id. After the car is validated, it will request the user to modify each of the details, if a particular detail remains unchanged, user can skip the changes by pressing enter to proceed on next field. After the necessary details are modified, it will remove the unmodified car details from the list and append the current modified version back into the carlist. Lastly, it will return a true value, indicating the process is success and a message.

5.3.4 Rented out car's detail (Admin)

```
def rented_out():
  # -------------------------
  # View vehicles that are currently rented out
  # access: admin
  # -------------------------
  clear()

  print("-"*25)
  print("CARS ON TRANSIT RECORDS")
  print("-"*25, "\n")

  carlist = read_file("carlist.txt")

  for car in carlist:
    if car[-2]:
      if len(car[-1]) > 0:
        booking_details = car[-1]
        num_plate = car[1].upper()
        brand = car[2]
        model = car[3]
        year = str(car[4])
        owner = car[5]
        condition = str(car[6])
        desc = car[7]
        price_rate = car[8]
        start_date = booking_details[2]
        str_date = start_date[0:11]
        end_date = booking_details[3]
        str_enddate = end_date[0:11]
        username = booking_details[0]
        duration = booking_details[1]

        total_price = "{:.2f}".format(int(duration) * float(price_rate))

        print("-"*20)
        print(f"Booked on {str_date} for the duration of {duration} days\n")
        print(f"Ends by {str_enddate}\n")
        print(f"Vehicle: {brand} {model}, {year}")
        print(f"Plate number: {num_plate}, owned by {owner}\n")
        print(f"Condition: {condition}/10")
        print(f"Description: {desc}\n")
        print(f"Total price deducted from wallet: -RM{total_price}\n")
        print(f"Rented by {username}")
        print("-"*20, "\n")

  end = input("<Enter> to go back...")
  clear()
```

Figure 13 Rented car detail function source code

Only administrator is allowed to view a complete record of car that is on rental by the users, it will read the entire carlist from the text file. After that, from the extracted carlist, it will perform a for loop to check onto the car rental status, which is positioned on the second last element on every list. If the rental status appear to be True, it will be displayed on the interface along with all the rental status and customer that rented it with the duration remaining.

5.3.5 Customer query (Admin)

```
def customer_payment():
  # ------------------------
  # View customer bookings and payments
  # access: admin
  # ------------------------
  clear()
  print("-"*20)
  print("SCRS Customer Order Record")
  print("-"*20, "\n")

  userlist = read_file("userlist.txt")

  for user in userlist:
    if len(user[-2]) > 0:
      username = user[0]
      email = user[1]
      total_spent = 0

      print("-"*15)
      print(f"Username: {username}")
      print(f"Email: {email}")
      print("-"*15, "\n")

      for data in user[-2]:
        start_date = data[-1][2][0:11]
        end_date = data[-1][3][0:11]
        duration = data[-1][1]
        vehicle = f"{data[2]} {data[3]}, {data[4]}"
        price_per_order = "{:.2f}".format(data[8] * int(duration))
        print(f"{vehicle}")
        print(f"Order booked on {start_date}for {duration} days")
        print(f"Total spent: RM{price_per_order}")
        print(f"Expire on {end_date}\n")

        total_spent += float(price_per_order)

      str_spent = "{:.2f}".format(total_spent)
      print(f"Total amount earned: {str_spent}\n")

  end = input("<Enter> to go back...")
  clear()
  return end
```

Figure 14 Customer query function source code

This function is designed for admin to query through the userlist to review rental history on a selected customer. It will display selected customer's payment amount and booking duration. The function works by pulling out the rental history on the selected user and display it on the interface. It will display the total amount spent by the user at the end of the query.

6.0 Additional Features

6.1 Assign new administrator

```python
def assign_admin():
    # -------------------------
    # Assign a new user to be an administrator
    # access: admin
    # -------------------------
    userlist = read_file("userlist.txt")

    usernames = [] # list of registered usernames

    # display usernames
    for user in userlist: ...
    usernames = list(set(usernames))
    usernames.sort()

    print("ASSIGN AN ADMINISTRATOR\n")

    print(f"Search by usernames:\n")
    count = 0
    for names in usernames: ...

    print("\n<Enter> to return")
    selected_username = input("\nType an username that are in the list: ")

    # return
    if selected_username == "": ...

    # error handling      You, 2 days ago • -update: assign admin + feedback setup
    if selected_username.isnumeric(): ...

    # error handling 2
    if not selected_username.isnumeric() and not selected_username.lower() in usernames: ...

    for user in userlist:
        if user[0] == selected_username:
            confirmation = input(f"\nDo you wanna assign {user[0]} as an admin? [yes/No] ")

            if confirmation.lower() == "yes":
                user[7] = "admin"
                write_file("userlist.txt", userlist)
                clear()
                print("-"*30)
                print("SCRS MEMBER MANAGEMENT")
                print("-"*30, "\n")
                print(f"{user[0]} has successfully promoted as an administrator for SUPER CAR RENTAL SYSTEM\n")
                return input("<Enter> to return to admin menu...")
            else:
                return ""
```

Figure 15 Assign admin function

The first additional feature included in the SCRS program is the ability to assign new admin. This allows the manager to make their new employees admin accounts and they would not need to rely on only one admin account. The only users to access this feature is an existing admin.

```
Welcome, Admin

1. Add a Vehicle
2. Modify a Vehicle's Details
3. Update Personal Information
4. Vehicle Rental Records
5. Query Customer Record

6. Assign a new administrator
7. Customer Feedback

0. Logout

Please enter your choice: 6
@ASSIGN AN ADMINISTRATOR

Search by usernames:

* chiawx
* eugenetin
* kelvin
* lololan
* newbie
* qwerty
* valorant
* wenxuen

<Enter> to return

Type an username that are in the list:
```

Figure 4 Choice to add Admin

The users shown are not assigned as admin. The existing admin would now need to enter the name of user to assign as admin. Of course, there is also a return option for the admin to change their mind of promoting an account.

6.2 Display Feedback/Suggestion

```
def display_feedback(current_user=[]):
  # ------------------------
  # Display feedbacks by customers that has used our service
  # access: everyone
  # ------------------------
  userlist = read_file("userlist.txt")

  print("-"*30)
  print("SUPER CAR RENTAL SERVICE (SCRS)")
  print("-"*30, "\n")

  header = ["Username", "Rating", "Customer Feedback"]
  format_row = "|{:^25}|{:^40}|{:^80}|"

  print(format_row.format(*header))
  print("-"*150)

  for user in userlist:
    if len(user) == 9:
      username = user[0]
      rating = user[8][0]
      feedback = user[8][1][0:60] + "..."

      print(format_row.format(username, "☆ "*rating, feedback))

  if len(current_user) > 0:
    while len(current_user[6]) != 0 and len(current_user) == 8:
      choice = submit_feedback(current_user[0])

      if len(choice[1]) > 0:
        current_user = choice[1]
      if choice[0] == "":
        break

  end = input("\nPress <Enter> to return to main menu...")
  clear()

  return [end, current_user]
```

Figure 16 Display feedback function source code

In this additional feature, it is added to display the customer feedback after enjoying the service provided by SCRS. It can be access by anyone including logged in user, admin user, and regular user. The function is called by passing in a logged in user details, or else it will be an empty list. When the function is called, it will loop through userlist to check if the customer has a feedback. If feedback is existed, it will print out the username, rating, and feedback submitted by the user. Other than that, if the logged in user has not provided any feedback and used our services before, it will ask whether the user is interested in leaving a feedback for this online service.

6.3 Submit Feedback/Suggestion

```python
def submit_feedback(username):
  # ------------------------
  # Allow customer to submit feedback
  # access: customer that used SCRS at least once
  # ------------------------
  userlist = read_file("userlist.txt")
  submission = input("\nDo you want to submit your own feedback or provide any suggestions? [yes/No] ")

  if submission.lower() == "yes":
    clear()

    print("-"*30)
    print("SUPER CAR RENTAL SERVICE (SCRS)")
    print("-"*30, "\n")

    print("Customer Feedback Form\n")

    while True:
      rating = input("On a scale of 1-5, how would you rate Super Car Rental Service? ")

      if rating.isnumeric() and rating > "0" and rating < "6":
        break

    while True:
      feedback = input("Feel free to give short opinion on our service: ")

      if len(feedback) < 10:
        print("Message length should be greater than 10 characters")
      else:
        break

    print("-"*30)

    for user in userlist:
      if user[0] == username:
        review = [int(rating), feedback]
        user.append(review)
        write_file("userlist.txt", userlist)
        end = input("Your feedback has been submitted successfully! Press <Enter> key to return...")
        return [end, user]
  else:···
```

Figure 17 Submit feedback by customer function source code

This function is extended from the display feedback function, it is only accessible by user that fulfill these 2 conditions, including the user must be a customer, and it must at least use our services once to submit the feedback. This function must be called with a logged in username as a parameter in order to continue, if the user has confirmed to submit a feedback, it will prompt user with the rating and feedback form to fill up. Once the feedback form has been filled and validated, the data will be stored by appending into the user's database.

7.0 Sample input/output with explanation

7.1 Login as admin

```
C:\Users\wenxu\AppData\Local\Progra
--------------------
Super Car Rental Service (SCRS)
--------------------

1. Login
2. Register
3. View Cars
4. Feedback/Suggestion

0. Quit

Please select a choice: 1
LOGIN

Username: admin
Password: admin
```

Figure 7.1.1Main_Menu

```
Username: admin
Password: admin
You have login successfully
--------------------
Super Car Rental Service (SCRS)
--------------------

Welcome, Admin

1. Add a Vehicle
2. Modify a Vehicle's Details
3. Update Personal Information
4. Vehicle Rental Records
5. Query Customer Record

6. Assign a new administrator
7. Customer Feedback

0. Logout

Please enter your choice:
```

Figure 7.1.2 Admin Interface

When running the python file, the first thing that appears will be the main menu. When the user chooses login, they will be prompt to enter login credentials such as username and password. In this case, the user is using the admin login credentials. Then, when the account username and password is the same as the data in user list, then it will show the admin page, The features here include add vehicle, modify vehicle, update personal information, vehicle rental records, customer query, assign new admin and customer feedback.

```
Please enter your choice: 4
--------------------
SCRS Vehicle Management
--------------------

1. Vehicles in transit
2. Vehicles available for Rent
3. Customer Payments for a specific time duration

0.Back

Please enter your choice:
```

Figure 7.1.3 choice 4

When one of the choices is entered (in this case No.4. View vehicle rent records), This will then output the records available to view. Next the admin will need to enter the records they wish to see. The output will be as follow:

Next the only option left is to return to previous stage. The admin can choose to view other records or go back to admin menu.

```
<Enter> to go back...
000--------------------
SCRS Vehicle Management
--------------------

1. Vehicles in transit
2. Vehicles available for Rent
3. Customer Payments for a specific time duration

0.Back

Please enter your choice:
```

Figure 7.1.5 Back to previous selection

```
Please enter your choice: 1
00-----------------------
CARS ON TRANSIT RECORDS
-----------------------

    Number Plate    |      Vehicle       |     Booked on      |     Expi
-------------------------------------------------------------------------
      ABC5678       |Mercedes Benz, 2019 |    2021-06-13      |    2021-06-23      |       eugene       |      1000.00       |       kelvin       |
      DEF1234       | Honda Civic, 2010  |    2021-06-12      |    2021-06-17      |       Eugene       |       500.00       |      eclipse       |

<Enter> to go back...
```

Figure 7.1.4 Vehicles in transit

```
1. Vehicles in transit
2. Vehicles available for Rent
3. Customer Payments for a specific time duration

0.Back

Please enter your choice: 0
0--------------------
Super Car Rental Service (SCRS)
--------------------

Welcome, Admin

1. Add a Vehicle
2. Modify a Vehicle's Details
3. Update Personal Information
4. Vehicle Rental Records
5. Query Customer Record

6. Assign a new administrator
7. Customer Feedback

0. Logout

Please enter your choice:
```

Figure 7.1.6 Back to admin menu

If the users decide to go back to admin menu to use other admin functions, they could just input 0 in the previous stage and they will see the admin menu selection.

When the admin is done with their work and wants to logout, they could simply do so by input 0 in the admin menu. The main menu will then be displayed.

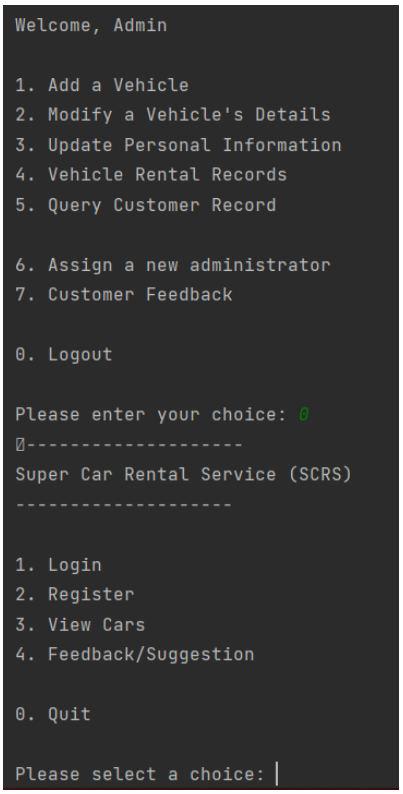


Figure 7.1.7 back to main menu

7.2 Register

```
Please enter your choice: 0
--------------------
Super Car Rental Service (SCRS)
--------------------

1. Login
2. Register
3. View Cars
4. Feedback/Suggestion

0. Quit

Please select a choice: 2
REGISTRATION
------------
Username: newbie
Email: newbie@gmail.com
Password: newbie6789
Confirm Password: newbie6789
----------------
Personal Details
---------------
Contact Number: +60163546273
Currently lived in [state]: Selangor

Would you like to make an initial deposit into your wallet?
<Enter> to skip the deposit

Deposit amount: RM500
Total amount of RM500.00 deposited into your account
You have registered successfully, please login now...
```

Figure 7.2.1 Register new account

When the user does not have an account and wants to create an account, they could input choice 2 to register as a member of SCRS. Once executed, the user is required to enter basic information such as a username, email, account password, contact number, and residing state.

Furthermore, the user would be asked if they would want to make an initial deposit. If they agree, then they can input the amount wished to deposit, else they can simply press <Enter> to skip this step.

```
Deposit amount: RM500
Total amount of RM500.00 deposited into your account
You have registered successfully, please login now...

1. Login
2. Register
3. View Cars
4. Feedback/Suggestion

0. Quit

Please select a choice:
```

Figure 7.2.2 Registration complete

```
Please select a choice: 1
LOGIN

Username: newbie
Password: newbie6789
You have login successfully
------------------------------
SUPER CAR RENTAL SERVICE
------------------------------

Welcome, Newbie

1. Rent a Car
2. Update Personal Information
3. Rental History
4. Check Wallet
5. Customer Feedback

0. Logout

Please enter your choice:
```

Figure 7.2.3 Login as new account

After the account is created, they will then be brought back to the menu to login to their account.

7.3 View Cars

```
------------------------------
Select Vehicle Brand
------------------------------

1. 404
2. Honda
3. Mercedes
4. Perodua
5. Proton

0. Go back to home page
Select a model: 2
--------------------
Honda
--------------------

  ID  | Number Plate  |      Vehicle       | Seats  |         Short Description         | Condition  |     Owner     |Price Rate|   Rental Status    |
-------------------------------------------------------------------------------------------------------------------------------------------------------
  2   |    DEF1234    | Honda Civic, 2010  |   5    |          Old but gold...          |     4.0    |    Eugene     |  100.00  | Rented by eclipse  |

Press Enter to quit:
```

Figure 7.3.1 View Car

If the user does not have an account but still wants to view cars, they can input option 3 to view cars. Next, the user is required to choose the brand to view. After input of the desired brand, the model of the cars will be displayed according to the chosen brand. After viewing the car details, the user may quit to return to main menu.

8.0 Conclusion

In a nutshell, the SCRS program created was able to run after uncountable trials and errors. It is through these trials and errors that the codes could be further polished to run smoothly. This program lets user to login to the site. If the user is an admin, he or she may have access to admin functions, if the user is a customer, he or she will only have access to customer functions. If the user is not logged in, they may choose to view all cars or register as a member.

9.0 Limitations

- ✓ Customer payments does not integrate with payment methods such as credit/debit and PayPal
- ✓ Forget password is not implemented as the solution does not met with the assignment criteria.
- ✓ Validation might be inconsistent as it does not validate certain details correctly such as contact number and state in Malaysia.
- ✓ Bugs might still exist in the program

References

Bartleby, 2020. *bartleby.* [Online]
Available at: https://www.bartleby.com/questions-and-answers/part-1-write-pseudocode-for-an-iterative-algorithm-which-finds-the-maximum-value-of-a-list-of-intege/e41a8d50-eb4e-4fa8-bc18-1e680ed3f747
[Accessed 13 Mei 2021].

Stackoverflow, 2021. *Stackoverflow.* [Online]
Available at: https://stackoverflow.com/
[Accessed 20 May 2021].

W3Schools, 2021. *W3Schools: Python-Lists.* [Online]
Available at: https://www.w3schools.com/python/python_lists.asp
[Accessed 20 May 2021].