# GROUP ASSIGNMENT

**TECHNOLOGY PARK MALAYSIA**

**AAPP010-4-2-PWP**

**PROGRAMMING WITH PYTHON**

**UCDF2005-ICT(1)ICT(DI)**

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

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

**WEIGHTAGE:** **100%**

**LECTURER:** **Mr. Ts. Sivaguru Subarmaniyan**

**PREPARED BY:**


| No. | Name | TP Number |
|---|---|---|
| 1 | Lai Kai Yong | TP059040 |
| 2 | Lim Wye Yee | TP059371 |


---

**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**

P.S.:

**To whoever it may concern**, we have a total of 90 main functions and nested functions.

We had tried our best to condense the explanation and manage to decrease the number of pages from 500 to 100 (Intro to conclusion).

Thank you for your understanding.

## 1.0 Introduction

The Super Car Rental Services (SCRS) provides services on online car rental within Malaysia. An online car rental system (OCRS) is planned and developed aiming to implement the service operations involving all customers and administrators. It enables customers to book their desired vehicle with upfront payment while the admins handle the transactions and record statement through the online system. The OCRS is developed to monitor the vehicles, customers and admins virtually. Booking and payment statements are displayed with specific status and recorded with details which had simplify the business process. The OCRS is centralized based so that the portal is accessible in any locations and anytime. Besides, the system improves the coordination between staffs because redundant data can be avoided if the data is stored and retrieved from the same database files. With the use of the OCRS, the confidential customer's details can be kept secured with fixed integrity and renting progress will be more efficient and automated.

## 1.1 Assumptions for the program

Many assumptions had been made while creating the online car rental system (OCRS) for Super Car Rental Services (SCRS). The usernames and passwords are already created for respective admins so that they do not have to create a new account. Admins are required to access the functionalities with the specific credentials where no external users will be able to access the admins system. In addition, admins will need to acquire their functionalities with username and password, but customers can just access the system without credentials with limited functionalities. For privacy concern, the customers will not be able to access the car renting system without signing up as registered users as the customers' usernames or passwords that are not registered to the system database file are invalid. After selecting roles, the respective menu will be shown to the users based on their selected roles such as admins, visitors and registered customers.

All options that are not included as a choice are invalid and users will be given several chances to reinsert their request. For the registered customers, they are given only three attempts to log in or they will be directed back to the role selection page / welcome page. The car IDs and customer username within the system cannot be redundant as it is a unique identifier for each customer and car. Furthermore,

some columns of the record can be null (put it as N/A) especially for some attributes in newly created booking statement. Users (admins / customers) can direct themselves back to the main menu or any other available menus after they finish an action.

For the admins, they can:

- Modify records in any database except for customer details database file
- Display the data in database, search for specific information to inspect the record
- Return the car to the system after customers had finished their renting period
- Notify customers after processing their orders successfully by marking the cars as “Ready”
- Access the analytics dashboard to determine the sales performance of the car renting system

Additionally, the new details will be added to the end of the database file and the new modified details will overwrite the original data in the database. When no records are available inside the database, a note stating that “no relevant data for records on that particular status” will be displayed and automation of redirection will be triggered. Meanwhile, the admin can create the database when it is corrupted or file is not available.

In the customers’ perspective, a customer who rents a new car will add a new booking / payment record for the customer. To confirm their booking and make a valid request, they will have to proceed to payment, their car will also be reserved after that. They can complete their payment through their balance in the system or credit card payments. If their balance amount is insufficient, they can reload their balance through credit/debit card or FPX online banking with unlimited value of money. The top up system is supported by five merchants for FPX online banking – Maybank, Public bank, Ambank, RHB bank and CIMB bank. The rental system charges the customers based on how many days the customers will be renting the car and also the price per day. After the customers confirm their booking, they can select a desired rent date for the car they had just booked. Besides, customers must verify themselves by entering their username to access private data such as personal details and balance. Moreover, they can rent more than one car at the same time and they will need to claim their car once it is ready to be rented. Many validations had also been included to avoid value fault of users’ insertion and choice validation.

## 1.2 Status Explanation

**Car status:**

**X:** The car is no longer available for rent.

**Open:** The car is available for renting.

**Booked:** The car is considered as booked after a customer finishes their payment.

**Rented:** The car is in the process of renting after a customer has claimed their car.

**Status Flow:** Open → Booked → Rented

The car will return to the 'Open' status after the customer end their rental period and the admin manually altered the record in the database.

**Customer payment status:**

**Pending:** The customer has not paid yet / the payment is waiting for transaction.

**Paid:** The customer had paid the full amount of rent.

**Status Flow:** Pending → Paid

**Customer booking/reservation status:**

**N/A:** The reservation status will display as this when a new booking is placed.

**In Queue:** The car rent had been paid and the admins are processing the order while preparing the requested car after conducting several checking on the car.

**Ready:** The requested car is ready to be claimed by the customer, cars with this status will be displayed in the customer's claim car page to ease the process.

**Renting:** The car is deemed to be in this status after the customer had claimed their car and is already using the rented car.

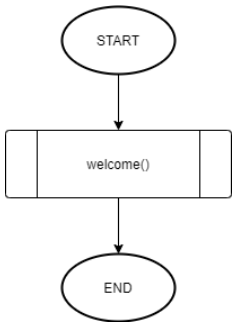
**Completed:** This status will be displayed after the admin manually alter the records when customer return their car. .

**Status Flow:** N/A → In Queue → Ready → Renting → Completed

## 2.0 Design of the program

- Pseudocode & FlowChart

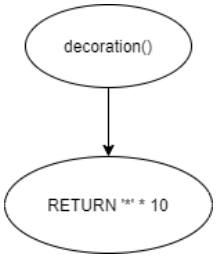
```
PROGRAM executeOCRS
BEGIN
        CALL welcome()
END
```




### Section A: General Code Functionalities

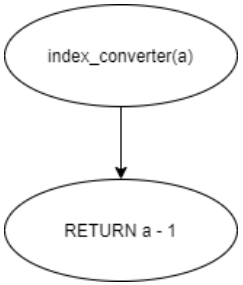
a. Decoration function

```
FUNCTION decoration()
        RETURN '*' * 10
ENDFUNCTION
```




b. Converting lines to modify to index

```
FUNCTION index_converter(a)
        RETURN a - 1
ENDFUNCTION
```




### Section B: Headers

a. Header for car tables

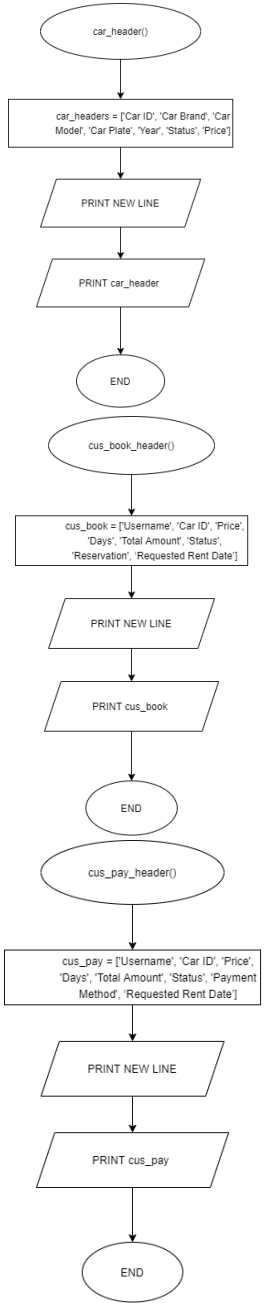
```
FUNCTION car_header()
        car_headers = ['Car ID', 'Car Brand', 'Car Model',
                'Year', 'Status', 'Price']
        PRINT NEW LINE
        PRINT car_headers
ENDFUNCTION
```

b. Header for customer statement tables (booking)

```
FUNCTION cus_book_header()
        cus_book = ['Username', 'Car ID', 'Price', 'Days', 'Total Amount', 'Status',
        'Reservation', 'Requested Rent Date']
        PRINT NEW LINE
        PRINT cus_book
ENDFUNCTION
```

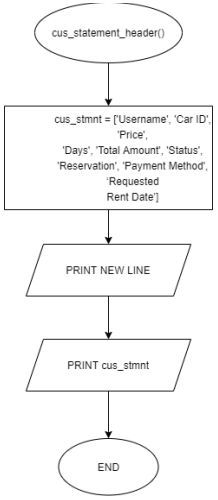
c. Header for customer statement tables (payment)

```
FUNCTION cus_pay_header()
        cus_pay = ['Username', 'Car ID', 'Price', 'Days', 'Total Amount', 'Status', 'Payment
        Method', 'Requested Rent Date']
        PRINT NEW LINE
        PRINT cus_pay
ENDFUNCTION
```



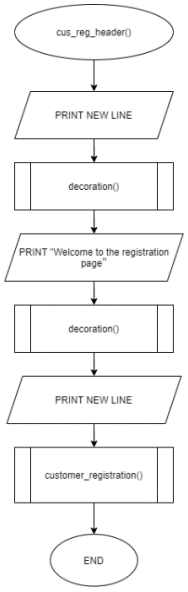
d. Header for customer statement tables

```
FUNCTION cus_statement_header()
        cus_stmnt = ['Username', 'Car ID', 'Price', 'Days', 'Total Amount', 'Status',
        'Reservation', 'Payment Method', 'Requested Rent Date']
        PRINT NEW LINE
        PRINT cus_stmnt
ENDFUNCTION
```



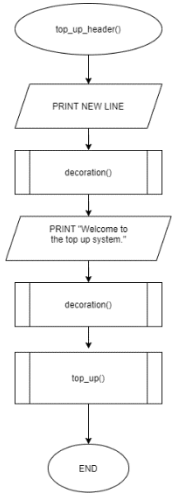

e. Customer registration page header

```
FUNCTION cust_reg_header()
        PRINT NEW LINE
        CALL decoration()
        PRINT "Welcome to the registration page"
        CALL decoration()
        PRINT NEW LINE
        CALL customer_registration()
ENDFUNCTION
```




f. Customer top up page header

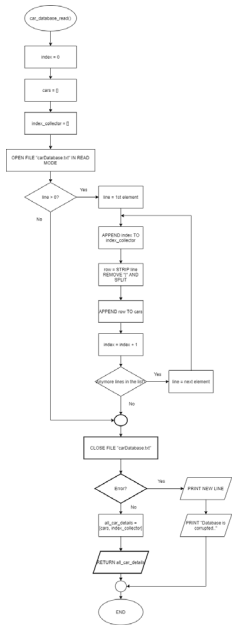
```
FUNCTION top_up_header()
        PRINT NEW LINE
        CALL decoration()
        PRINT "Welcome to the top up system."
        CALL decoration()
        CALL top_up()
ENDFUNCTION
```




## Section C: Database / Text file data management

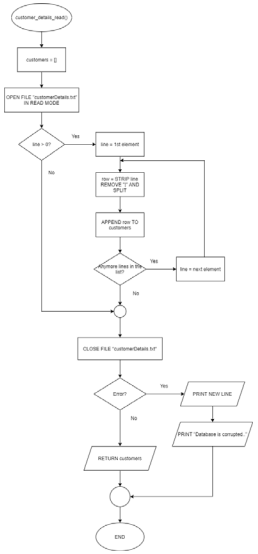
a. Read car database file

```
FUNCTION car_database_read()
        TRY
                index = 0
                cars = []
                index_collector = []
                OPEN FILE "carDatabase.txt" IN READ MODE AS car_file
                        FOR EACH line IN car_file
                                ADD index TO index_collector
                                row = STRIP line REMOVE " | " AND SPLIT
                                ADD row TO cars
                                index = index + 1
                        ENDFOR
                CLOSE FILE "carDatabase.txt"
        EXCEPT ERROR THEN
                PRINT NEW LINE
                PRINT "Database is corrupted.."
        ENDTRY
        all_car_details = [cars, index_collector]
        RETURN all_car_details
ENDFUNCTION
```



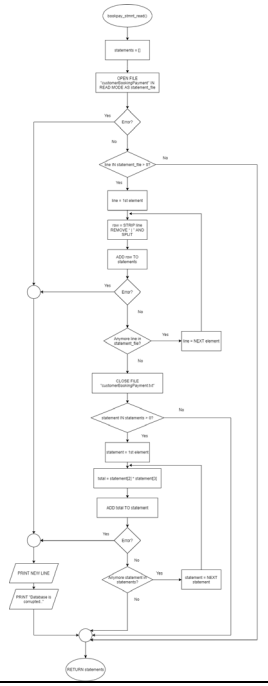
b. Read customer details database file

```
FUNCTION customer_details_read()
    TRY
        customers = []
        OPEN FILE “customerDetails.txt” IN READ MODE AS customers_file
            FOR EACH line IN customers_file
                row = STRIP line REMOVE “ | ” AND SPLIT
                ADD row TO customers
            ENDFOR
        CLOSE FILE “customerDetails.txt”
    EXCEPT ERROR THEN
        PRINT NEW LINE
        PRINT "Database is corrupted.."
    ENDTRY
    RETURN customers
ENDFUNCTION
```




c. Read customer booking / payment statement

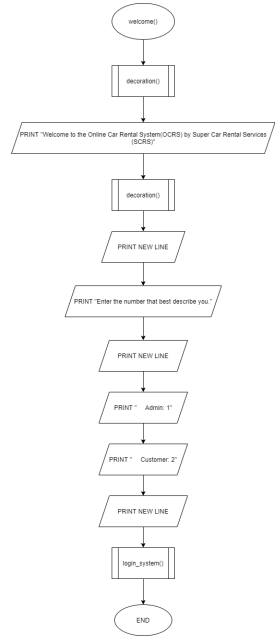
```
FUNCTION bookpay_stmnt_read()
    TRY
        statements = []
        OPEN FILE “customerBookingPayment” IN READ MODE AS statement_file
            FOR EACH line IN statement_file
                row = STRIP line REMOVE “ | ” AND SPLIT
                ADD row TO statements
            ENDFOR
        CLOSE FILE “customerBookingPayment.txt”
        FOR EACH statement IN statements
            total = statement[2] * statement[3]
            ADD total TO statement
        ENDFOR
    EXCEPT ERROR THEN
        PRINT NEW LINE
        PRINT "Database is corrupted.."
     ENDTRY
     RETURN statements
ENDFUNCTION
```




## Section D: General Users Interface

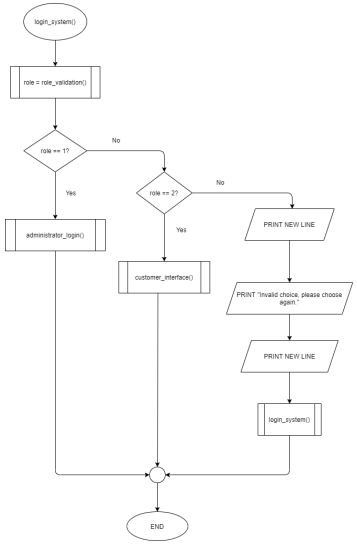
a. Welcome page

```
FUNCTION welcome()
    CALL decoration()
    PRINT "Welcome to the Online Car Rental System(OCRS) by Super Car Rental Services(SCRS)"
    CALL decoration()
    PRINT NEW LINE
    PRINT “Enter the number that best describe you.”
    PRINT NEW LINE
    PRINT “   Admin : 1”
    PRINT “   Customer : 2”
    PRINT NEW LINE
    CALL login_system()
ENDFUNCTION
```



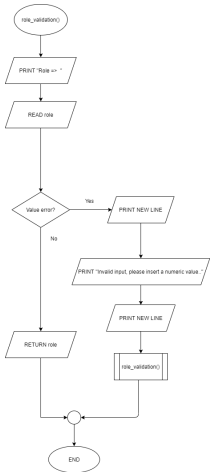

b. Login system

```
FUNCTION login_system()
    role = CALL role_validation()
    IF (role == 1) THEN
        CALL administrator_login()
    ELSE
        IF (role == 2) THEN
            CALL customer_interface()
        ELSE
            PRINT NEW LINE
            PRINT "Invalid choice, please choose again."
            PRINT NEW LINE
            CALL login_system()
        ENDIF
    ENDIF
ENDFUNCTION
```



- Role validation

```
FUNCTION role_validation()
        TRY
                PRINT “Role  => ”
                READ role
                RETURN role
        EXCEPT VALUE ERROR THEN
                PRINT NEW LINE
                PRINT “Invalid input, please insert a numeric value..”
                PRINT NEW LINE
                CALL role_validation()
        ENDTRY
ENDFUNCTION
```



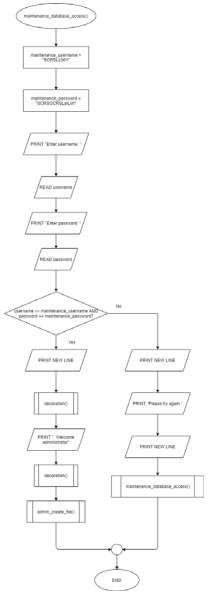

c. Exit program

```
FUNCTION exit_system()
        PRINT NEW LINE
        PRINT “Are you sure you want to exit the Online Car Rental System?”
        PRINT NEW LINE
        PRINT "   [YES] or [NO]”
        PRINT NEW LINE
        PRINT “Note: Selecting [NO] will navigate you back to the welcome page.”
       PRINT “Option => "
       READ confirmation
       IF (confirmation == "YES") THEN
                PRINT NEW LINE
                CALL decoration()
                PRINT "Exit"
                CALL decoration()
                PRINT NEW LINE
                END
       ELSE
                IF (confirmation == "NO") THEN
                        PRINT "You will be redirect to the welcome page shortly..."
                        PRINT NEW LINE
                        CALL welcome()
                ELSE
                        PRINT "Invalid input, please enter YES or NO."
                        CALL exit_system()
                ENDIF
       ENDIF
ENDFUNCTION
```


## Section E: Administrator

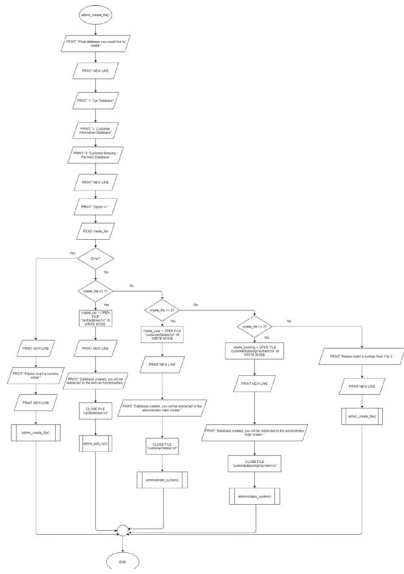
a. Database corrupted / Admin create database file

```
FUNCTION maintenance_database_access()
        maintenance_username = “SCRSLL001”
        maintenance_password = “SCRSOCRSLaiLim”
        PRINT “Enter username: ”
        READ username
        PRINT "Enter password: "
        READ password
        IF (username == maintenance_username) AND (password ==
maintenance_password)        THEN
                PRINT NEW LINE
                CALL decoration()
                PRINT “ Welcome administrator ”
                CALL decoration()
                CALL admin_create_file()
        ELSE
                PRINT NEW LINE
                PRINT “Please try again.”
                PRINT NEW LINE
                CALL maintenance_database_access()
        ENDIF
ENDFUNCTION
```



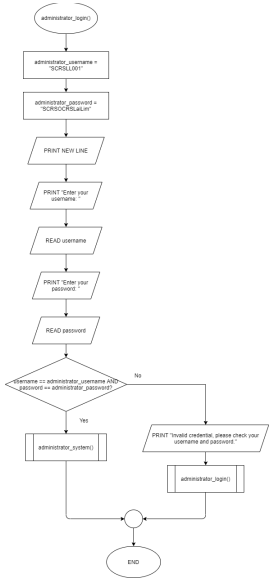

- Admin create file

```
FUNCTION admin_create_file()
    TRY
        PRINT “What database you would like to create”
        PRINT NEW LINE
        PRINT “1: Car Database”
        PRINT “2: Customer information Database”
        PRINT “3: Customer Booking / Payment Database”
        PRINT NEW LINE
        PRINT “Option => ”
        READ create_file
        IF (create_file == 1) THEN
            create_car = OPEN FILE “carDatabase.txt” IN WRITE MODE
            PRINT NEW LINE
            PRINT “Database created, you will be redirected to the add car
                  functionalities.”
            CLOSE FILE “carDatabase.txt”
            CALL admin_add_car()
        ELSE
            If (create_file == 2) THEN
                create_user = OPEN FILE “customerDetails.txt” IN WRITE MODE
                PRINT NEW LINE
                PRINT “Database created, you will be redirected to the administrator main screen”
                CLOSE FILE “customerDetails.txt”
                CALL administrator_system()
            ELSE
                IF (create_file == 3) THEN
                    create_booking = OPEN FILE “customerBookingPayment.txt” IN WRITE MODE
                    PRINT NEW LINE
                    PRINT “Database created, you will be redirected to the administrator main screen.”
                    CLOSE FILE “customerBookingPayment.txt”
                    CALL administrator_system()
                ELSE
                    PRINT “Please insert a number from 1 to 3.”
                    PRINT NEW LINE
                    CALL admin_create_file()
                ENDIF
            ENDIF
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT “Please insert a numeric value.”
        PRINT NEW LINE
        CALL admin_create_file()
    ENDTRY
ENDFUNCTION
```



## b. Admin login

```
FUNCTION administrator_login()
        administrator_username = “SCRSLL001”
        administrator_password = “SCRSOCRSLaiLim”
        PRINT NEW LINE
        PRINT "Enter your username: "
        READ username
        PRINT "Enter your password: "
        READ password
        IF (username == administrator_username) AND (password ==
      administrator_password) THEN
              CALL administrator_system()
        ELSE
              PRINT "Invalid credential, please check your username and password."
              CALL administrator_login()
        ENDIF
ENDFUNCTION
```

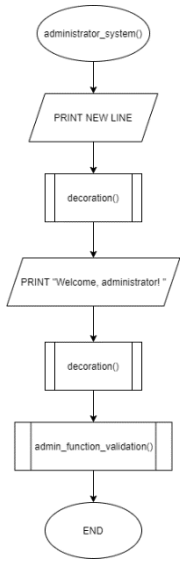


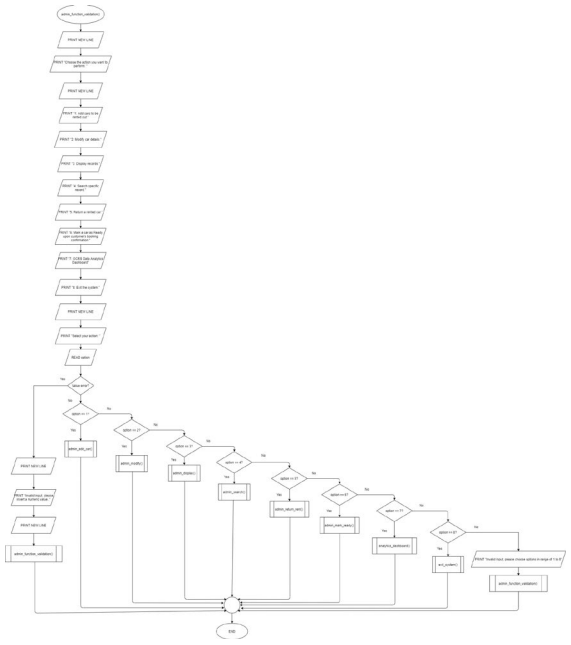

## Admin function validation

```
FUNCTION admin_function_validation()
    TRY
        PRINT NEW LINE
        PRINT “Choose the action you want to perform: ”
        PRINT NEW LINE
        PRINT “1: Add cars to be rented out.”
        PRINT “2: Modify car details.”
        PRINT “3: Display records.”
        PRINT “4: Search specific record.”
        PRINT “5: Return a rented car.”
        PRINT “6: Mark a car as Ready upon customer's booking confirmation”
        PRINT “7: OCRS Data Analytics Dashboard”
        PRINT “8: Exit the system.”
        PRINT NEW LINE
        PRINT “Select your action: ”
        READ option
        IF (option == 1) THEN
            CALL admin_add_car()
        ELSE
            IF (option == 2) THEN
                CALL admin_modify()
            ELSE
                IF (option == 3) THEN
                    CALL admin_display()
                ELSE
                    IF (option == 4) THEN
                        CALL admin_search()
                    ELSE
                        IF (option == 5) THEN
                            CALL admin_return_rent()
                        ELSE
                            IF (option == 6) THEN
                                CALL admin_mark_ready()
                            ELSE
                                IF (option == 7) THEN
                                    CALL analytics_dashboard()
                                ELSE
                                    IF (option == 8) THEN
                                        CALL exit_system()
                                    ELSE
                                        PRINT "Invalid input, please choose options in range of
                                        1 to 8"
                                        CALL admin_function_validation()
                                    ENDIF
                                ENDIF
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value.."
        PRINT NEW LINE
        CALL admin_function_validation()
    ENDTRY
ENDFUNCTION
```

## c. Administrator access system

```
FUNCTION administrator_system()
        PRINT NEW LINE
        CALL decoration()
        PRINT “Welcome, administrator!”
        CALL decoration()
        CALL admin_function_validation()
ENDFUNCTION
```

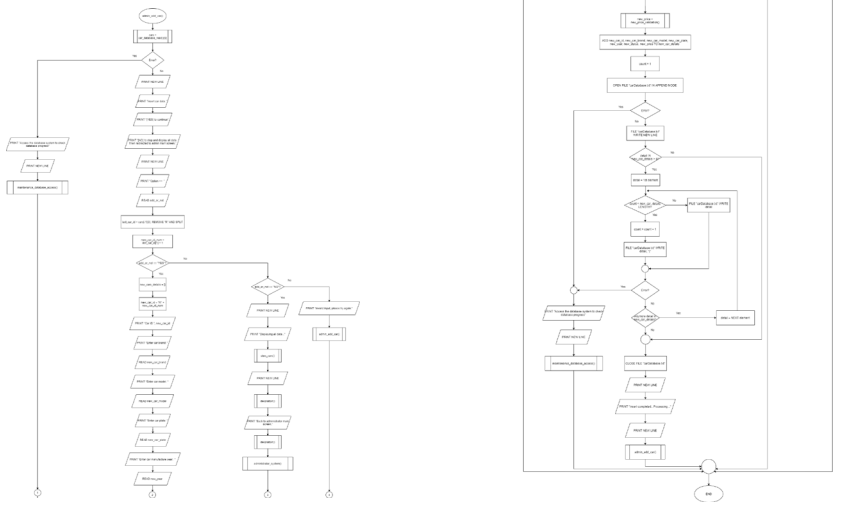


## Section E01: Administrator functionalities

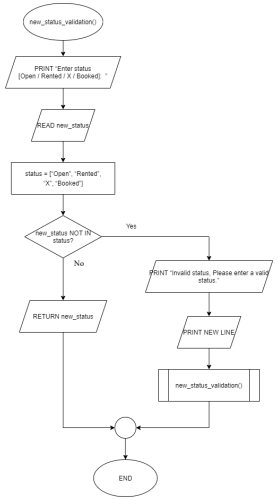
### a. Add car

```
FUNCTION admin_add_car()
    TRY
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database progress"
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    PRINT NEW LINE
    PRINT "Insert car data:"
    PRINT "[YES] to continue"
    PRINT "[NO] to stop and display all data. Then redirected to admin main screen."
    PRINT NEW LINE
    PRINT "Option =>     "
    READ add_or_not
    last_car_id = cars[-1][0], REMOVE "R" AND SPLIT
    new_car_id_num = last_car_id[1] + 1
    IF (add_or_not == "YES") THEN
        new_cars_details = []
        new_car_id = "R" + new_car_id_num
        PRINT "Car ID: ", new_car_id
        PRINT "Enter car brand: "
        READ new_car_brand
        PRINT "Enter car model: "
        READ new_car_model
        PRINT "Enter car plate: "
        READ new_car_plate
        PRINT "Enter car manufacture year: "
        READ new_year
        new_status = CALL new_status_validation()
        new_price = CALL new_price_validation()
        ADD new_car_id, new_car_brand, new_car_model, new_car_plate, new_year,
            new_status, new_price TO new_car_details
        TRY
            count = 1
            OPEN FILE "carDatabase.txt" IN APPEND MODE
                FILE "carDatabase.txt" WRITE NEW LINE
                FOR EACH detail IN new_car_details
                    IF (count < new_car_details LENGTH) THEN
                        count = count + 1
                        FILE "carDatabase.txt" WRITE detail, " | "
                    ELSE
                        FILE "carDatabase.txt" WRITE detail
                    ENDIF
                ENDFOR
            CLOSE FILE "carDatabase.txt"
            PRINT NEW LINE
            PRINT "Insert completed... Processing..."
            PRINT NEW LINE
        EXCEPT ERROR THEN
            PRINT "Access the database system to check database progress"
            PRINT NEW LINE
            CALL maintenance_database_access()
        ENDTRY
        CALL admin_add_car()
    ELSE
        IF (add_or_not == "NO") THEN
            PRINT NEW LINE
            PRINT "Displaying all data..."
            CALL view_cars()
            PRINT NEW LINE
            CALL decoration()
            PRINT "Back to administrator main screen."
            CALL decoration()
            CALL administrator_system()
        ELSE
            PRINT "Invalid input, please try again."
            CALL admin_add_car()
        ENDIF
    ENDIF
ENDFUNCTION
```



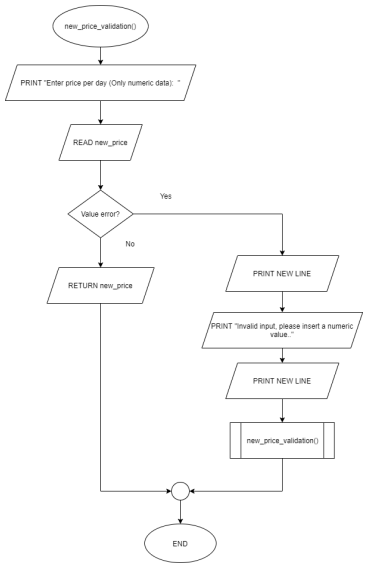
- New status validation

```
FUNCTION new_status_validation()
    PRINT "Enter status [Open / Rented / X / Booked]:  "
    READ new_status
    status = ["Open", "Rented", "X", "Booked"]
    IF new_status NOT IN status THEN
        PRINT "Invalid status, Please enter a valid status."
        PRINT NEW LINE
        CALL new_status_validation()
    ELSE
        RETURN new_status
    ENDIF
ENDFUNCTION
```



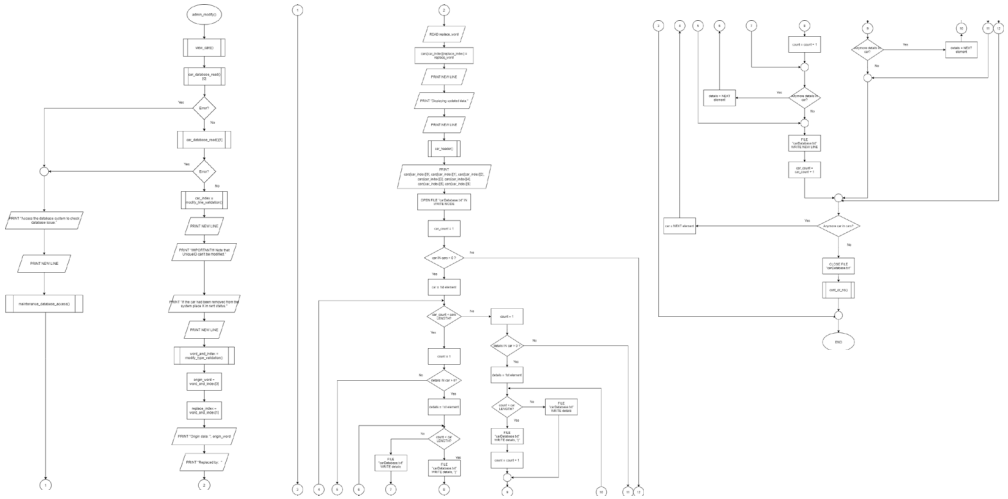

- New price validation

```
FUNCTION new_price_validation()
    TRY
        PRINT "Enter price per day (Only numeric data): "
        READ new_price
        RETURN new_price
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value.."
        PRINT NEW LINE
        CALL new_price_validation()
    ENDTRY
ENDFUNCTION
```



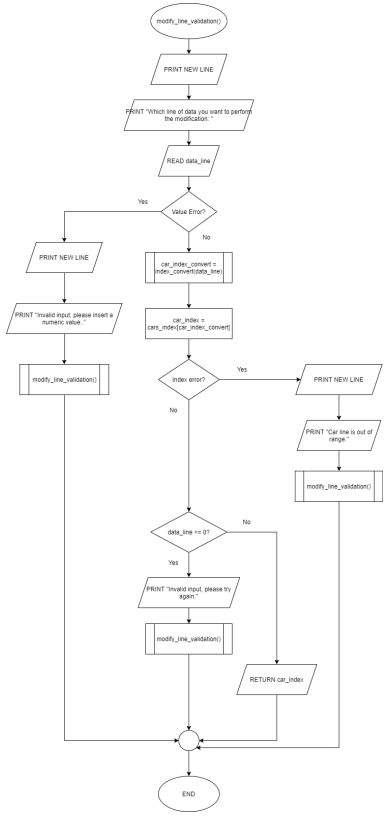
## b. Modify car

```
FUNCTION admin_modify()
    CALL view_cars()
    TRY
        cars = CALL car_database_read()[0]
        cars_index = CALL car_database_read()[1]
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    car_index = CALL modify_line_validation()
    PRINT NEW LINE
    PRINT "IMPORTANT!!! Note that uniqueID can't be modified."
    PRINT "If the car had been removed from the system place X in rent status."
    PRINT NEW LINE
    word_and_index = CALL modify_type_validation()
    origin_word = word_and_index[0]
    replace_index = word_and_index[1]
    PRINT "Origin data: ", origin_word
    PRINT "Replaced by: "
    READ replace_word
    cars[car_index][replace_index] = replace_word
    PRINT NEW LINE
    PRINT "Displaying updated data."
    PRINT NEW LINE
    CALL car_header()
    PRINT cars[car_index][0], cars[car_index][1], cars[car_index][2], cars[car_index][3],
    cars[car_index][4], cars[car_index][5], cars[car_index][6]
    OPEN FILE "carDatabase.txt" IN WRITE MODE
        car_count = 1
        FOR EACH car IN cars
            IF (car_count < cars LENGTH) THEN
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "carDatabase.txt" WRITE NEW LINE
                car_count = car_count + 1
            ELSE
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE "carDatabase.txt"
    CALL cont_or_no()
ENDFUNCTION
```



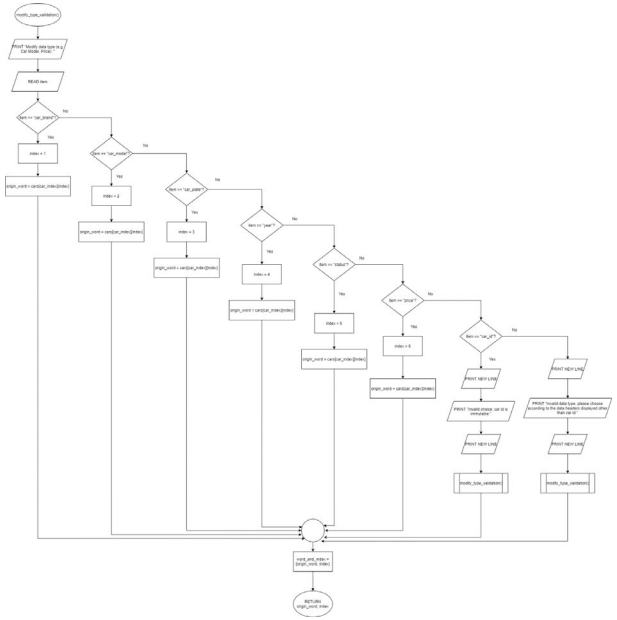
## • Modify line validation

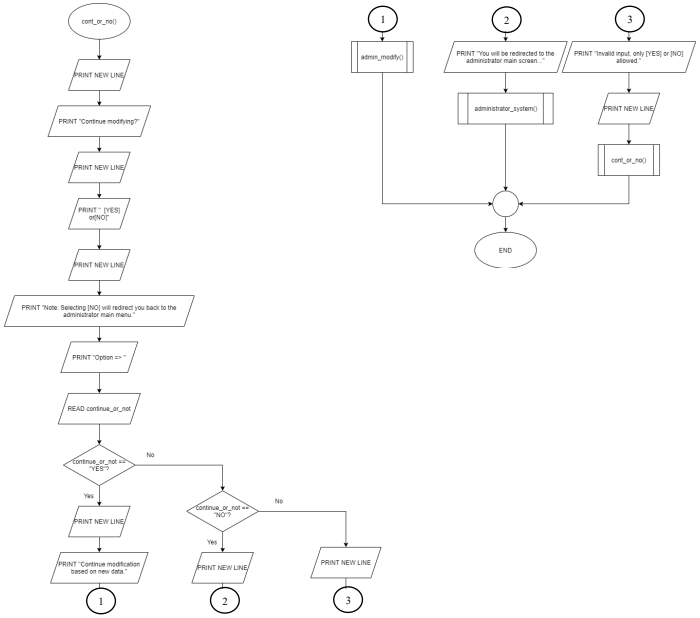
```
FUNCTION modify_line_validation()
    TRY
        PRINT NEW LINE
        PRINT "Which line of data you want to perform the modification: "
        READ data_line
        car_index_convert = CALL index_convert(data_line)
        car_index = cars_index[car_index_convert]
        IF (data_line <= 0) THEN
            PRINT "Invalid input, please try again."
            CALL modify_line_validation()
        ELSE
            RETURN car_index
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value.."
        CALL modify_line_validation()
    EXCEPT INDEX ERROR THEN
        PRINT NEW LINE
        PRINT "Car line is out of range."
        CALL modify_line_validation()
    ENDTRY
ENDFUNCTION
```




## • Modify type validation

```
FUNCTION modify_type_validation()
    PRINT "Modify data type (e.g. Car Model, Price): "
    READ item
    IF (item == "car_brand") THEN
        index = 1
        origin_word = cars[car_index][index]
    ELSE
        IF (item == "car_model") THEN
            index = 2
            origin_word = cars[car_index][index]
        ELSE
            IF (item == "car_plate") THEN
                index = 3
                origin_word = cars[car_index][index]
            ELSE
                IF (item == "year") THEN
                    index = 4
                    origin_word = cars[car_index][index]
                ELSE
                    IF (item == "status") THEN
                        index = 5
                        origin_word = cars[car_index][index]
                    ELSE
                        IF (item == "price") THEN
                            index = 6
                            origin_word = cars[car_index][index]
                        ELSE
                            IF (item == "car_id") THEN
                                PRINT NEW LINE
                                PRINT "Invalid choice, car id is
                                immutable."
                                PRINT NEW LINE
                                CALL modify_type_validation()
                            ELSE
                                PRINT NEW LINE
                                PRINT "Invalid data type, please
                                choose according to the data headers
                                displayed other than car id."
                                PRINT NEW LINE
                                CALL modify_type_validation()
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDIF
    word_and_index = [origin_word, index]
    RETURN word_and_index
ENDFUNCTION
```



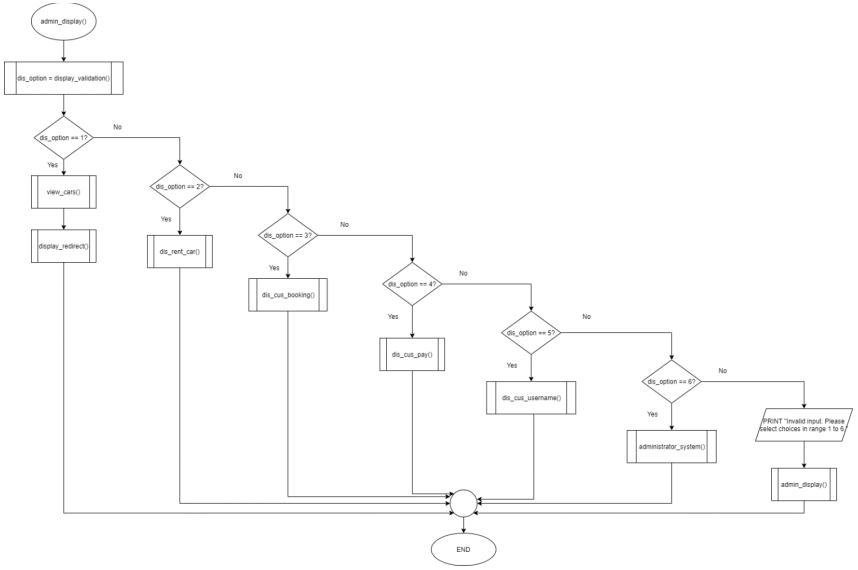


c. Display all data

- Continue modification or not

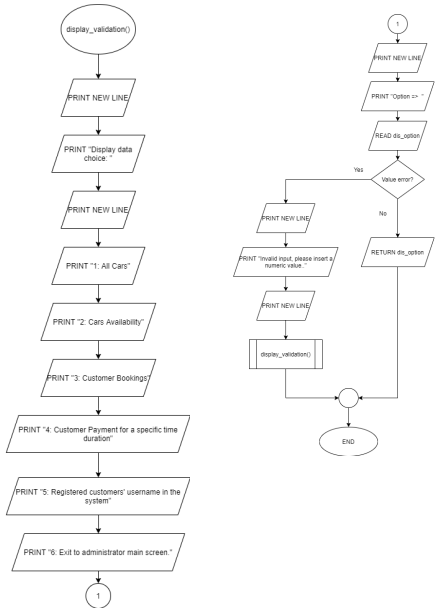
```
FUNCTION cont_or_no()
    PRINT NEW LINE
    PRINT “Continue modifying?”
    PRINT NEW LINE
    PRINT “    [YES] or [NO]”
    PRINT NEW LINE
    PRINT “Note: Selecting [NO] will redirect you back to the administrator main menu.”
    PRINT “Option => ”
    READ continue_or_not
    IF (continue_or_not == "YES") THEN
        PRINT NEW LINE
        PRINT "Continue modification based on new data."
        CALL admin_modify()
    ELSE
        IF (continue_or_not == "NO") THEN
            PRINT NEW LINE
            PRINT “You will be redirected to the administrator main screen…”
            CALL administrator_system()
        ELSE
            PRINT NEW LINE
            PRINT “Invalid input, only [YES] or [NO] allowed.”
            PRINT NEW LINE
            CALL cont_or_no()
        ENDIF
    ENDIF
ENDFUNCTION
```

```
FUNCTION admin_display()
    dis_option = CALL display_validation()
    IF (dis_option == 1) THEN
        CALL view_cars()
        CALL display_redirect()
    ELSE
        IF (dis_option == 2) THEN
            CALL dis_rent_car()
        ELSE
            IF (dis_option == 3) THEN
                CALL dis_cus_booking()
            ELSE
                IF (dis_option == 4) THEN
                    CALL dis_cus_pay()
                ELSE
                    IF (dis_option == 5) THEN
                        CALL dis_cus_username()
                    ELSE
                        IF (dis_option == 6) THEN
                            CALL administrator_system()
                        ELSE
                            PRINT “Invalid input. Please select choices in range 1 to 6.”
                            CALL admin_display()
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDIF
ENDFUNCTION
```



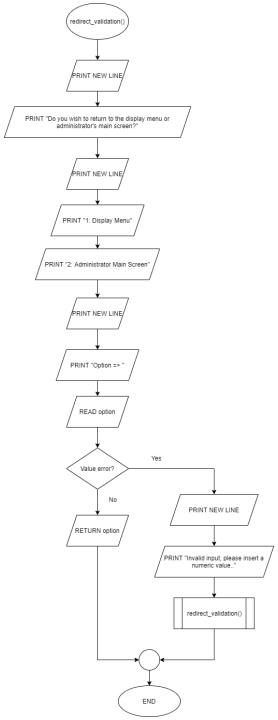

- Display validation

```
FUNCTION display_validation()
    TRY
        PRINT NEW LINE
        PRINT “Display data choice:”
        PRINT NEW LINE
        PRINT “1: All Cars”
        PRINT “2: Cars Availability”
        PRINT “3: Customer Bookings”
        PRINT “4: Customer Payment for a specific time duration”
        PRINT “5: Registered customers' username in the system”
        PRINT “6: Exit to administrator main screen.”
        PRINT NEW LINE
        PRINT “Option =>  ”
        READ dis_option
        RETURN dis_option
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT “Invalid input, please insert a numeric value..”
        PRINT NEW LINE
        CALL display_validation()
    ENDTRY
ENDFUNCTION
```



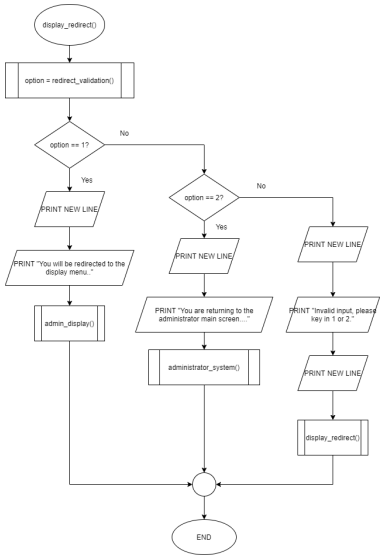

- Redirect validation

```
FUNCTION redirect_validation()
    TRY
        PRINT NEW LINE
        PRINT “Do you wish to return to the display menu or administrator's main screen?
        PRINT NEW LINE
        PRINT “1: Display Menu”
        PRINT “2: Administrator Main Screen”
        PRINT NEW LINE
        PRINT “Option =>  ”
        READ option
        RETURN option
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT “Invalid input, please insert a numeric value..”
        CALL redirect_validation()
    ENDTRY
ENDFUNCTION
```




- Display redirection

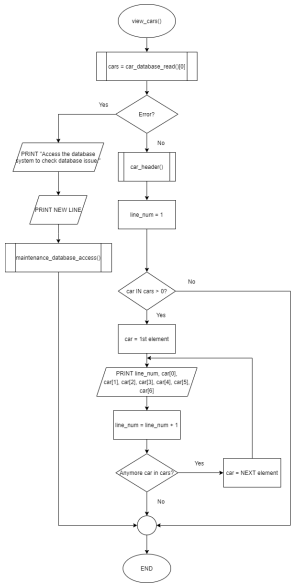
```
FUNCTION display_redirect()
    option = CALL redirect_validation()
    IF (option == 1) THEN
        PRINT NEW LINE
        PRINT "You will be redirected to the display menu..."
        CALL admin_display()
    ELSE
        IF (option == 2) THEN
            PRINT NEW LINE
            PRINT "You are returning to the administrator main screen...."
            CALL administrator_system()
        ELSE
            PRINT NEW LINE
            PRINT "Invalid input, please key in 1 or 2."
            PRINT NEW LINE
            CALL display_redirect()
        ENDIF
    ENDIF
ENDFUNCTION
```



## Display dataset

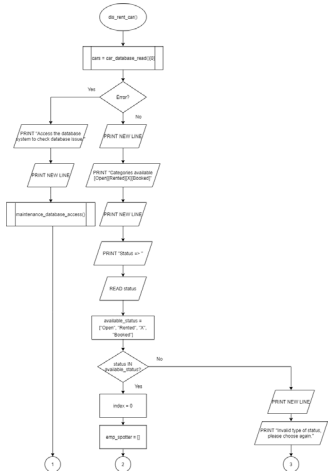
### i. View all car data

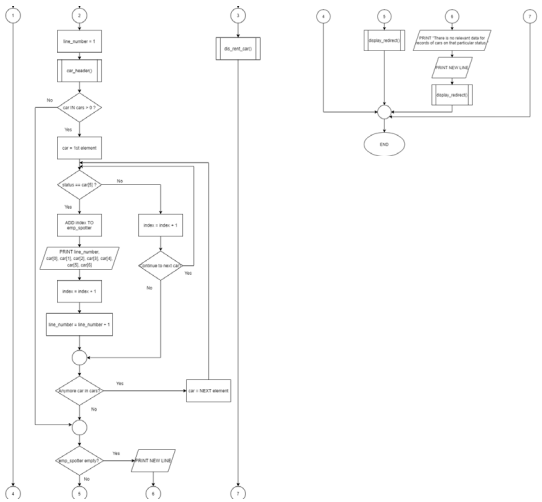
```
FUNCTION view_cars()
    TRY
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT “Access the database system to check database issue.”
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    CALL car_header()
    line_num = 1
    FOR EACH car IN cars
        PRINT line_num, car[0], car[1], car[2], car[3], car[4], car[5], car[6]
        line_num = line_num + 1
    ENDFOR
ENDFUNCTION
```




### ii. Display car with different status

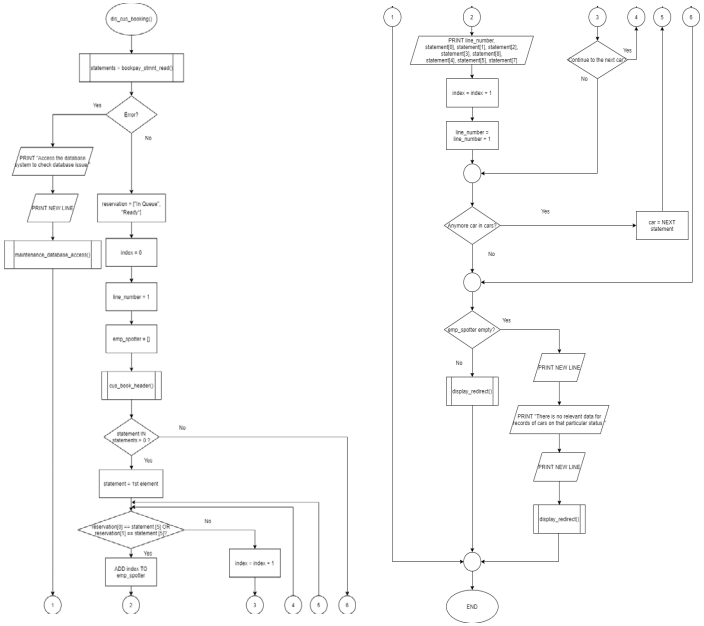
```
FUNCTION dis_rent_car()
    TRY
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT “Access the database system to check database issue.”
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    PRINT NEW LINE
    PRINT “Categories available [Open] [Rented] [X] [Booked]”
    PRINT NEW LINE
    PRINT “Status => ”
    READ status
    available_status = ["Open", "Rented", "X", "Booked"]
    IF status IN available_status THEN
        index = 0
        emp_spotter = []
        line_number = 1
        CALL car_header()
        FOR EACH car IN cars
            IF (status == car[5]) THEN
                ADD index TO emp_spotter
                PRINT line_number, car[0], car[1], car[2], car[3], car[4],
                car[5], car[6]
                index = index + 1
                line_number = line_number + 1
            ELSE
                index = index + 1
                NEXT car
            ENDIF
        ENDIF
        IF emp_spotter EMPTY THEN
            PRINT NEW LINE
            PRINT “There is no relevant data for records of cars on that particular
            status.”
            PRINT NEW LINE
            CALL display_redirect()
        ENDIF
        CALL display_redirect()
    ELSE
        PRINT NEW LINE
        PRINT “Invalid type of status, please choose again.”
        CALL dis_rent_car()
    ENDIF
ENDFUNCTION
```

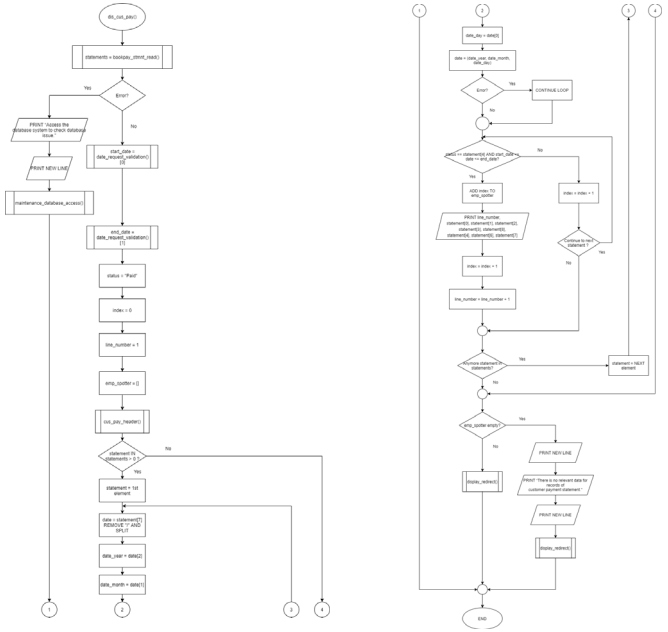



### iii. Display customer booking statement

```
FUNCTION dis_cus_booking()
    TRY
        statements = CALL bookpay_stmnt_read()
    EXCEPT ERROR THEN
        PRINT “Access the database system to check database issue.”
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    reservation = [“In Queue”, “Ready”]
    index = 0
    line_number = 1
    emp_spotter = []
    CALL cus_book_header()
    FOR EACH statement IN statements
        IF (reservation[0] == statement[5]) OR (reservation[1] == statement[5])
        THEN
            ADD index TO emp_spotter
            PRINT line_number, statement[0], statement[1], statement[2],
            statement[3], statement[8], statement[4], statement[5], statement[7]
            index = index + 1
            line_number = line_number + 1
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF emp_spotter EMPTY THEN
        PRINT NEW LINE
        PRINT “There is no relevant data for records of customer booking statement.”
        PRINT NEW LINE
        CALL display_redirect()
    ENDIF
    CALL display_redirect()
ENDFUNCTION
```



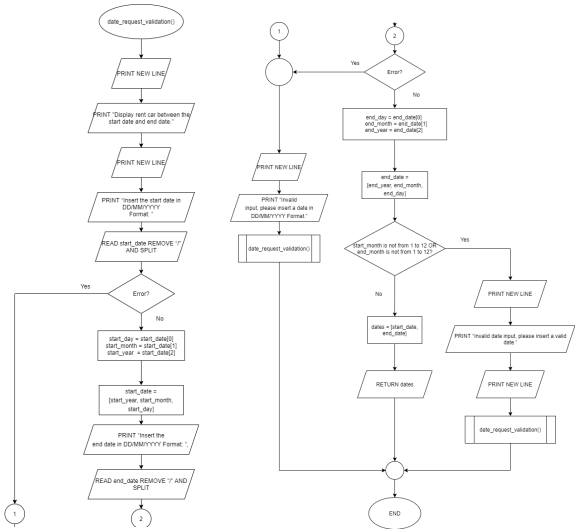




## iv. Display customer payment statement

```
FUNCTION dis_cus_pay()
    TRY
        statements = CALL bookpay_stmnt_read()
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    dates = CALL date_request_validation()
    start_date = dates[0]
    end_date = dates[1]
    status = "Paid"
    index = 0
    line_number = 1
    emp_spotter = []
    CALL cus_pay_header()
    FOR EACH statement IN statements
        TRY
            date = statement[7] REMOVE "/"AND SPLIT
            date_year = date[2]
            date_month = date[1]
            date_day = date[0]
            date = [date_year, date_month, date_day]
        EXCEPT ERROR THEN
            CONTINUE LOOP
        ENDTRY
        IF (status = statement[4]) AND (start_date <= date <= end_date) THEN
            ADD index TO emp_spotter
            PRINT line_number, statement[0], statement[1], statement[2], statement[3],
            statement[8], statement[4], statement[6], statement[7]
            index = index + 1
            line_number = line_number + 1
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF emp_spotter EMPTY THEN
        PRINT NEW LINE
        PRINT "There is no relevant data for records of customer payment statement."
        PRINT NEW LINE
        CALL display_redirect()
    ENDIF
    CALL display_redirect()
ENDFUNCTION
```

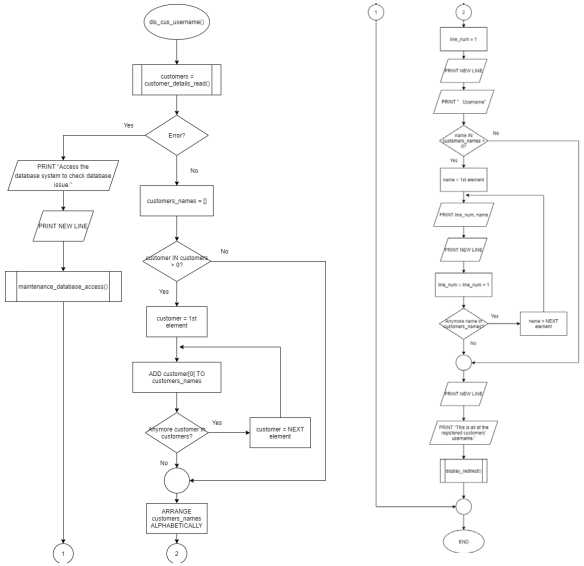
- Date request validation

```
FUNCTION date_request_validation()
    PRINT NEW LINE
    PRINT "Display rent car between the start date and end date."
    PRINT NEW LINE
    TRY
        PRINT "Insert start date in DD/MM/YYYY Format: "
        READ start_date REMOVE "/" AND SPLIT
        start_day = start_date[0]
        start_month = start_date[1]
        start_year = start_date[2]
        start_date = [start_year, start_month, start_day]
        PRINT "Insert end date in DD/MM/YYYY Format: "
        READ end_date REMOVE "/" AND SPLIT
        end_day = end_date[0]
        end_month = end_date[1]
        end_year = end_date[2]
        end_date = [end_year, end_month, end_day]
        IF (start_month IS NOT FROM 1 TO 12) OR (end_month IS NOT FROM 1 TO 12)
        THEN
            PRINT NEW LINE
            PRINT "Invalid date input, please insert a valid date."
            PRINT NEW LINE
            CALL date_request_validation()
        ELSE
            dates = [start_date, end_date]
            RETURN dates
        ENDIF
    EXCEPT ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a date in DD/MM/YYYY Format."
        CALL date_request_validation()
    ENDTRY
ENDFUNCTION
```



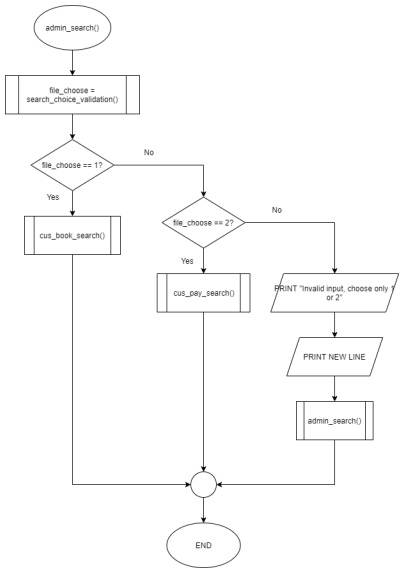
v. Display all registered customer's username

```
FUNCTION dis_cus_username()
    TRY
        customers = CALL customer_details_read()
    EXCEPT
        PRINT "Access the database system to check database issue."
        ERROR THEN
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    customers_names = []
    FOR EACH customer IN customers
        ADD customer[0] TO customers_names
    ENDFOR
    ARRANGE customers_names ALPHABETICALLY
    line_num = 1
    PRINT NEW LINE
    PRINT "      Username"
    FOR EACH name IN customers_names
        PRINT line_num, name
        PRINT NEW LINE
        line_num = line_num + 1
    ENDFOR
    PRINT NEW LINE
    PRINT "This is all of the registered customers' username."
    CALL display_redirect()
ENDFUNCTION
```



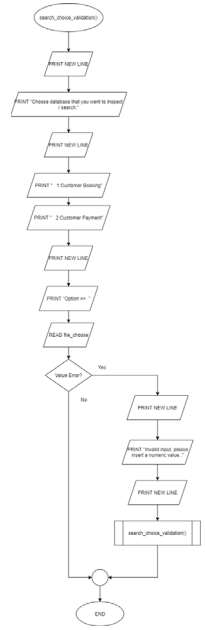

d. Search record

```
FUNCTION admin_search()
    file_choose = CALL search_choice_validation()
    IF (file_choose == 1) THEN
        CALL cus_book_search()
    ELSE
        IF (file_choose == 2) THEN
            CALL cus_pay_search()
        ELSE
            PRINT "Invalid input, choose only 1 or 2."
            PRINT NEW LINE
            CALL admin_search()
        ENDIF
    ENDIF
ENDFUNCTION
```



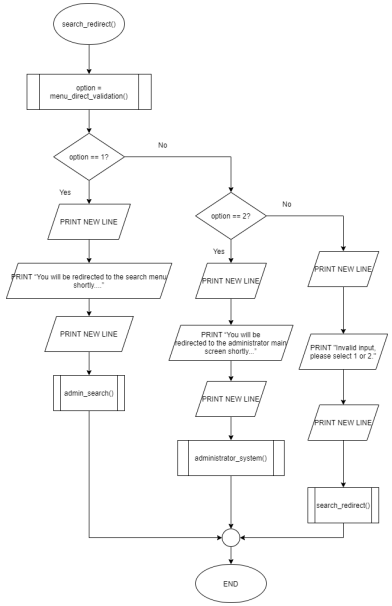

- Search choice validation

```
FUNCTION search_choice_validation()
    TRY
        PRINT NEW LINE
        PRINT "Choose database that you want to inspect / search:"
        PRINT NEW LINE
        PRINT "    1: Customer Booking"
        PRINT "    2: Customer Payment"
        PRINT NEW LINE
        PRINT "Option =>  "
        READ file_choose
        RETURN file_choose
    EXCEPT VALUE ERROR
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value.."
        PRINT NEW LINE
        CALL search_choice_validation()
    ENDTRY
ENDFUNCTION
```



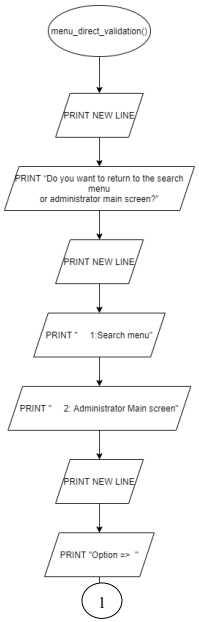

- Search redirection

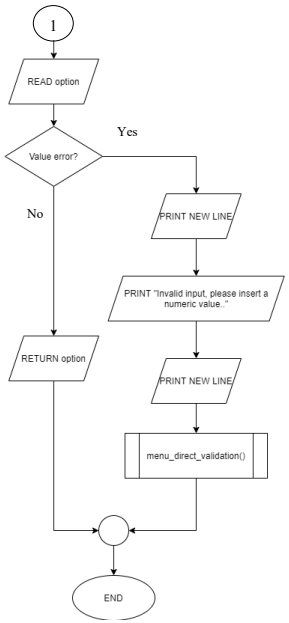
```
FUNCTION search_redirect()
    option = CALL menu_direct_validation()
    IF (option == 1) THEN
        PRINT NEW LINE
        PRINT "You will be redirected to the search menu shortly...."
        PRINT NEW LINE
        CALL admin_search()
    ELSE
        IF (option == 2) THEN
            PRINT NEW LINE
            PRINT "You will be redirected to the administrator main screen
             shortly…
            PRINT NEW LINE
            CALL administrator_system()
        ELSE
            PRINT NEW LINE
            PRINT "Invalid input, please select 1 or 2."
            PRINT NEW LINE
            CALL search_redirect()
        ENDIF
    ENDIF
ENDFUNCTION
```

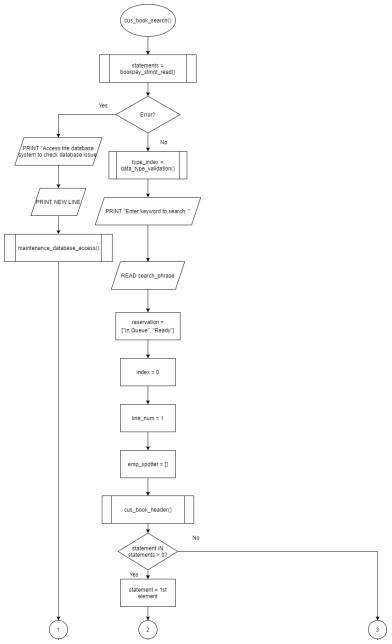


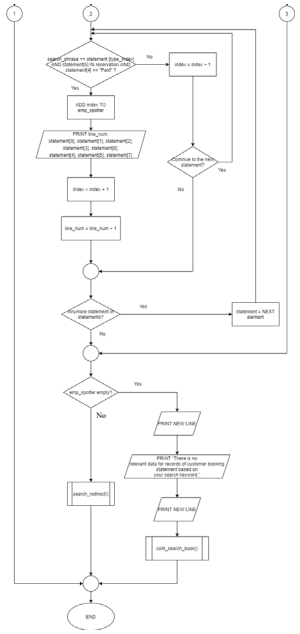
- Menu direct validation

```
FUNCTION menu_direct_validation()
    TRY
        PRINT NEW LINE
        PRINT "Do you want to return to the search menu or administrator main screen?"
        PRINT NEW LINE
        PRINT "    1: Search menu"
        PRINT "    2: Administrator Main screen"
        PRINT NEW LINE
        PRINT "Option => "
        READ option
        RETURN option
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value.."
        PRINT NEW LINE
        CALL menu_direct_validation()
    ENDTRY
ENDFUNCTION
```



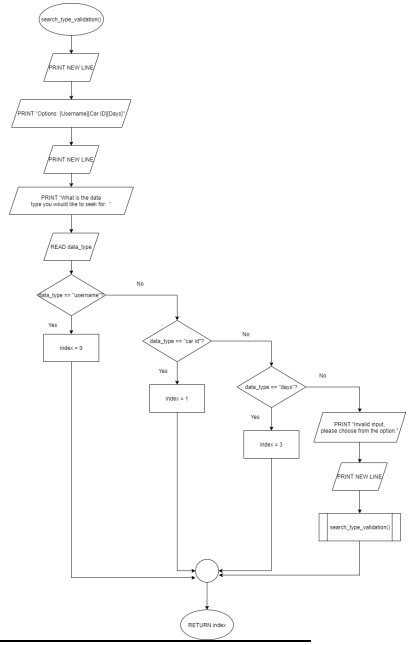

## Search Dataset

i. Customer Booking

```
FUNCTION cus_book_search()
    TRY
        statements = CALL bookpay_stmnt_read()
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    type_index = CALL search_type_validation()
    PRINT "Enter keyword to search: "
    READ search_phrase
    reservation = ["In Queue", "Ready"]
    index = 0
    line_num = 1
    emp_spotter = []
    CALL cus_book_header()
    FOR EACH statement IN statements
        IF (search_phrase == statement[type_index]) AND (statement[5] IN reservation)
        AND (statement[4] == "Paid") THEN
            ADD index TO emp_spotter
            PRINT line_num, statement[0], statement[1], statement[2], statement[3],
            statement[8], statement[4], statement[5], statement[7]
            index = index + 1
            line_num = line_num + 1
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF emp_spotter EMPTY THEN
        PRINT NEW LINE
        PRINT "There is no relevant data for records of customer booking statement
        based on your search keyword."
        PRINT NEW LINE
        CALL cont_search_book()
    ENDIF
    CALL search_redirect()
ENDFUNCTION
```

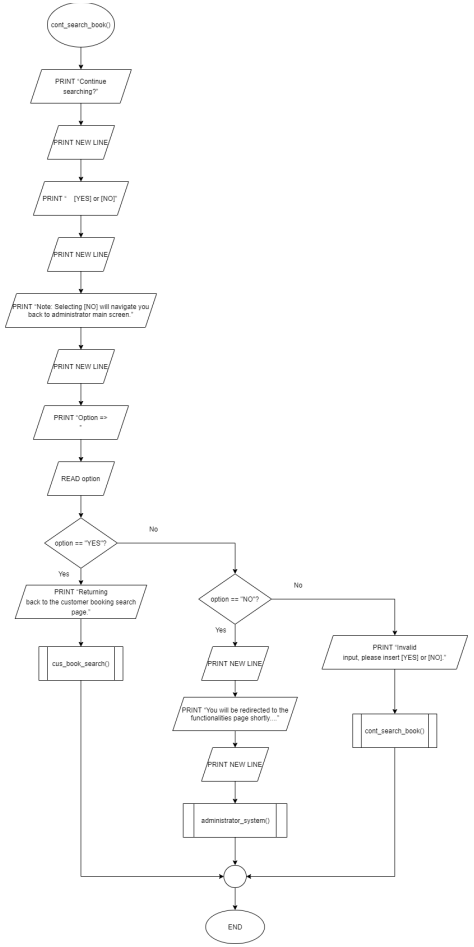
- Search type validation

```
FUNCTION search_type_validation()
    PRINT NEW LINE
    PRINT "Options: [Username] [Car ID] [Days]"
    PRINT NEW LINE
    PRINT "What is the data type you would like to seek for:  "
    READ data_type
    IF (data_type == "username") THEN
        index = 0
    ELSE
        IF (data_type == "car id") THEN
            index = 1
        ELSE
            IF (data_type == "days") THEN
                index = 3
            ELSE
                PRINT "Invalid input, please choose from the option."
                PRINT NEW LINE
                CALL search_type_validation()
            ENDIF
        ENDIF
    ENDIF
    RETURN index
ENDFUNCTION
```



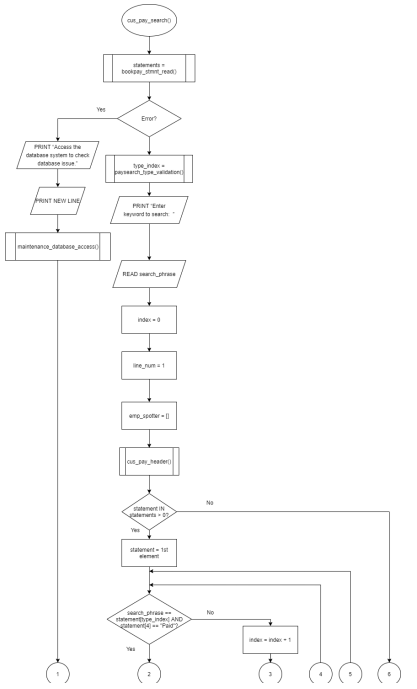
- Continue search booking?

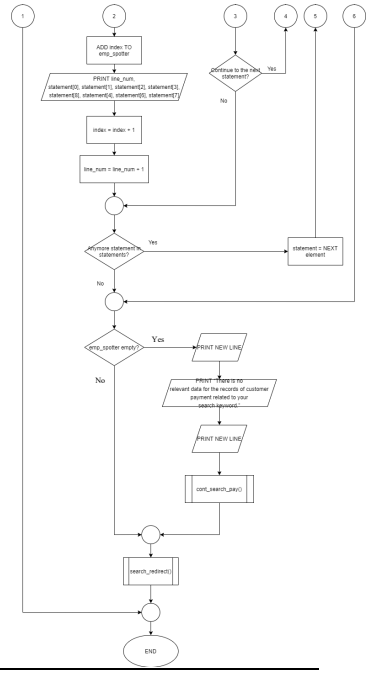
```
FUNCTION cont_search_book()
    PRINT “Continue searching?”
    PRINT NEW LINE
    PRINT “    [YES] or [NO]”
    PRINT NEW LINE
    PRINT “Note: Selecting [NO] will navigate you back to administrator main screen.”
    PRINT NEW LINE
    PRINT “Option => ”
    READ option
    IF (option == "YES") THEN
        PRINT “Returning back to the customer booking search page.”
        CALL cus_book_search()
    ELSE
        IF (option == "NO") THEN
            PRINT NEW LINE
            PRINT “You will be redirected to the functionalities page shortly....”
            PRINT NEW LINE
            CALL administrator_system()
        ELSE
            PRINT “Invalid input, please insert [YES] or [NO].”
            CALL cont_search_book()
        ENDIF
    ENDIF
ENDFUNCTION
```




- **Customer Payment**

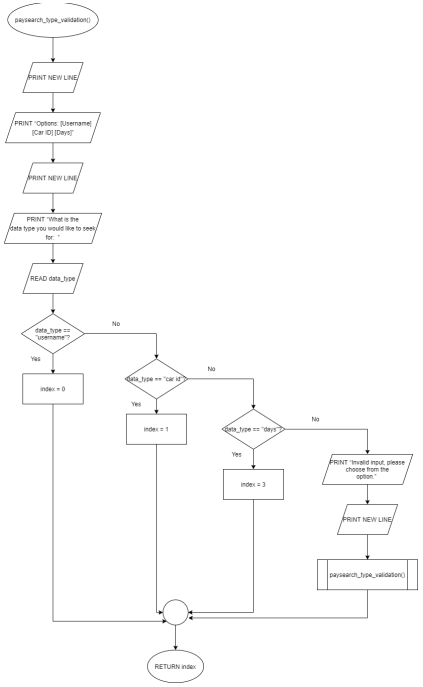
```
FUNCTION cus_pay_search()
    TRY
        statements = CALL bookpay_stmnt_read()
    EXCEPT ERROR THEN
        PRINT “Access the database system to check database issue.”
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    type_index = CALL paysearch_type_validation()
    PRINT “Enter keyword to search:  ”
    READ search_phrase
    index = 0
    line_num = 1
    emp_spotter = []
    CALL cus_pay_header()
    FOR EACH statement IN statements
        IF (search_phrase == statement[type_index]) AND (statement[4] == “Paid”) THEN
            ADD index TO emp_spotter
            PRINT line_num, statement[0], statement[1], statement[2], statement[3], statement[8], statement[4], statement[6], statement[7]
            index = index + 1
            line_num = line_num + 1
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF emp_spotter EMPTY THEN
        PRINT NEW LINE
        PRINT “There is no relevant data for the records of customer payment related to your search keyword.”
        PRINT NEW LINE
        CALL cont_search_pay()
    ENDIF
    CALL search_redirect()
ENDFUNCTION
```



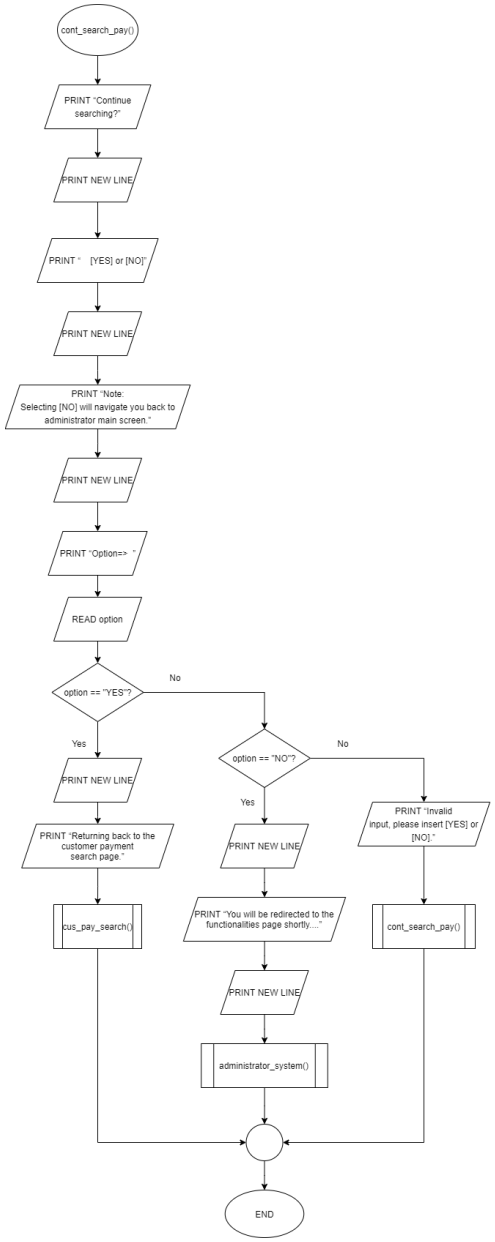
- Search payment type validation

```
FUNCTION paysearch_type_validation()
        PRINT NEW LINE
        PRINT “Options: [Username] [Car ID] [Days]”
        PRINT NEW LINE
        PRINT “What is the data type you would like to seek for:  ”
        READ data_type
        IF (data_type == “username”) THEN
                index = 0
        ELSE
                IF (data_type == “car id”) THEN
                        index = 1
                ELSE
                        IF (data_type == “days”) THEN
                                index = 3
                        ELSE
                                PRINT “Invalid input, please choose from the option.”
                                PRINT NEW LINE
                                CALL paysearch_type_validation()
                        ENDIF
                ENDIF
        ENDIF
        RETURN index
ENDFUNCTION
```



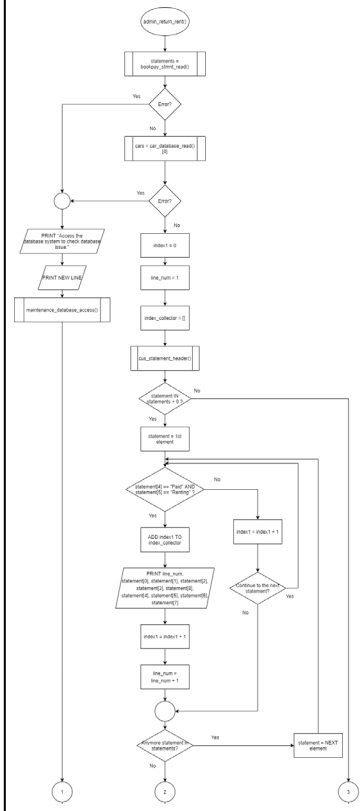

- Continue search payment

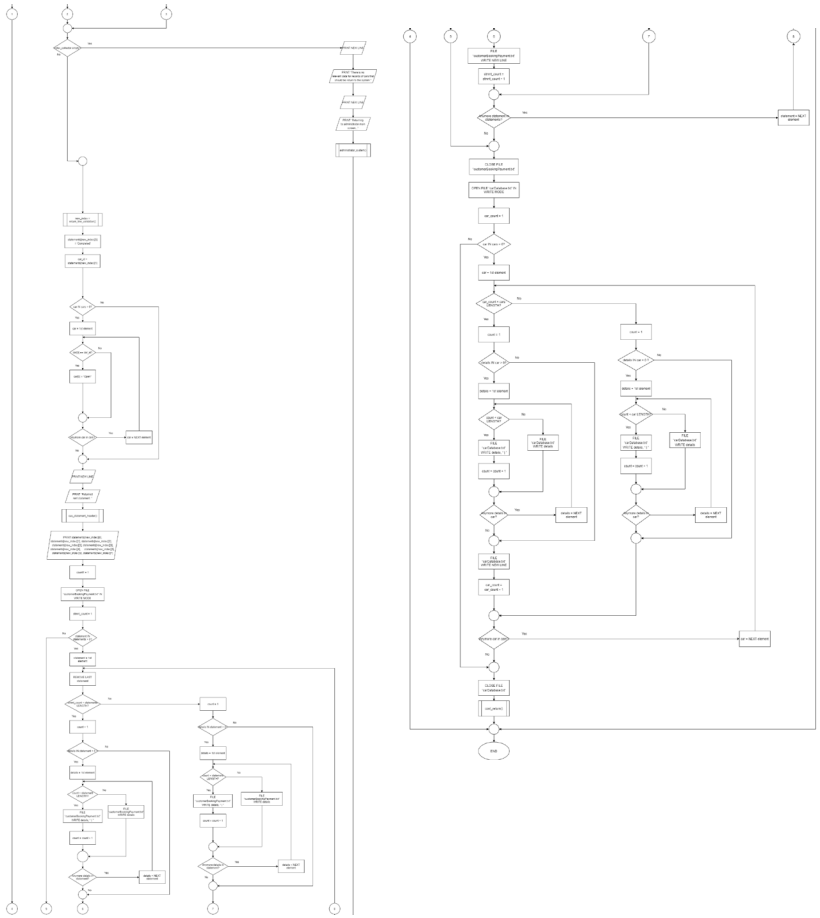
```
FUNCTION cont_search_pay()
        PRINT “Continue searching?”
        PRINT NEW LINE
        PRINT “    [YES] or [NO]”
        PRINT NEW LINE
        PRINT “Note: Selecting [NO] will navigate you back to administrator main screen.”
        PRINT NEW LINE
        PRINT “Option=>  ”
        READ option
        IF (option == “YES”) THEN
                PRINT NEW LINE
                PRINT “Returning back to the customer payment search page.”
                CALL cus_pay_search()
        ELSE
                IF (option == “NO”) THEN
                        PRINT NEW LINE
                        PRINT “You will be redirected to the functionalities page shortly....”
                        PRINT NEW LINE
                        CALL administrator_system()
                ELSE
                        PRINT “Invalid input, please insert [YES] or [NO].”
                        CALL cont_search_pay()
                ENDIF
        ENDIF
ENDFUNCTION
```



## vi. Return a rented car

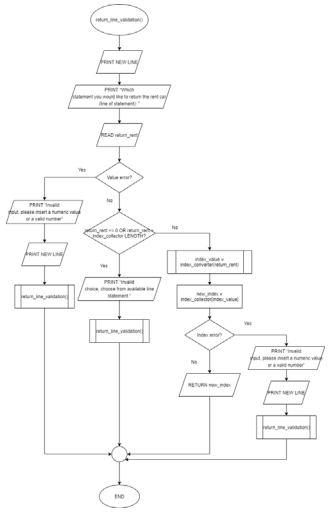
```
FUNCTION admin_return_rent()
    TRY
        statements = CALL bookpay_stmnt_read()
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    index1 = 0
    line_num = 1
    index_collector = []
    CALL cus_statement_header()
    FOR EACH statement IN statements
        IF (statement[4] == "Paid") AND (statement[5] == "Renting") THEN
            ADD index1 TO index_collector
            PRINT line_num, statement[0], statement[1], statement[2], statement[3], statement[8], statement[4], statement[5], statement[6], statement[7]
            index1 = index1 + 1
            line_num = line_num + 1
        ELSE
            index1 = index1 + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF index_collector EMPTY THEN
        PRINT NEW LINE
        PRINT "There is no relevant data for records of cars that should be return to the system."
        PRINT NEW LINE
        PRINT "Returning to administrator main screen..."
        CALL administrator_system()
    ENDIF
    new_index = CALL return_line_validation()
    statements[new_index][5] = 'Completed'
    car_id = statements[new_index][1]
    FOR EACH car IN cars
        IF (car[0] == car_id) THEN
            car[5] = "Open"
        ENDIF
    ENDFOR
    PRINT NEW LINE
    PRINT "Returned rent statement: "
    CALL cus_statement_header()
    PRINT statements[new_index][0], statements[new_index][1], statements[new_index][2], statements[new_index][3], statements[new_index][8], statements[new_index][4], statements[new_index][5], statements[new_index][6], statements[new_index][7]
    OPEN FILE "customerBookingPayment.txt" IN WRITE MODE
        stmnt_count = 1
        FOR EACH statement IN statements
            REMOVE LAST statement
            IF (stmnt_count < statements LENGTH) THEN
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE "customerBookingPayment.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "customerBookingPayment.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "customerBookingPayment.txt" WRITE NEW LINE
                stmnt_count = stmnt_count + 1
            ELSE
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE "customerBookingPayment.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "customerBookingPayment.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE "customerBookingPayment.txt"
    OPEN FILE "carDatabase.txt" IN WRITE MODE
        car_count = 1
        FOR EACH car IN cars
            IF (car_count < cars LENGTH) THEN
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "carDatabase.txt" WRITE NEW LINE
                car_count = car_count + 1
            ELSE
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE "carDatabase.txt"
    CALL cont_return()
ENDFUNCTION
```



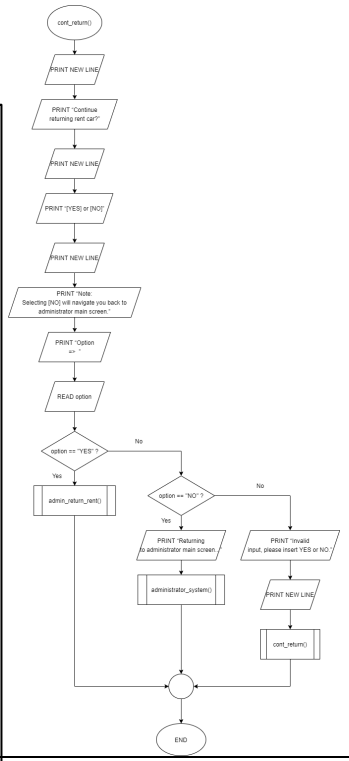
- Statement validation

```
FUNCTION return_line_validation()
    TRY
        PRINT NEW LINE
        PRINT “Which statement you would like to return the rent car (line of statement):”
        READ return_rent
        IF (return_rent <= 0) OR (return_rent > index_collector LENGTH) THEN
            PRINT “Invalid choice, choose from available line statement.”
            CALL return_line_validation()
        ELSE
            index_value = index_converter(return_rent)
            new_index = index_collector[index_value]
            RETURN new_index
        ENDIF
    EXCEPT VALUE ERROR OR INDEX ERROR THEN
        PRINT “Invalid input, please insert a numeric value Or a valid number..”
        PRINT NEW LINE
        CALL return_line_validation()
    ENDTRY
ENDFUNCTION
```



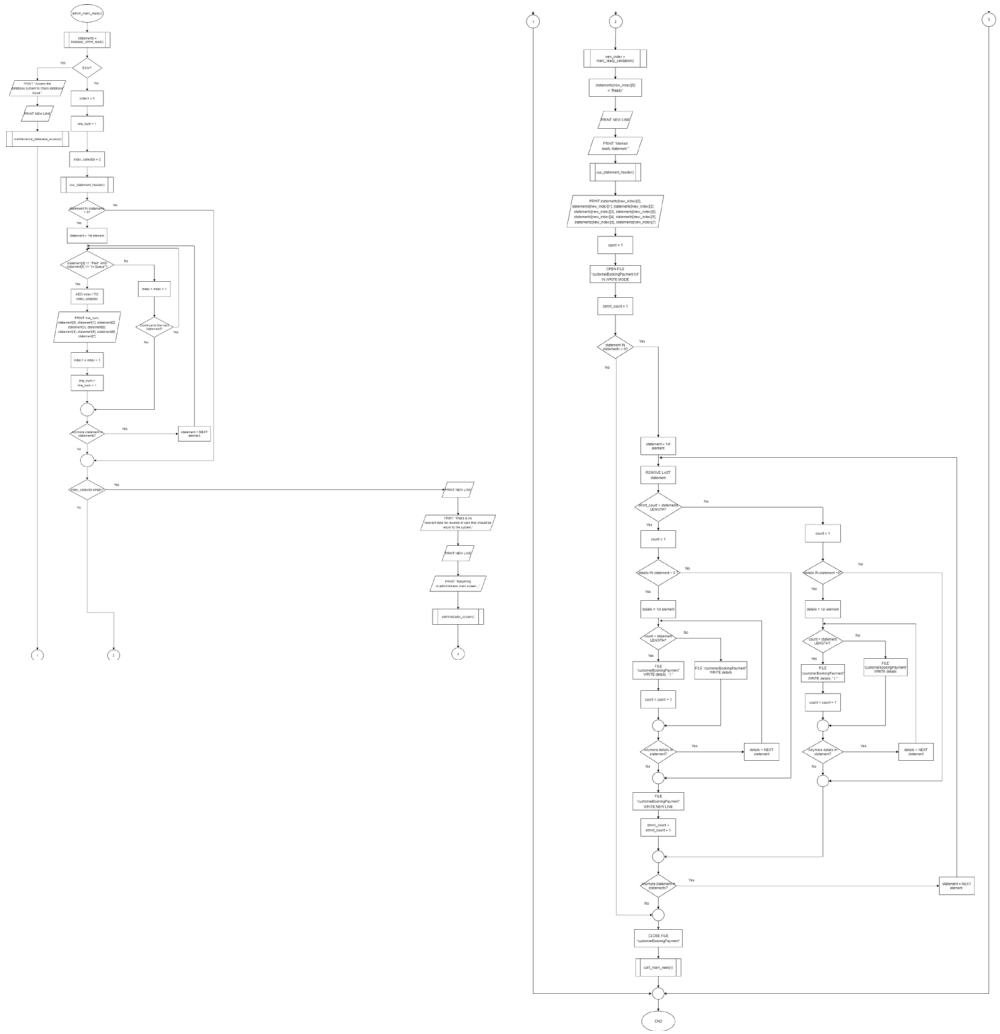

- Continue returning car

```
FUNCTION cont_return()
    PRINT NEW LINE
    PRINT “Continue returning rent car?”
    PRINT NEW LINE
    PRINT “    [YES] or [NO]”
    PRINT NEW LINE
    PRINT “Note: Selecting [NO] will navigate you back to administrator main screen.”
    PRINT “Option =>  ”
    READ option
    IF (option == “YES”) THEN
        CALL admin_return_rent()
    ELSE
        IF (option == “NO”) THEN
            PRINT “Returning to administrator main screen...”
            CALL administrator_system()
        ELSE
            PRINT “Invalid input, please insert YES or NO.”
            PRINT NEW LINE
            CALL cont_return()
        ENDIF
    ENDIF
ENDFUNCTION
```



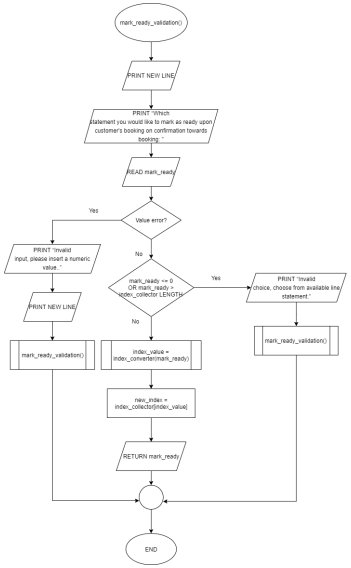

**vii.** Mark ready on a car statement

```
FUNCTION admin_mark_ready()
    TRY
        statements = CALL bookpay_stmnt_read()
    EXCEPT ERROR THEN
        PRINT “Access the database system to check database issue.”
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    index1 = 0
    line_num = 1
    index_collector = []
    CALL cus_statement_header()
    FOR EACH statement IN statements
        IF (statement[4] == “Paid”) AND (statement[5] == “In Queue”) THEN
            ADD index1 TO index_collector
            PRINT line_num, statement[0], statement[1], statement[2], statement[3], statement[8], statement[4], statement[5], statement[6], statement[7]
            index1 = index + 1
            line_num = line_num + 1
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF index_collector EMPTY THEN
        PRINT NEW LINE
        PRINT “There is no relevant data for records of cars that should be return to the system.”
        PRINT NEW LINE
        PRINT “Returning to administrator main screen...”
        CALL administrator_system()
    ENDIF
    new_index = CALL mark_ready_validation()
    statements[new_index][5] = “Ready”
    PRINT NEW LINE
    PRINT “Marked ready statement: ”
    CALL cus_statement_header()
    PRINT statements[new_index][0], statements[new_index][1], statements[new_index][2], statements[new_index][3], statements[new_index][8], statements[new_index][4], statements[new_index][5], statements[new_index][6], statements[new_index][7]
    count = 1
    OPEN FILE “customerBookingPayment.txt” IN WRITE MODE
        stmnt_count = 1
        FOR EACH statement IN statements
            REMOVE LAST statement
            IF (stmnt_count < statements LENGTH) THEN
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE “customerBookingPayment” WRITE details, “ | ”
                        count = count + 1
                    ELSE
                        FILE “customerBookingPayment” WRITE details
                    ENDIF
                ENDFOR
                FILE “customerBookingPayment” WRITE NEW LINE
                stmnt_count = stmnt_count + 1
            ELSE
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE “customerBookingPayment” WRITE details, “ | ”
                        count = count + 1
                    ELSE
                        FILE “customerBookingPayment” WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
        CLOSE FILE “customerBookingPayment”
        CALL cont_mark_ready()
ENDFUNCTION
```



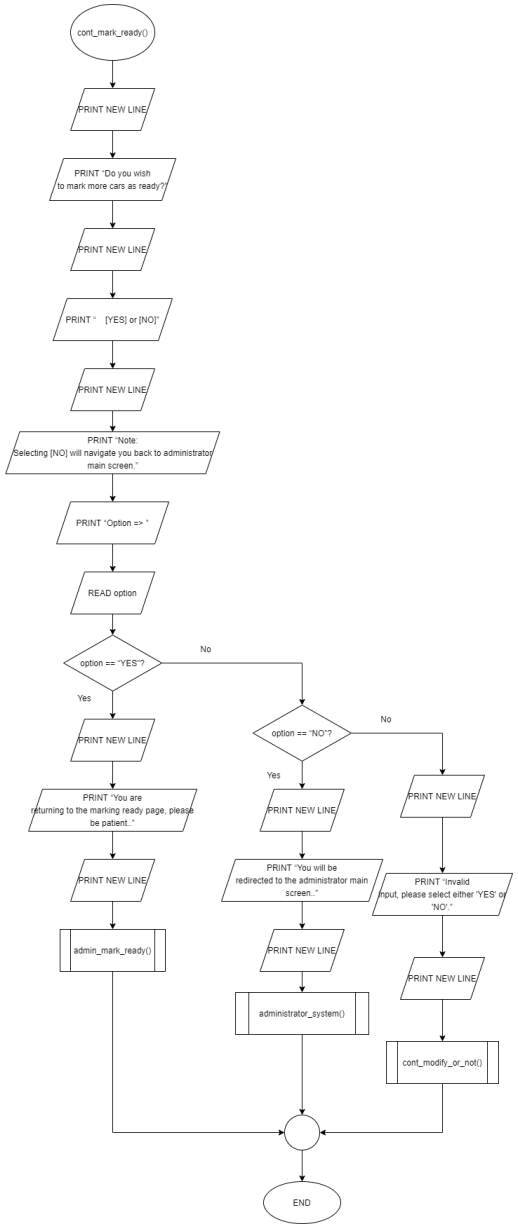
- Mark ready validation

```
FUNCTION mark_ready_validation()
    TRY
        PRINT NEW LINE
        PRINT “Which statement you would like to mark as ready upon customer's booking on confirmation towards booking: ”
        READ mark_ready
        IF (mark_ready <= 0) OR (mark_ready > index_collelctor) THEN
            PRINT “Invalid choice, choose from available line statement.”
            CALL mark_ready_validation()
        ELSE
            index_value = index_converter(mark_ready)
            new_index = index_collector[index_value]
            RETURN new_index
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT “Invalid input, please insert a numeric value..”
        PRINT NEW LINE
        CALL mark_ready_validation()
```



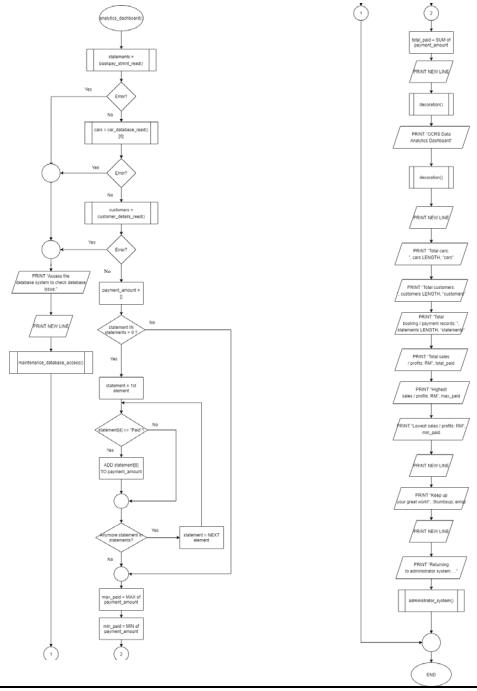

- Continue mark ready

```
FUNCTION cont_mark_ready()
    PRINT NEW LINE
    PRINT “Do you wish to mark more cars as ready?”
    PRINT NEW LINE
    PRINT “    [YES] or [NO]”
    PRINT NEW LINE
    PRINT “Note: Selecting [NO] will navigate you back to administrator main screen.”
    PRINT “Option => ”
    READ option
    IF (option == “YES”) THEN
        PRINT NEW LINE
        PRINT “You are returning to the marking ready page, please be patient..”
        PRINT NEW LINE
        CALL admin_mark_ready()
    ELSE
        IF (option == “NO”) THEN
            PRINT NEW LINE
            PRINT “You will be redirected to the administrator main screen..”
            PRINT NEW LINE
            CALL administrator_system()
        ELSE
            PRINT NEW LINE
            PRINT “Invalid input, please select either 'YES' or 'NO'.”
            PRINT NEW LINE
            CALL cont_mark_ready()
        ENDIF
    ENDIF
ENDFUNCTION
```




## Analytics Dashboard

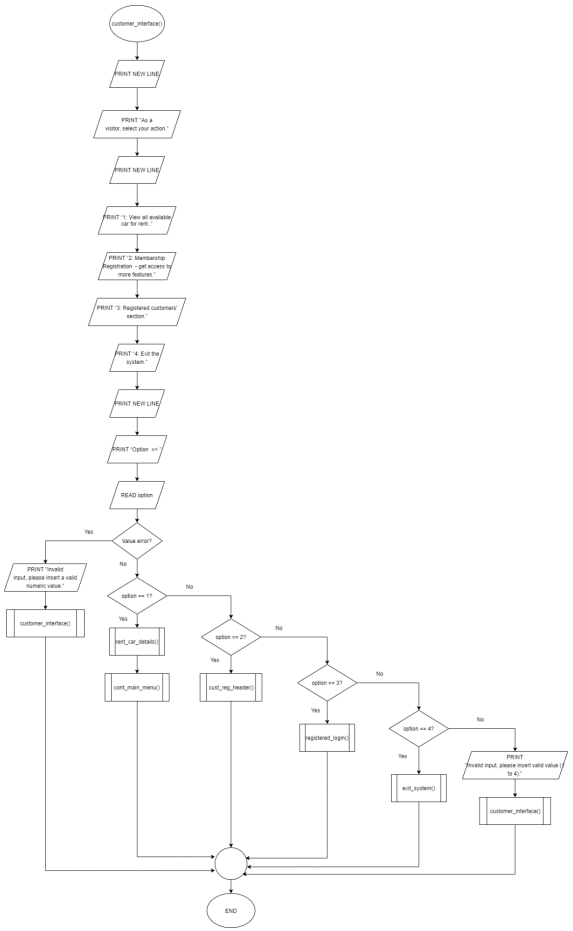
```
FUNCTION analytics_dashboard()
    TRY
        statements = CALL bookpay_stmnt_read()
        cars = CALL car_database_read()[0]
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    payment_amount = []
    FOR EACH statement IN statements
        IF (statement[4] == "Paid") THEN
            ADD statement[8] TO payment_amount
        ENDIF
    ENDFOR
    max_paid =  MAX(payment_amount)
    min_paid = MIN(payment_amount)
    total_paid = SUM(payment_amount)
    PRINT NEW LINE
    CALL decoration()
    PRINT "OCRS Data Analytics Dashboard"
    CALL decoration()
    PRINT NEW LINE
    PRINT "Total cars: ", cars LENGTH, "cars"
    PRINT "Total customers: ", customers LENGTH, "customers"
    PRINT "Total booking / payment records: ", statements LENGTH, "statements"
    PRINT "Total sales / profits: RM", total_paid
    PRINT "Highest sales / profits: RM", max_paid
    PRINT "Lowest sales / profits: RM", min_paid
    PRINT NEW LINE
    PRINT "Keep up your great work!", :thumbsup; emoji
    PRINT NEW LINE
    PRINT "Returning to administrator system…."
    CALL administrator_system()
ENDFUNCTION
```



## Section F: Customer

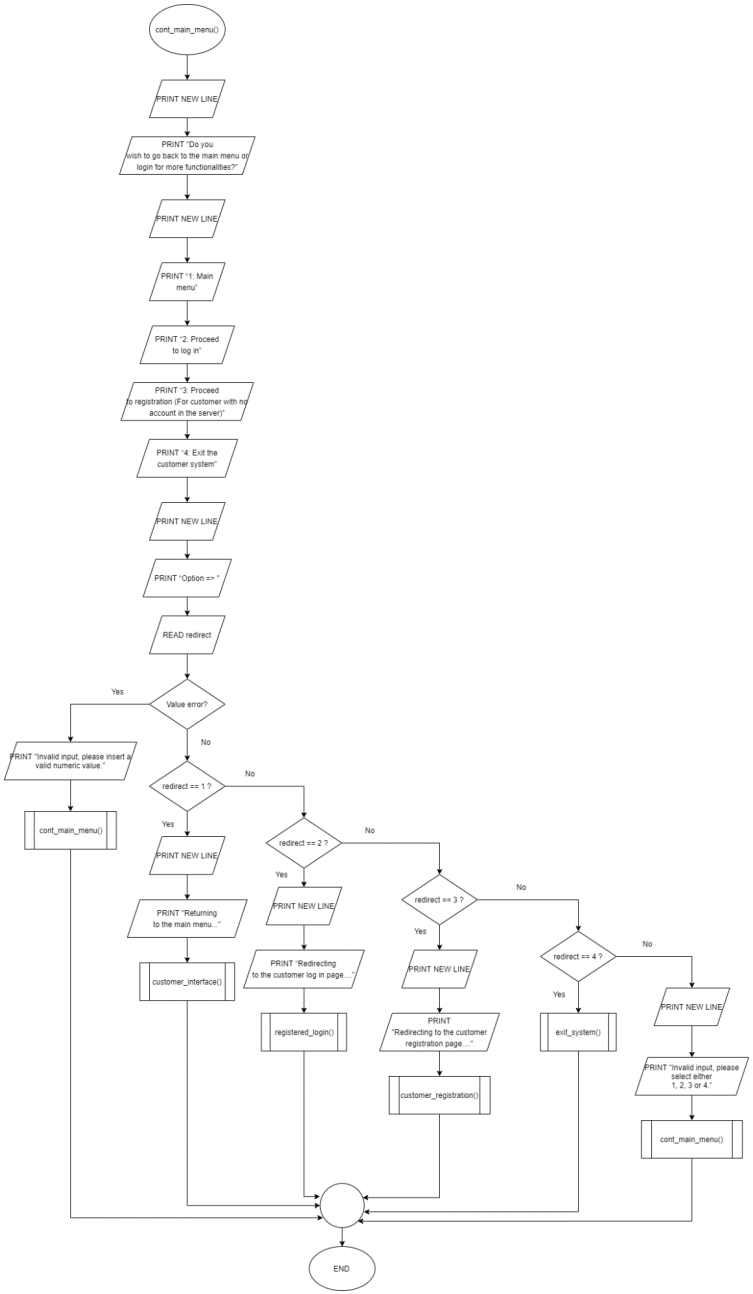
a. Customer landing page

```
FUNCTION customer_interface()
    PRINT NEW LINE
    PRINT "As a visitor, select your action."
    PRINT NEW LINE
    PRINT "1: View all available car for rent.."
    PRINT "2: Membership Registration  - get access to more features."
    PRINT "3: Registered customers' section."
    PRINT "4: Exit the system."
    PRINT NEW LINE
    TRY
        PRINT "Option  => "
        READ option
        IF (option == 1) THEN
            CALL rent_car_details()
            CALL cont_main_menu()
        ELSE
            IF (option == 2) THEN
                CALL cust_reg_header()
            ELSE
                IF (option == 3) THEN
                    CALL registered_login()
                ELSE
                    IF (option == 4) THEN
                        CALL exit_system()
                    ELSE
                        PRINT "Invalid input, please insert valid value (1 to
                            4)."
                        CALL customer_interface()
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT "Invalid input, please insert a valid numeric value."
        CALL customer_interface()
    ENDTRY
ENDFUNCTION
```



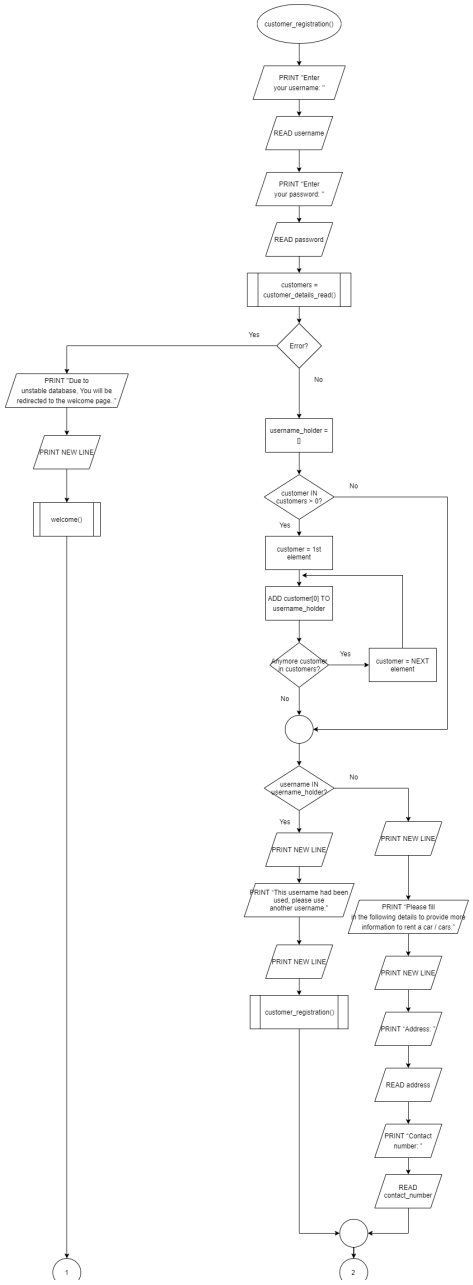

b. Redirect customer landing page, login or register

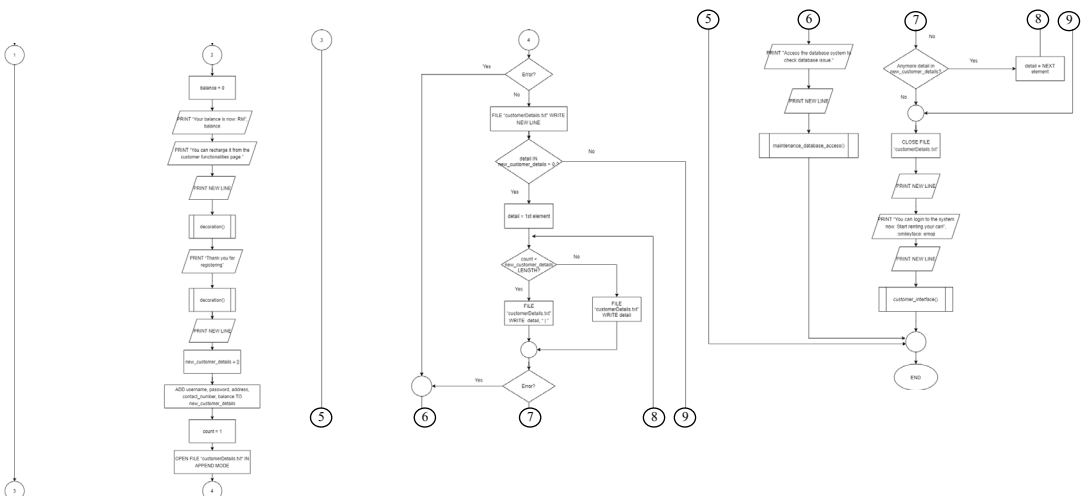
```
FUNCTION cont_main_menu()
    TRY
        PRINT NEW LINE
        PRINT "Do you wish to go back to the main menu or login for more
        functionalities?"
        PRINT NEW LINE
        PRINT "1: Main menu"
        PRINT "2: Proceed to log in"
        PRINT "3: Proceed to registration (For customer with no account in the
        server)"
        PRINT "4: Exit the customer system"
        PRINT NEW LINE
        PRINT "Option => "
        READ redirect
        IF (redirect == 1) THEN
            PRINT NEW LINE
            PRINT "Returning to the main menu..."
            CALL customer_interface()
        ELSE
            IF (redirect == 2) THEN
                PRINT NEW LINE
                PRINT "Redirecting to the customer log in page...."
                CALL registered_login()
            ELSE
                IF (redirect == 3) THEN
                    PRINT NEW LINE
                    PRINT "Redirecting to the customer registration page...."
                    CALL customer_registration()
                ELSE
                    IF (redirect == 4) THEN
                        CALL exit_system()
                    ELSE
                        PRINT NEW LINE
                        PRINT "Invalid input, please select either 1, 2, 3 or 4."
                        CALL cont_main_menu()
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT "Invalid input, please insert a valid numeric value."
        CALL cont_main_menu()
    ENDTRY
ENDFUNCTION
```

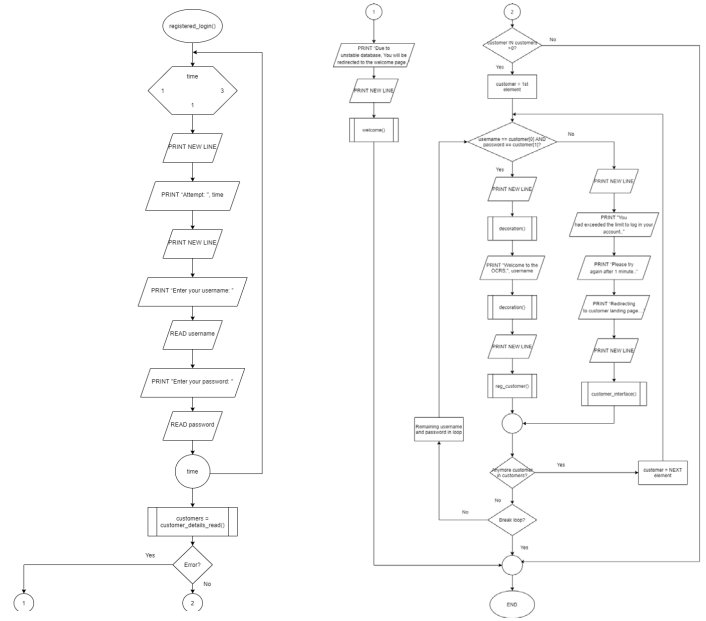



- Customer registration

```
FUNCTION customer_registration()
    PRINT "Enter your username: "
    READ username
    PRINT "Enter your password: "
    READ password
    TRY
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT "Due to unstable database, You will be redirected to the welcome
         page.."
        CALL welcome()
    ENDTRY
    username_holder = []
    FOR EACH customer IN customers
        ADD customer[0] TO username_holder
    ENDFOR
    IF username IN username_holder
        PRINT NEW LINE
        PRINT "This username had been used, please use another username."
        PRINT NEW LINE
        CALL customer_registration()
    ELSE
        PRINT NEW LINE
        PRINT "Please fill in the following details to provide more information to rent a car /
            cars."
        PRINT NEW LINE
        PRINT "Address: "
        READ address
        PRINT "Contact number: "
        READ contact_number
    ENDIF
    balance = 0
    PRINT "Your balance is now: RM", balance
    PRINT "You can recharge it from the customer functionalities page."
    PRINT NEW LINE
    CALL decoration()
    PRINT "Thank you for registering"
    CALL decoration()
    PRINT NEW LINE
    new_customer_details = []
    ADD username, password, address, contact_number, balance TO new_customer_details
    TRY
        count = 1
        OPEN FILE "customerDetails.txt" IN APPEND MODE
            FILE "customerDetails.txt" WRITE NEW LINE
            FOR EACH detail IN new_customer_details
                IF( count < new_customer_details LENGTH) THEN
                    FILE "customerDetails.txt" WRITE detail, "|"
                ELSE
                    FILE "customerDetails.txt" WRITE detail
                ENDIF
            ENDFOR
        CLOSE FILE "customerDetails.txt"
    EXCEPT ERROR THEN
        PRINT "Access the database system to check database issue."
        PRINT NEW LINE
        CALL maintenance_database_access()
    ENDTRY
    PRINT NEW LINE
    PRINT "You can login to the system now. Start renting your car!", :smileyface: emoji
    PRINT NEW LINE
    CALL customer_interface()
ENDFUNCTION
```





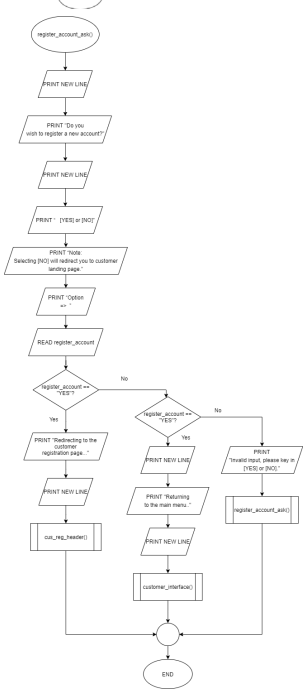


- Customer login

```
FUNCTION registered_login()
    LOOP time FROM 1 TO 3 STEP 1
        PRINT NEW LINE
        PRINT “Attempt: ", time
        PRINT NEW LINE
        PRINT “Enter your username: ”
        READ username
        PRINT "Enter your password: "
        READ password
        TRY
            customers = CALL customer_details_read()
        EXCEPT ERROR THEN
            PRINT “Due to unstable database, You will be redirected to the welcome
                page..”
            PRINT NEW LINE
            CALL welcome()
        ENDTRY
        FOR EACH customer IN customers
            IF (username == customer[0]) AND (password == customer[1]) THEN
                PRINT NEW LINE
                CALL decoration()
                PRINT " Welcome to the OCRS, ", username
                CALL decoration()
                PRINT NEW LINE
                CALL reg_customer()
                BREAK LOOP
            ENDIF
        ELSE
            PRINT NEW LINE
            PRINT "Invalid username or password, please try again..."
            CALL registration_inquiry()
        ENDFOR
    ELSE
        PRINT NEW LINE
        PRINT "You had exceeded the limit to log in your account.."
        PRINT “Please try again after 1 minute.."
        PRINT “Redirecting to customer landing page....”
        PRINT NEW LINE
        CALL customer_interface()
    ENDFOR
ENDFUNCTION
```

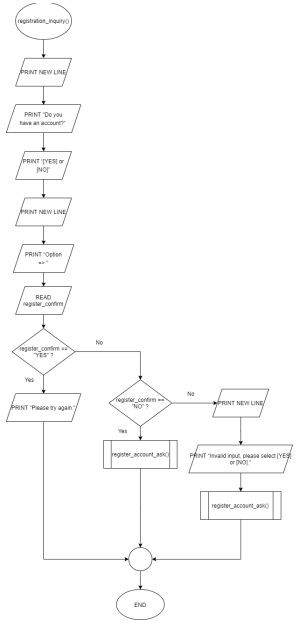
- Register account queries

```
FUNCTION register_account_ask()
    PRINT NEW LINE
    PRINT "Do you wish to register a new account?"
    PRINT NEW LINE
    PRINT “  [YES] or [NO]”
    PRINT “Note: Selecting [NO] will redirect you to customer landing page.”
    PRINT “Option =>  ”
    READ register_account
    IF (register_account == "YES") THEN
        PRINT "Redirecting to the customer registration page..."
        PRINT NEW LINE
        CALL cust_reg_header()
    ELSE
        IF (register_account == "NO") THEN
            PRINT NEW LINE
            PRINT "Returning to the main menu.."
            PRINT NEW LINE
            CALL customer_interface()
        ELSE
            PRINT "Invalid input, please key in [YES] or [NO]."
            CALL register_account_ask()
        ENDIF
    ENDIF
ENDFUNCTION
```



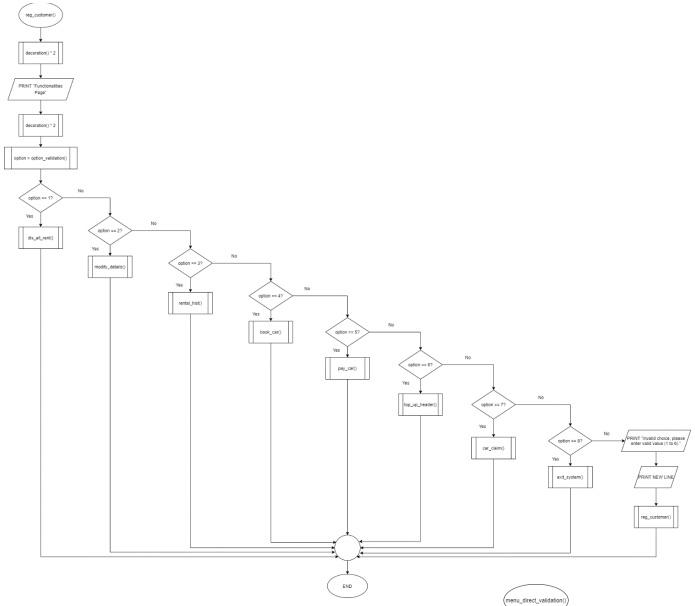
- Registration inquiry

```
FUNCTION registration_inquiry()
    PRINT NEW LINE
    PRINT "Do you have an account?"
    PRINT "[YES] or [NO]"
    PRINT NEW LINE
    PRINT "Option => "
    READ register_confirm
    IF (register_confirm == "YES") THEN
        PRINT "Please try again."
    ELSE
        IF (register_confirm == "NO") THEN
            CALL register_account_ask()
        ELSE
            PRINT NEW LINE
            PRINT "Invalid input, please select [YES] or [NO]."
            CALL register_account_ask()
        ENDIF
    ENDIF
ENDFUNCTION
```



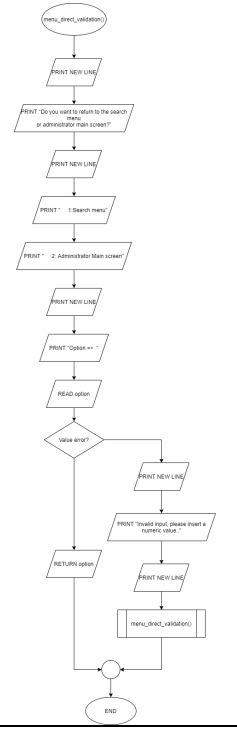

a. Customer functionalities menu

```
FUNCTION reg_customer()
    CALL decoration() * 2
    PRINT "Functionalities Page"
    CALL decoration() * 2
    option = CALL option_validation()
    IF (option == 1) THEN
        CALL dis_all_rent()
    ELSE
        IF (option == 2) THEN
            CALL modify_details()
        ELSE
            IF (option == 3) THEN
                CALL rental_hist()
            ELSE
                IF (option == 4) THEN
                    CALL book_car()
                ELSE
                    IF (option == 5) THEN
                        CALL pay_car()
                    ELSE
                        IF (option == 6) THEN
                            CALL top_up_header()
                        ELSE
                            IF (option == 7) THEN
                                CALL car_claim()
                            ELSE
                                IF (option == 8) THEN
                                    CALL exit_system()
                                ELSE
                                    PRINT "Invalid choice,
                                    please  enter valid value
                                (1 to  6)."
                                    PRINT NEW LINE
                                    CALL reg_customer()
                                ENDIF
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDIF
ENDFUNCTION
```




- Option validation

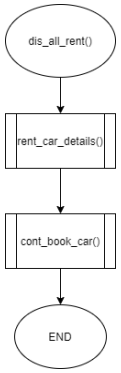
```
FUNCTION option_validation()
    TRY
        PRINT NEW LINE
        PRINT "What would you like to perform?"
        PRINT NEW LINE
        PRINT "1: View all available car for rent."
        PRINT "2: Modify personal details."
        PRINT "3: View personal rental history."
        PRINT "4: Book a car."
        PRINT "5: Pay the car that you booked earlier."
        PRINT "6: Top up your balance."
        PRINT"7: Claim car that is Ready to be rented"
        PRINT"8: Exit the portal."
        PRINT NEW LINE
        PRINT"Option => "
        READ option
        RETURN option
    EXCEPT VALUE ERROR THEN
        PRINT "Invalid input, please insert a numeric value."
        CALL option_validation()
    ENDTRY
ENDFUNCTION
```



## Section F01: Registered customers' functionalities

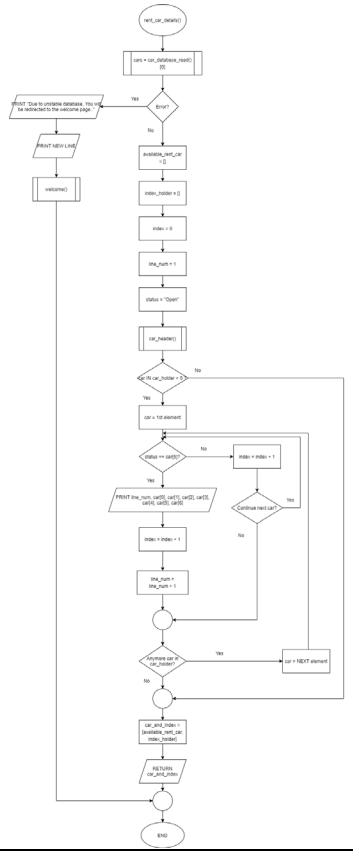
a. Display all available to rent cars

```
FUNCTION dis_all_rent()
        CALL rent_car_details()
        CALL cont_book_car()
ENDFUNCTION
```



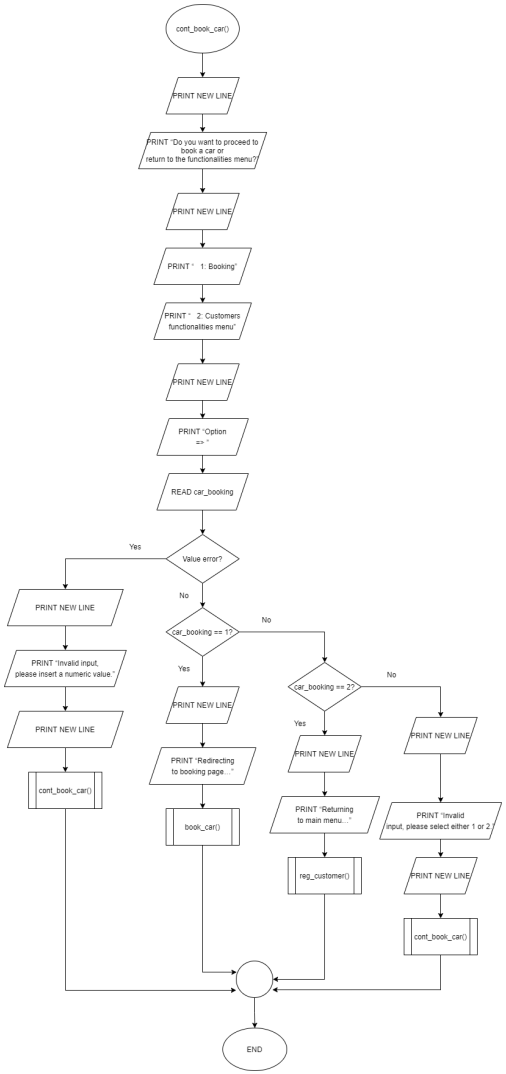

- Rent car details

```
FUNCTION rent_car_details()
    TRY
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT "Due to unstable database, You will be redirected to the welcome page.."
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    available_rent_car = []
    index_holder = []
    index = 0
    line_num = 1
    status = "Open"
    CALL car_header()
    FOR EACH car IN car_holder
        IF (status == car[5]) THEN
            PRINT line_num, car[0], car[1], car[2], car[3], car[4], car[5], car[6]
            index = index + 1
            line_num = line_num + 1
        ELSE
            index = index + 1
            NEXT car
        ENDIF
    ENDFOR
    car_and_index = [available_rent_car, index_holder]
    RETURN car_and_index
ENDFUNCTION
```



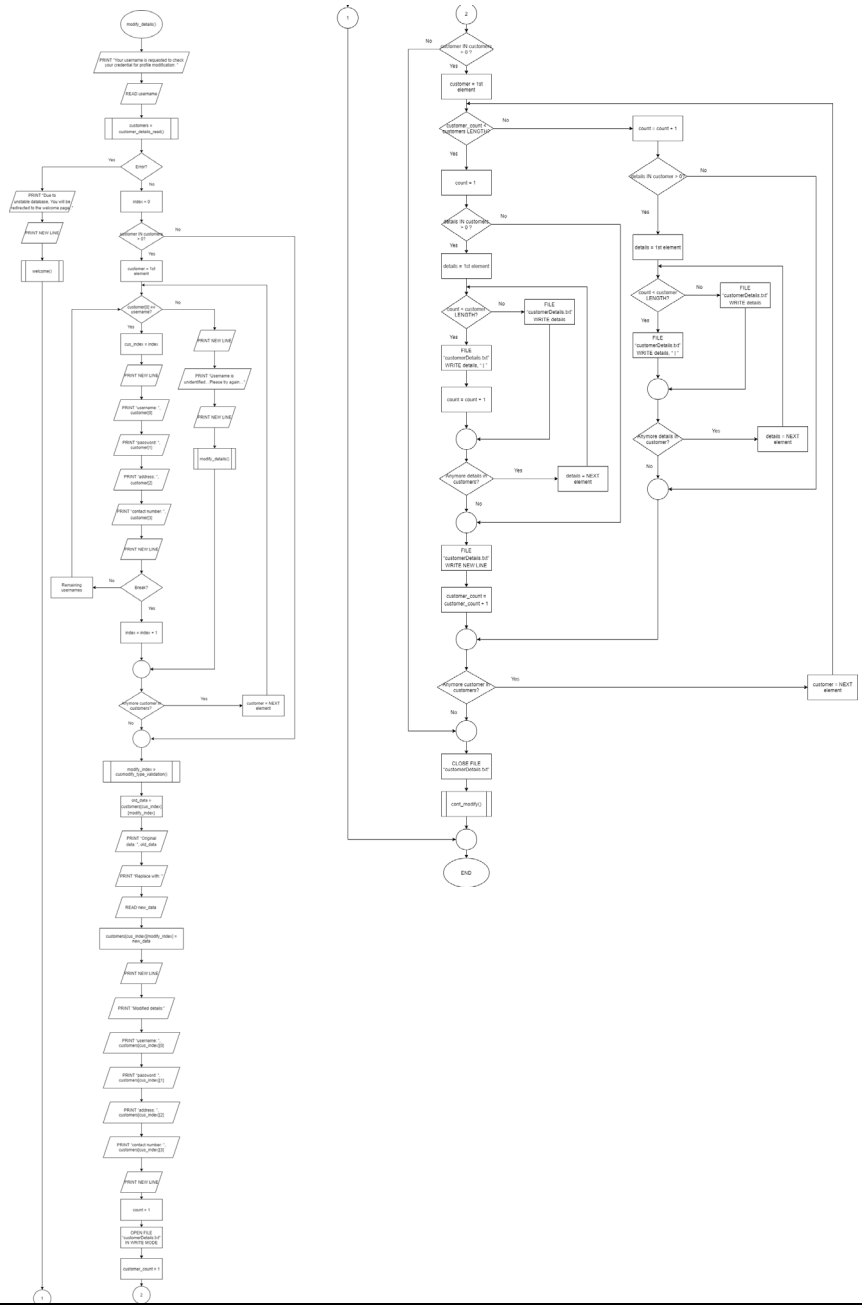
- Continue booking car?

```
FUNCTION cont_book_car()
    TRY
        PRINT NEW LINE
        PRINT "Do you want to proceed to book a car or return to the functionalities menu?"
        PRINT NEW LINE
        PRINT "  1: Booking"
        PRINT "  2: Customers functionalities menu"
        PRINT NEW LINE
        PRINT "Option => "
        READ car_booking
        IF (car_booking == 1) THEN
            PRINT NEW LINE
            PRINT "Redirecting to booking page…"
            CALL book_car()
        ELSE
            IF (car_booking == 2) THEN
                PRINT "Returning to main menu…"
                CALL reg_customer()
            ELSE
                PRINT NEW LINE
                PRINT "Invalid input, please select either 1 or 2."
                PRINT NEW LINE
                CALL cont_book_car()
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value."
        PRINT NEW LINE
        CALL cont_book_car()
    ENDTRY
ENDFUNCTION
```



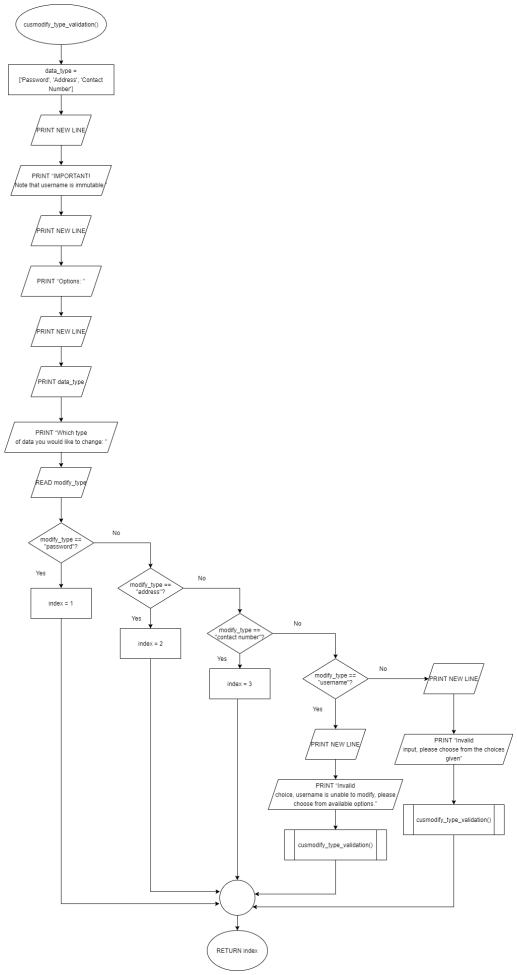
## **b.** Modify personal details

```
FUNCTION modify_details()
    PRINT "Your username is requested to check your credential for profile modification:
"
    READ username
    TRY
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT "Due to unstable database, You will be redirected to the welcome
page.."
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    index = 0
    FOR EACH customer IN customers
        IF (customer[0] == username) THEN
            cus_index = index
            PRINT NEW LINE
            PRINT "username: ", customer[0]
            PRINT "password: ", customer[1]
            PRINT "address: ", customer[2]
            PRINT "contact number: ", customer[3]
            PRINT NEW LINE
            BREAK LOOP
        index = index + 1
        ENDIF
    ELSE
        PRINT NEW LINE
        PRINT "Username is unidentified…Please try again…"
        PRINT NEW LINE
        CALL modify_details()
    ENDFOR
    modify_index = CALL cusmodify_type_validation()
    old_data = customers[cus_index][modify_index]
    PRINT "Original data: ", old_data
    PRINT "Replace with: "
     READ new_data
     customers[cus_index][modify_index] = new_data
     PRINT NEW LINE
     PRINT "Modified details:"
     PRINT "username: ", customers[cus_index][0]
     PRINT "password: ", customers[cus_index][1]
     PRINT "address: ", customers[cus_index][2]
     PRINT "contact number: " customers[cus_index][3]
     PRINT NEW LINE
     count = 1
     OPEN FILE "customerDetails.txt" IN WRITE MODE
        customer_count = 1
        FOR EACH customer IN customers
            IF (customer_count < customers LENGTH) THEN
                count = 1
                FOR EACH details IN customers
                    IF (count < customer LENGTH) THEN
                        FILE "customerDetails.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "customerDetails.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "customerDetails.txt" WRITE NEW LINE
                customer_count = customer_count + 1
            ELSE
                count = count + 1
                FOR EACH details IN customer
                    IF (count < customer LENGTH) THEN
                        FILE "customerDetails.txt" WRITE details, " | "
                    ELSE
                        FILE "customerDetails.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
        CLOSE FILE "customerDetails.txt"
        CALL cont_modify()
ENDFUNCTION
```



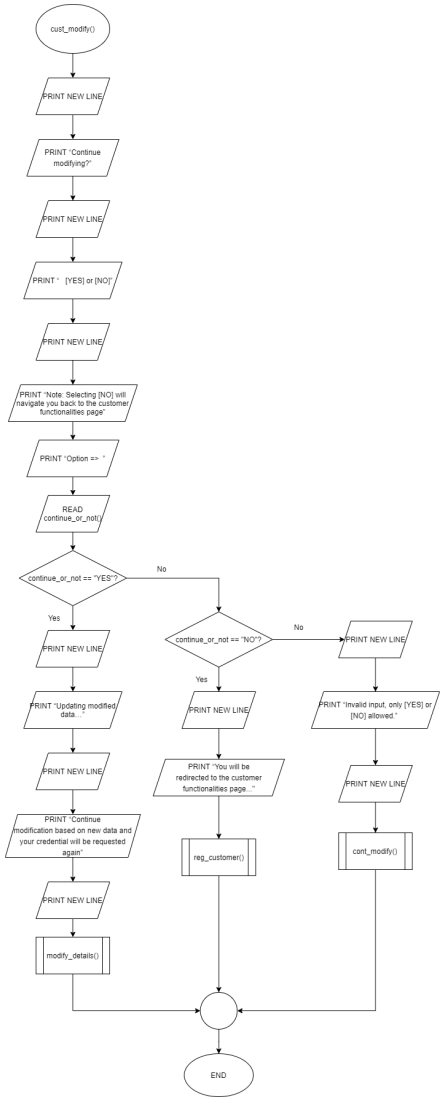
- Customer modify type validation

```
FUNCTION cusmodify_type_validation()
    data_type = ['Password', 'Address', 'Contact Number']
    PRINT NEW LINE
    PRINT “IMPORTANT! Note that username is immutable.”
    PRINT NEW LINE
    PRINT “Options: ”
    PRINT NEW LINE
    PRINT data_type
    PRINT “Which type of data you would like to change: ”
    READ modify_type
    IF (modify_type == “password”) THEN
        index = 1
    ELSE
        IF (modify_type == “address”) THEN
            index = 2
        ELSE
            IF (modify_type == “contact number”) THEN
                index = 3
            ELSE
                IF (modify_type == “username”) THEN
                    PRINT NEW LINE
                    PRINT “Invalid choice, username is unable to modify, please
                    choose from available options.”
                    CALL cusmodify_type_validation()
                ELSE
                    PRINT NEW LINE
                    PRINT “Invalid input, please choose from the choices given”
                    CALL cusmodify_type_validation()
                ENDIF
            ENDIF
        ENDIF
    ENDIF
    RETURN index
ENDFUNCTION
```



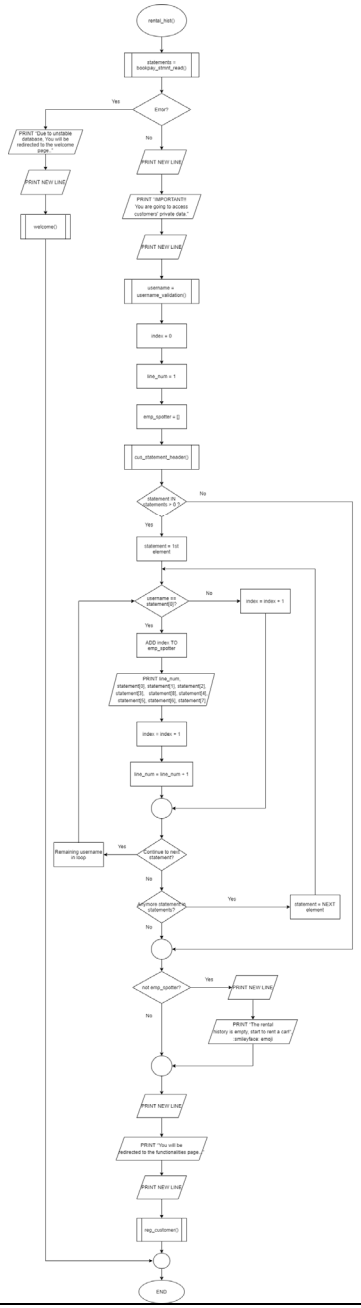

- Continue modify?

```
FUNCTION cont_modify()
    PRINT NEW LINE
    PRINT “Continue modifying?”
    PRINT NEW LINE
    PRINT “   [YES] or [NO]”
    PRINT NEW LINE
    PRINT “Note: Selecting [NO] will navigate you back to the customer functionalities page”
    PRINT “Option => ”
    READ continue_or_not
    IF (continue_or_not == “YES”) THEN
        PRINT NEW LINE
        PRINT “Updating modified data…”
        PRINT NEW LINE
        PRINT “Continue modification based on new data and your credential will be
            requested again”
        PRINT NEW LINE
        CALL modify_details()
    ELSE
        IF (continue_or_not == “NO”) THEN
            PRINT NEW LINE
            PRINT “You will be redirected to the customer functionalities page...”
            CALL reg_customer()
        ELSE
            PRINT NEW LINE
            PRINT “Invalid input, only [YES] or [NO] allowed.”
            PRINT NEW LINE
            CALL cont_modify()
        ENDIF
    ENDIF
ENDFUNCTION
```



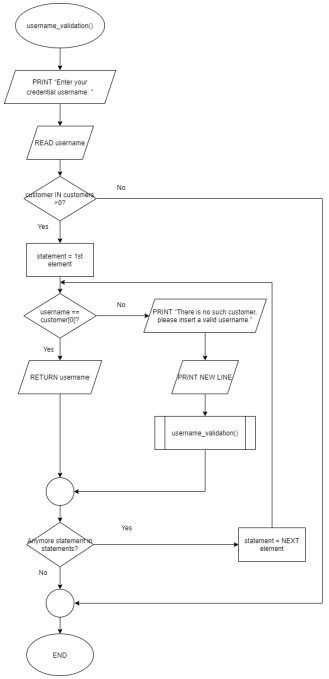
## b. View personal rental history

```
FUNCTION rental_hist()
    TRY
        statements = CALL bookpay_stmnt_read()
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT “Due to unstable database, You will be redirected to the welcome page..”
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    PRINT NEW LINE
    PRINT “IMPORTANT!! You are going to access customers' private data.”
    PRINT NEW LINE
    username = CALL username_validation()
    index = 0
    line_num = 1
    emp_spotter = []
    CALL cus_statement_header()
    FOR EACH customer IN customers
        IF (username == customer[0]) THEN
            ADD index TO emp_spotter
            PRINT line_num, statement[0], statement[1], statement[2], statement[3],
            statement[8], statement[4], statement[5], statement[6], statement[7]
            index = index + 1
            line_num = line_num + 1
            NEXT statement
        ELSE
            index = index + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF NOT emp_spotter THEN
        PRINT NEW LINE
        PRINT “The rental history is empty, start to rent a car!”, :smileyface: emoji
    ENDIF
    PRINT NEW LINE
    PRINT “You will be redirected to the functionalities page...”
    PRINT NEW LINE
    CALL reg_customer()
ENDFUNCTION
```



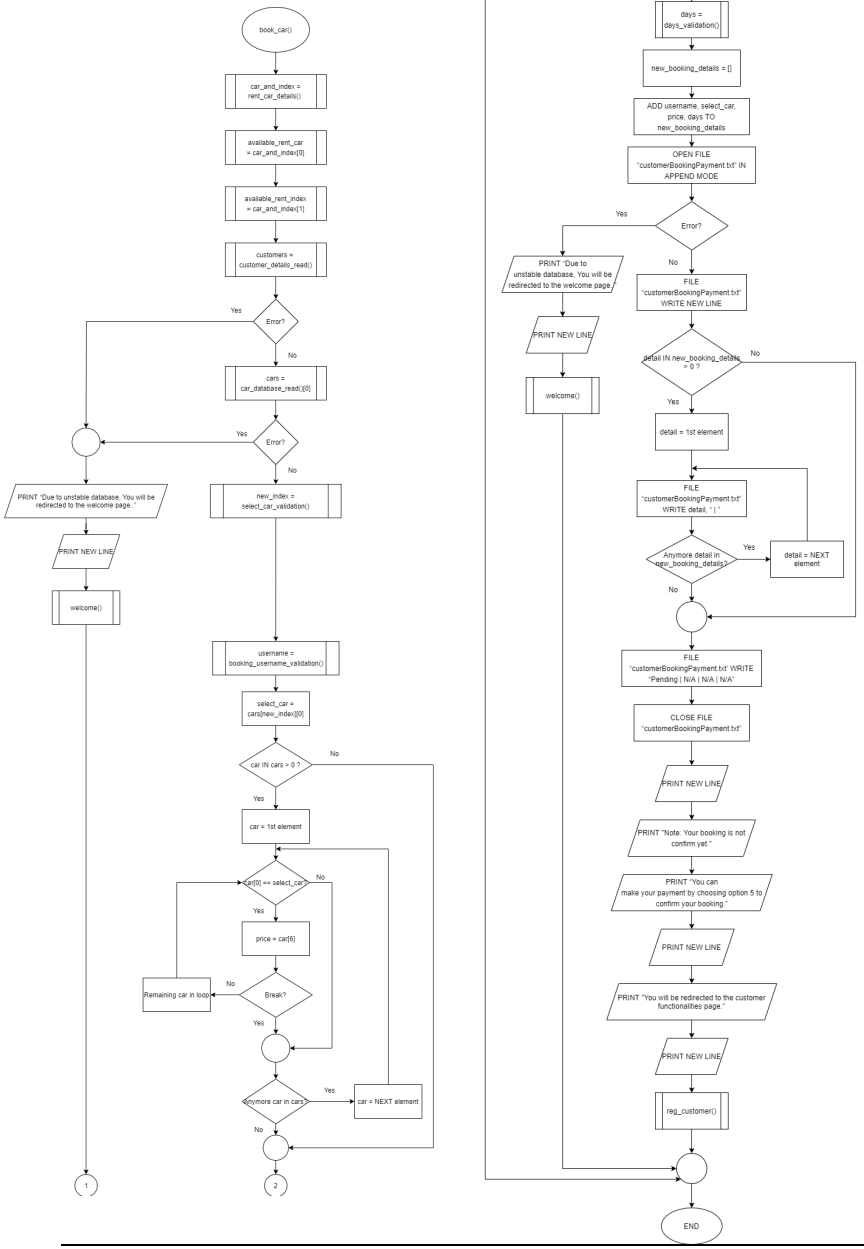

## • Username validation

```
FUNCTION username_validation()
    PRINT “Enter your credential username: ”
    READ username
    FOR EACH customer IN customers
        IF (username == customer[0]) THEN
            RETURN username
        ENDIF
    ELSE
        PRINT “There is no such customer, please insert a valid username.”
        PRINT NEW LINE
        CALL username_validation()
    ENDFOR
ENDFUNCTION
```



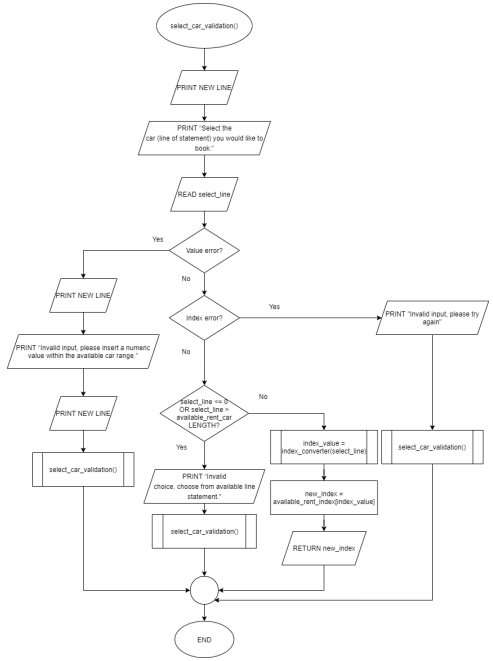

## c. Select and book a car for a specific duration

```
FUNCTION book_car()
    car_and_index = CALL rent_car_details()
    available_rent_car = car_and_index[0]
    available_rent_index = car_and_index[1]
    TRY
        customers = CALL customer_details_read()
        cars = CALL car_database_read()[0]
    EXCEPT ERROR THEN
        PRINT “Due to unstable database, You will be redirected to the welcome page..”
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    new_index = select_car_validation()
    username = CALL booking_username_validation()
    select_car = cars[new_index][0]
    FOR EACH car IN cars
        IF (car[0] == select_car) THEN
            price = car[6]
            BREAK LOOP
        ENDIF
    ENDFOR
    days = CALL days_validation()
    new_booking_details = []
    ADD username, select_car, price, days TO new_booking_details
    TRY
        OPEN FILE “customerBookingPayment.txt” IN APPEND MODE
            FILE “customerBookingPayment.txt” WRITE NEW LINE
            FOR EACH detail IN new_booking_details
                FILE “customerBookingPayment.txt” WRITE detail, “ | ”
            ENDFOR
            FILE “customerBookingPayment.txt’ WRITE “Pending | N/A | N/A | N/A”
        CLOSE FILE “customerBookingPayment.txt”
    EXCEPT ERROR THEN
        PRINT “Due to unstable database, You will be redirected to the welcome page..”
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    PRINT NEW LINE
    PRINT "Your booking is requested..."
    PRINT NEW LINE
    PRINT “Note: Your booking is not confirm yet."
    PRINT "You can make your payment by choosing option 5 to confirm your booking."
    PRINT NEW LINE
    PRINT “You will be redirected to the customer functionalities page.”
    PRINT NEW LINE
    CALL reg_customer()
ENDFUNCTION
```



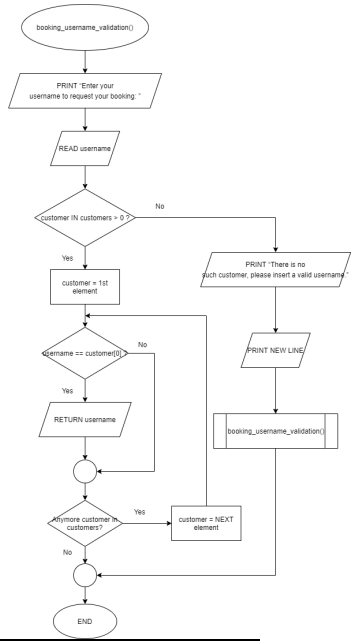

- Select car validation

```
FUNCTION select_car_validation()
    TRY
        PRINT NEW LINE
        PRINT "Select the car (line of statement) you would like to book: "
        READ select_line
        IF (select_line <= 0) OR (select_line > available_rent_car LENGTH) THEN
            PRINT "Invalid choice, choose from available line statement."
            CALL select_car_validation()
        ELSE
            index_value = index_converter(select_line)
            new_index = available_rent_index[index_value]
            RETURN new_index
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT "Invalid input, please insert a numeric value within the available car range."
        PRINT NEW LINE
        RETURN select_car_validation()
    EXCEPT INDEX ERROR THEN
        PRINT "Invalid input, please try again."
        CALL select_car_validation()
    ENDTRY
ENDFUNTION
```



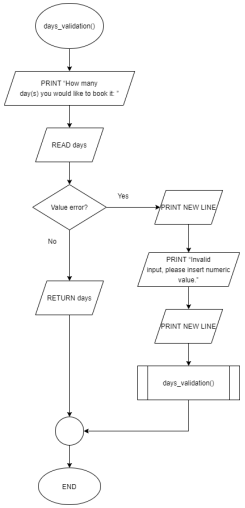

- Booking Username Validation

```
FUNCTION booking_username_validation()
    PRINT "Enter your username to request your booking: "
    READ username
    FOR EACH customer IN customers
        IF (username == customer[0]) THEN
            RETURN username
        ENDIF
    ELSE
        PRINT "There is no such customer, please insert a valid username."
        PRINT NEW LINE
        CALL booking_username_validation()
    ENDFOR
ENDFUNCTION
```



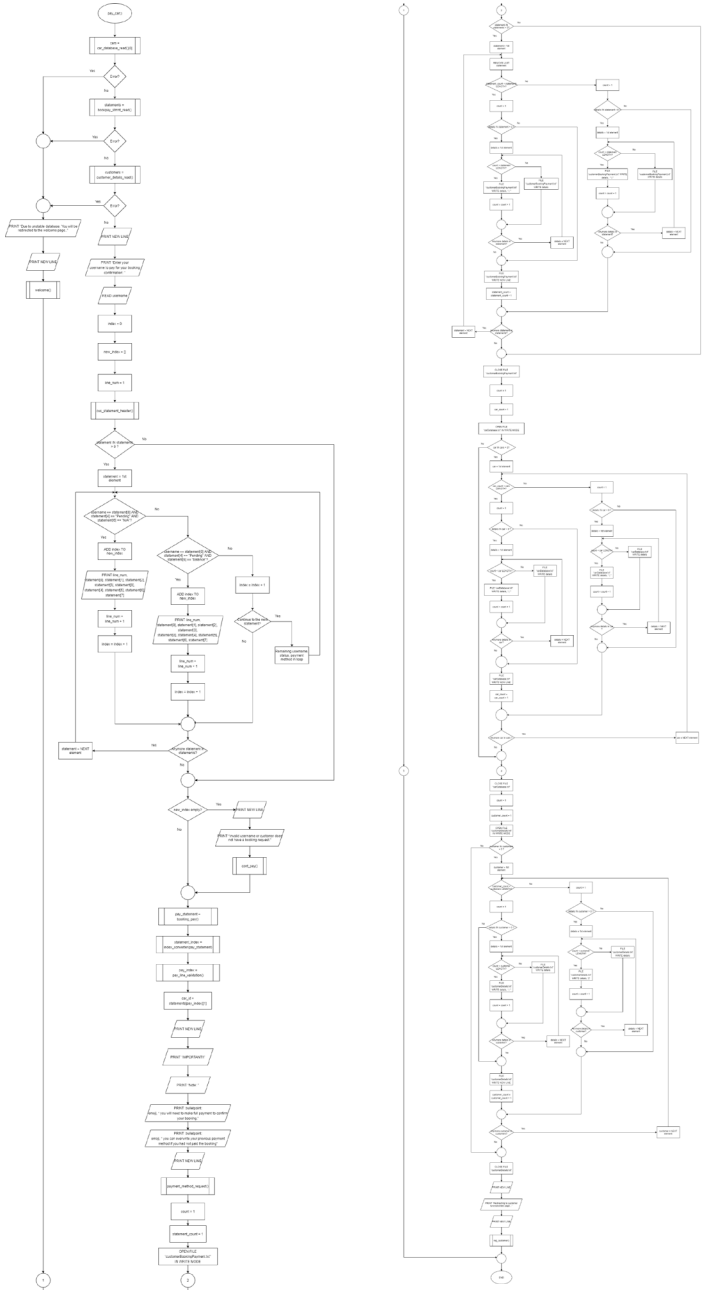
- Days validation

```
FUNCTION days_validation()
    TRY
        PRINT “How many day(s) you would like to book it: ”
        READ days
        RETURN days
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT “Invalid input, please insert numeric value.”
        PRINT NEW LINE
        CALL days_validation()
    ENDTRY
ENDFUNCTION
```



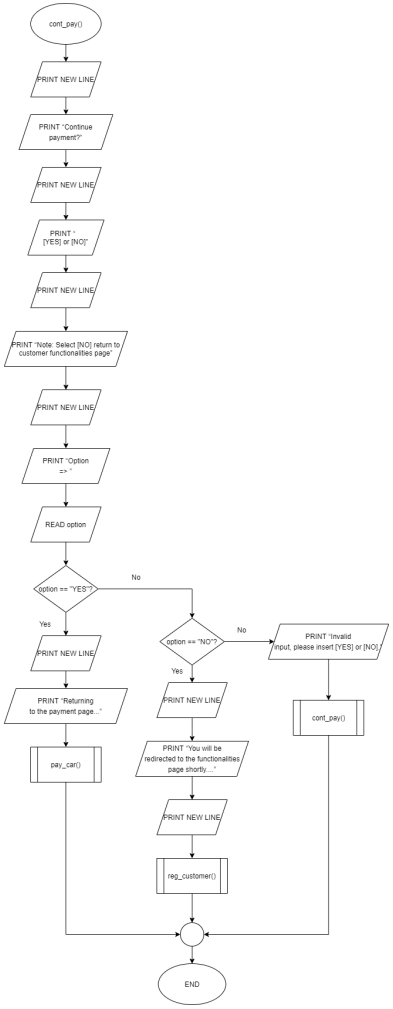

c. Do payment to confirm booking

```
FUNCTION pay_car()
    TRY
        cars = CALL car_database_read()[0]
        statements = CALL bookpay_stmnt_read()
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT “Due to unstable database, You[will be redirected to the welcome page..”
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    PRINT NEW LINE
    PRINT “Enter your username to pay for your booking confirmation: ”
    READ username
    index = 0
    new_index = []
    line_num = 1
    CALL cus_statement_header()
    FOR EACH statement IN statements
        IF (username == statement[0]) AND (statement[4] == “Pending”) AND
        (statement[6] == “N/A”) THEN
            ADD index TO new_index
            PRINT line_num, statement[0], statement[1], statement[2], statement[3],
            statement[8], statement[4], statement[5], statement[6], statement[7]
            line_num = line_num + 1
            index = index + 1
        ELSE
            IF (username == statement[0]) AND (statement[4] == “Pending”) AND
            (statement[6] == “balance”) THEN
                ADD index TO new_index
                PRINT line_num, statement[0], statement[1], statement[2],
                statement[3], statement[8], statement[4], statement[5], statement[6],
                statement[7]
                line_num = line_num + 1
                index = index + 1
            ELSE
                index = index + 1
                NEXT statement
            ENDIF
        ENDIF
    ENDFOR
    IF new_index EMPTY THEN
        PRINT NEW LINE
        PRINT “Invalid username or customer does not have a booking request.”
        CALL cont_pay()
    ENDIF
    pay_index = CALL pay_line_validation()
    car_id = statements[pay_index][1]
    PRINT NEW LINE
    PRINT “IMPORTANT!!”
    PRINT “Note: ”
    PRINT :bulletpoint: emoji, “ you will need to make full payment to confirm your booking.”
    PRINT :bulletpoint: emoji, “ you can overwrite your previous payment method if you had
    not paid the booking”
    PRINT NEW LINE
    CALL payment_method_request()
    count = 1
    statement_count = 1
    OPEN FILE “customerBookingPayment.txt” IN WRITE MODE
        FOR EACH statement IN statements
            REMOVE LAST statement
            IF (statement_count < statements LENGTH) THEN
                count  = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE “customerBookingPayment.txt” WRITE details,
                        “ | ”
                        count  = count + 1
                    ELSE
                        FILE “customerBookingPayment.txt” WRITE details
                    ENDIF
                ENDFOR
                FILE “customerBookingPayment.txt” WRITE NEW LINE
                statement_count = statement_count + 1
            ELSE
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE “customerBookingPayment.txt” WRITE details,
                        “ | ”
                        count = count + 1
                    ELSE
                        FILE “customerBookingPayment.txt” WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE “customerBookingPayment.txt”
    count = 1
    car_count = 1
    OPEN FILE “carDatabase.txt” IN WRITE MODE
    FOR EACH car IN cars
        IF (car_count < cars LENGTH) THEN
            count = 1
            FOR EACH details IN car
                IF (count < car LENGTH) THEN
                    FILE “carDatabase.txt” WRITE details, “ | ”
                    count = count + 1
                ELSE
                    FILE “carDatabase.txt” WRITE details
                ENDIF
            ENDFOR
            FILE “carDatabase.txt” WRITE NEW LINE
            car_count = car_count + 1
        ELSE
            count = 1
            FOR EACH details IN car
                IF (count < car LENGTH) THEN
                    FILE “carDatabase.txt” WRITE details, “ | ”
                    count = count + 1
                ELSE
                    FILE “carDatabase.txt” WRITE details
                ENDIF
            ENDFOR
        ENDIF
    ENDFOR
    CLOSE FILE “carDatabase.txt”
    count = 1
    customer_count = 1
    OPEN FILE “customerDetails.txt” IN WRITE MODE
        FOR EACH customer IN customers
            IF (customer_count < customers LENGTH) THEN
                count = 1
                FOR EACH details IN customer
                    IF (count < customer LENGTH) THEN
                        FILE “customerDetails.txt” WRITE details, “ | ”
                        count = count + 1
                    ELSE
                        FILE “customerDetails.txt” WRITE details
                    ENDIF
                ENDFOR
                FILE “customerDetails.txt” WRITE NEW LINE
                customer_count = customer_count + 1
            ELSE
                count = 1
                FOR EACH details IN customer
                    IF (count < customer LENGTH) THEN
                        FILE “customerDetails.txt” WRITE details, “|”
                        count = count + 1
                    ELSE
                        FILE “customerDetails.txt” WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE “customerDetails.txt”
    PRINT NEW LINE
    PRINT “Redirecting to customer functionalities page...”
    PRINT NEW LINE
    CALL reg_customer()
ENDFUNCTION
```



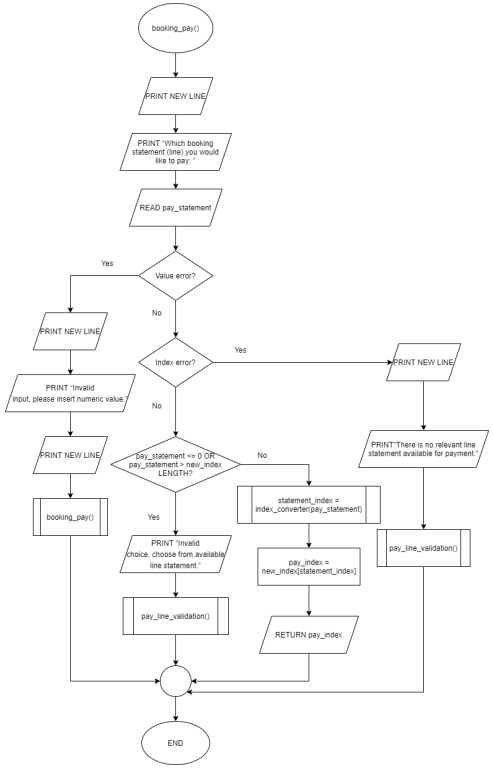
## Continue paying?

```
FUNCTION cont_pay()
        PRINT NEW LINE
        PRINT “Continue payment?”
        PRINT NEW LINE
        PRINT “    [YES] or [NO]”
        PRINT NEW LINE
        PRINT “Note: Select [NO] return to customer functionalities page”
        PRINT NEW LINE
        PRINT “Option => ”
        READ option
        IF (option == “YES”) THEN
                PRINT NEW LINE
                PRINT “Returning to the payment page...”
                CALL pay_car()
        ELSE
                IF (option == “NO”) THEN
                        PRINT NEW LINE
                        PRINT “You will be redirected to the functionalities page shortly....”
                        PRINT NEW LINE
                        CALL reg_customer()
                ELSE
                        PRINT “Invalid input, please insert [YES] or [NO].”
                        CALL cont_pay()
                ENDIF
        ENDIF
ENDFUNCTION
```



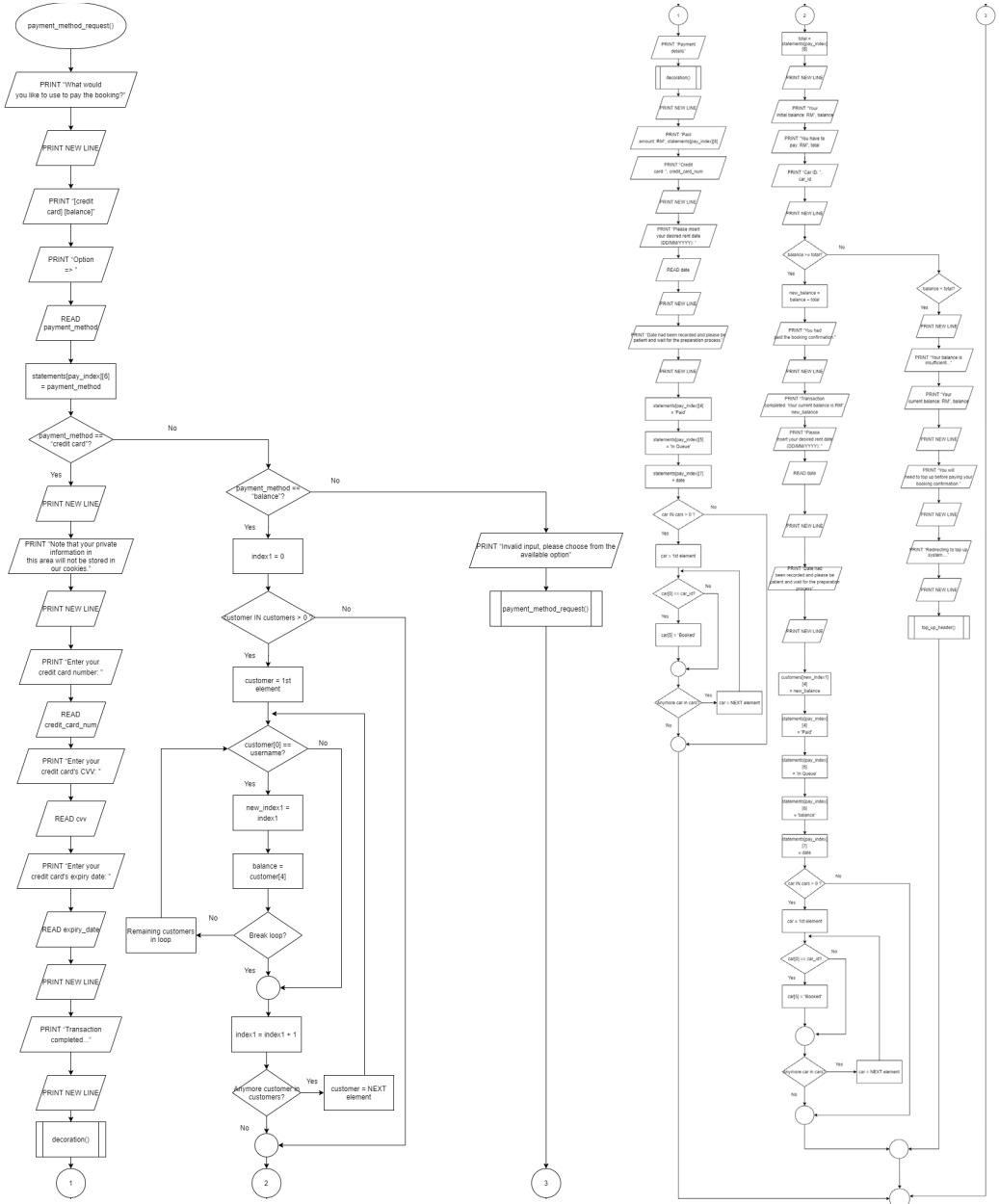

- Pay booking (Select line)

```
FUNCTION pay_line_validation()
        TRY
                PRINT NEW LINE
                PRINT “Which booking statement (line) you would like to pay: ”
                READ pay_statement
                IF (pay_statement <= 0) OR (pay_statement > new_index LENGTH) THEN
                        PRINT “Invalid choice, choose from available line statement”
                        CALL pay_line_validation()
                ELSE
                        statement_index = index_converter(pay_statement)
                        pay_index = new_index[statement_index]
                        RETURN pay_index
                ENDIF
        EXCEPT VALUE ERROR THEN
                PRINT NEW LINE
                PRINT “Invalid input, please insert numeric value.”
                PRINT NEW LINE
                RETURN pay_line_validation()
        EXCEPT INDEX ERROR THEN
                PRINT NEW LINE
                PRINT “There is no relevant line statement available for payment.”
                RETURN pay_line_validation()
        ENDTRY
ENDFUNCTION
```



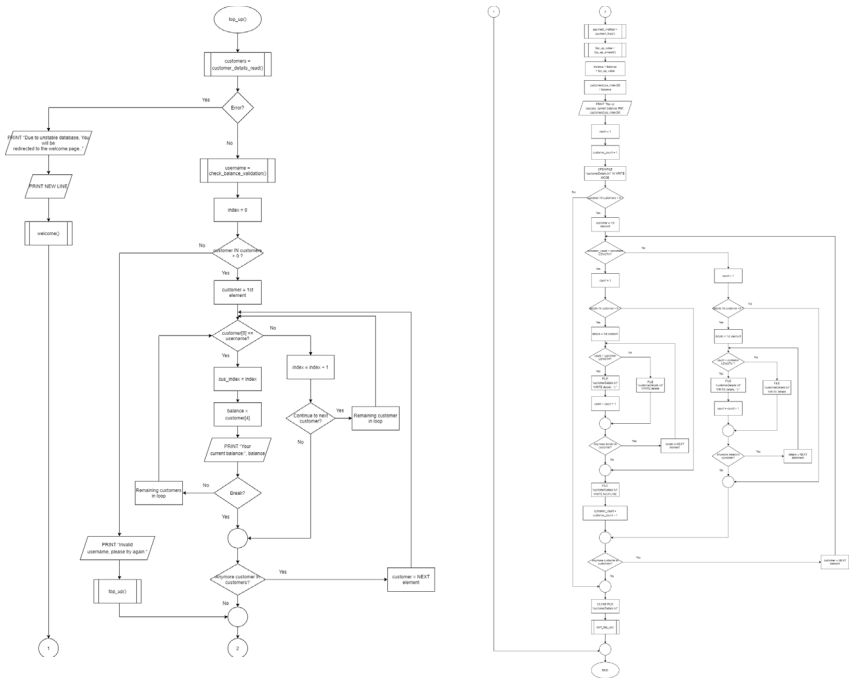
- Payment method request

```
FUNCTION payment_method_request()
    PRINT "What would you like to use to pay the booking?"
    PRINT NEW LINE
    PRINT "[credit card] [balance]"
    PRINT "Option => "
    READ payment_method
    statements[pay_index][6] = payment_method
    IF (payment_method == "credit card") THEN
        PRINT NEW LINE
        PRINT "Note that your private information in this area will not be stored in our cookies."
        PRINT NEW LINE
        PRINT "Enter your credit card number: "
        READ credit_card_num
        PRINT "Enter your credit card's CVV: "
        READ cvv
        PRINT "Enter your credit card's expiry date: "
        READ expiry_date                                    "
        PRINT NEW LINE
        PRINT "Transaction completed..."
        PRINT NEW LINE
        CALL decoration()
        PRINT "Payment details"
        CALL decoration()
        PRINT NEW LINE
        PRINT "Paid amount: RM", statements[pay_index][8]
        PRINT "Credit card: ", credit_card_num
        PRINT NEW LINE
        PRINT "Please insert your desired rent date (DD/MM/YYYY): "
        READ date
        PRINT NEW LINE
        PRINT "Date had been recorded and please be patient and wait for the preparation process."
        PRINT NEW LINE
        statements[pay_index][4] = 'Paid'
        statements[pay_index][5] = 'In Queue'
        statements[pay_index][7] = date
        FOR EACH car IN cars
            IF (car[0] == car_id) THEN
                car[5] = "Booked"
            ENDIF
        ENDFOR
    ELSE
        IF (payment_method == 'balance') THEN
            index1 = 0
            FOR EACH customer IN customers
                IF (customer[0] == username) THEN
                    new_index1 = index1
                    balance = customer[4]
                    BREAK LOOP
                ENDIF
                index1 = index + 1
            ENDFOR
            total = statements[pay_index][8]
            PRINT NEW LINE
            PRINT "Your initial balance: RM", balance
            PRINT "You have to pay: RM", total
            PRINT "Car ID: ", car_id
            PRINT NEW LINE
            IF (balance >= total) THEN
                new_balance = balance – total
                PRINT "You had paid the booking confirmation."
                PRINT NEW LINE
                PRINT "Transaction completed. Your current balance is RM", new_balance
                PRINT "Please insert your desired rent date (DD/MM/YYYY):
                READ date
                PRINT NEW LINE
                PRINT "Date had been recorded and please be patient and wait for the preparation process"
                PRINT NEW LINE
                customers[new_index1][4] = new_balance[
                statements[pay_index][4] = 'Paid'
                statements[pay_index][5] = 'In Queue'
                statements[pay_index][6] = 'balance'
                statements[pay_index][7] = date
                FOR EACH car IN cars
                    IF (car[0] == car_id) THEN
                        car[5] = "Booked"
                    ENDIF
                ENDFOR
            ELSE
                IF (balance < total) THEN
                    PRINT NEW LINE
                    PRINT "Your balance is insufficient..."
                    PRINT "Your current balance: RM", balance
                    PRINT NEW LINE
                    PRINT "You will need to top up before paying your booking confirmation."
                    PRINT NEW LINE
                    PRINT "Redirecting to top up system...."
                    PRINT NEW LINE
                    CALL top_up_header()
                ENDIF
            ENDIF
        ENDIF
    ELSE
        PRINT "Invalid input, please choose from the available option"
        CALL payment_method_request()
    ENDIF
ENDFUNCTION
```



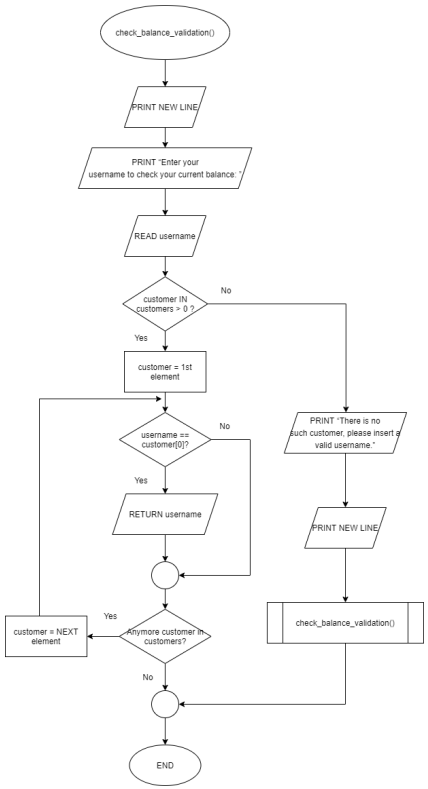
## d. Top up your balance

```
FUNCTION top_up()
    TRY
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT "Due to unstable database, You will be redirected to the welcome page.."
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    username = CALL check_balance_validation()
    index = 0
    FOR EACH customer IN customers
        IF (customer[0] == username) THEN
            cus_index = index
            balance = customer[4]
            PRINT "Your current balance:", balance
            BREAK LOOP
        ELSE
            index = index + 1
            NEXT customer
        ENDIF
    ELSE
        PRINT "Invalid username, please try again."
        CALL top_up()
    ENDFOR
    payment_method = CALL payment_type()
    top_up_value = CALL top_up_amount()
    balance = balance + top_up_value
    customers[cus_index][4] = balance
    PRINT "Top up success, current balance: RM", customers[cus_index][4]
    count = 1
    customer_count = 1
    OPEN FILE "customerDetails.txt" IN WRITE MODE
        FOR EACH customer IN customers
            IF (customer_count < customers LENGTH) THEN
                count = 1
                FOR EACH details IN customer
                    IF (customer_count < customers LENGTH) THEN
                        count = 1
                        FOR EACH details IN customer
                            IF (count < customer LENGTH) THEN
                                FILE "customerDetails.txt" WRITE details, " | "
                                count = count + 1
                            ELSE
                                FILE "customerDetails.txt" WRITE details
                            ENDIF
                        ENDFOR
                        FILE "customerDetails.txt" WRITE NEW LINE
                        customer_count = customer_count + 1
                    ELSE
                        count = 1
                        FOR EACH details IN customer
                            IF (count < customer LENGTH) THEN
                                FILE "customerDetails.txt" WRITE details, " | "
                                count = count + 1
                            ELSE
                                FILE "customerDetails.txt" WRITE details
                            ENDIF
                        ENDFOR
                    ENDIF
                ENDFOR
            CLOSE FILE "customerDetails.txt"
            CALL cont_top_up()
    ENDFUNCTION
```



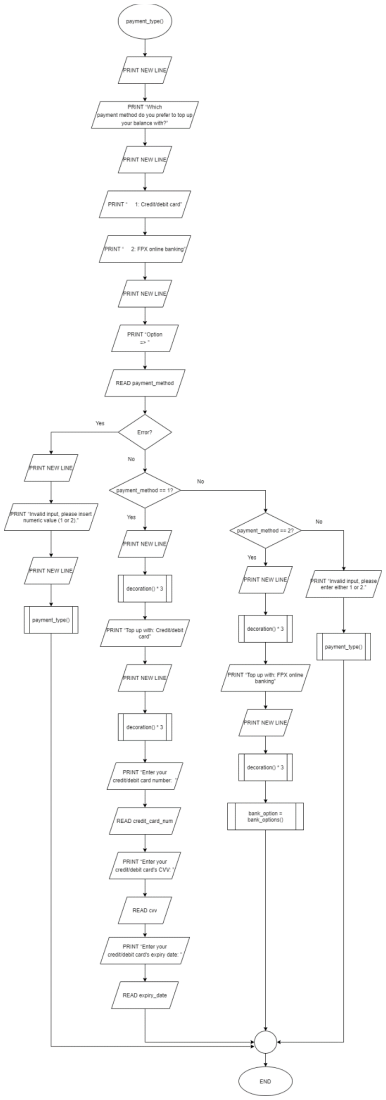
- Check balance validation

```
FUNCTION check_balance_validation()
    PRINT NEW LINE
    PRINT "Enter your username to check your current balance: "
    READ username
    FOR EACH customer IN customers
        IF (username == customer[0]) THEN
            RETURN username
        ENDIF
    ELSE
        PRINT "There is no such customer, please insert a valid username."
        PRINT NEW LINE
        CALL check_balance_validation()
    ENDFOR
ENDFUNCTION
```



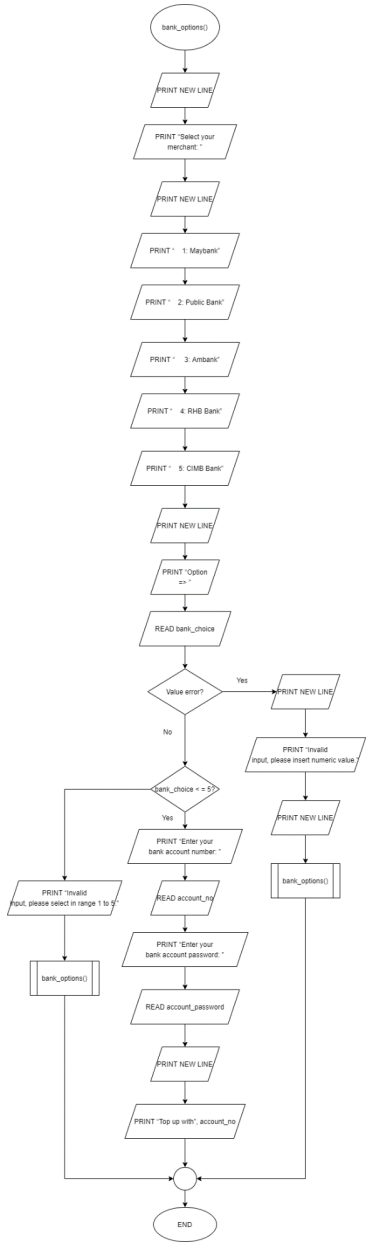
- Payment type

```
FUNCTION payment_type()
    TRY
        PRINT NEW LINE
        PRINT “Which payment method do you prefer to top up your balance with?”
        PRINT NEW LINE
        PRINT “    1: Credit/debit card”
        PRINT “    2: FPX online banking”
        PRINT NEW LINE
        PRINT “Option => ”
        READ payment_method
        IF (payment_method == 1) THEN
            PRINT NEW LINE
            CALL decoration() * 3
            PRINT “Top up with: Credit/debit card”
            PRINT NEW LINE
            CALL decoration() * 3
            PRINT “Enter your credit/debit card number:  ”
            READ credit_card_num
            PRINT “Enter your credit/debit card's CVV: ”
            READ cvv
            PRINT “Enter your credit/debit card's expiry date: ”
            READ expiry_date
        ELSE
            IF (payment_method == 2) THEN
                PRINT NEW LINE
                CALL decoration() * 3
                PRINT “Top up with: FPX online banking”
                PRINT NEW LINE
                CALL decoration() * 3
                bank_option = CALL bank_options()
            ELSE
                PRINT “Invalid input, please enter either 1 or 2.”
                CALL payment_type()
            ENDIF
        ENDIF
    EXCEPT ERROR THEN
        PRINT NEW LINE
        PRINT “Invalid input, please insert numeric value (1 or 2).”
        PRINT NEW LINE
        CALL payment_type()
    ENDTRY
ENDFUNCTION
```



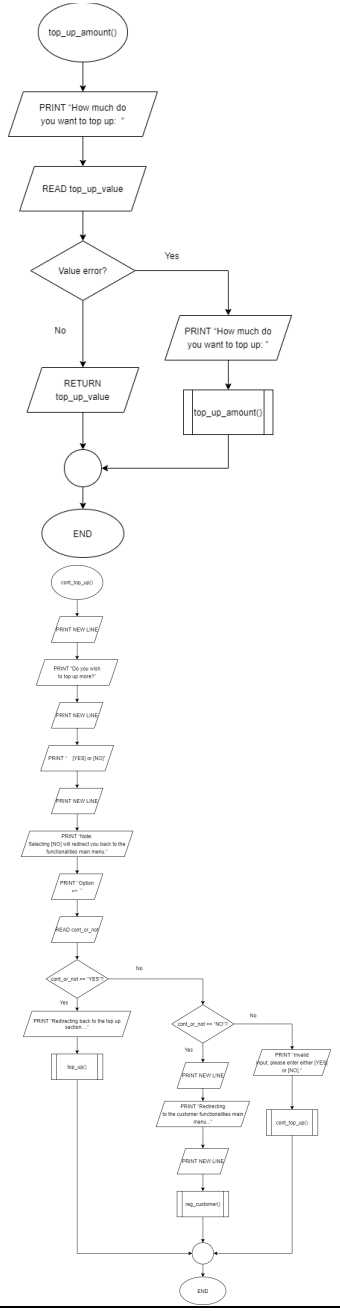

- Bank option

```
FUNCTION bank_options()
    PRINT NEW LINE
    PRINT “Select your merchant: ”
    PRINT NEW LINE
    PRINT “    1: Maybank”
    PRINT “    2: Public Bank”
    PRINT “    3: Ambank”
    PRINT “    4: RHB Bank”
    PRINT “    5: CIMB Bank”
    PRINT NEW LINE
    TRY
        PRINT “Option => ”
        READ bank_choice
        IF (bank_choice <= 5) THEN
            PRINT “Enter your bank account number: ”
            READ account_no
            PRINT “Enter your bank account password: ”
            READ account_password
            PRINT NEW LINE
            PRINT “Top up with”, account_no
        ELSE
            PRINT “Invalid input, please select in range 1 to 5.”
            CALL bank_options()
        ENDIF
    EXCEPT VALUE ERROR THEN
        PRINT NEW LINE
        PRINT “Invalid input, please insert numeric value.”
        PRINT NEW LINE
        CALL bank_options()
    ENDTRY
ENDFUNCTION
```



- Top up amount

```
FUNCTION top_up_amount()
    TRY
        PRINT "How much do you want to top up:  "
        READ top_up_value
        RETURN top_up_value
    EXCEPT VALUE ERROR THEN
        PRINT "How much do you want to top up: "
        CALL top_up_amount()
    ENDTRY
ENDFUNCTION
```

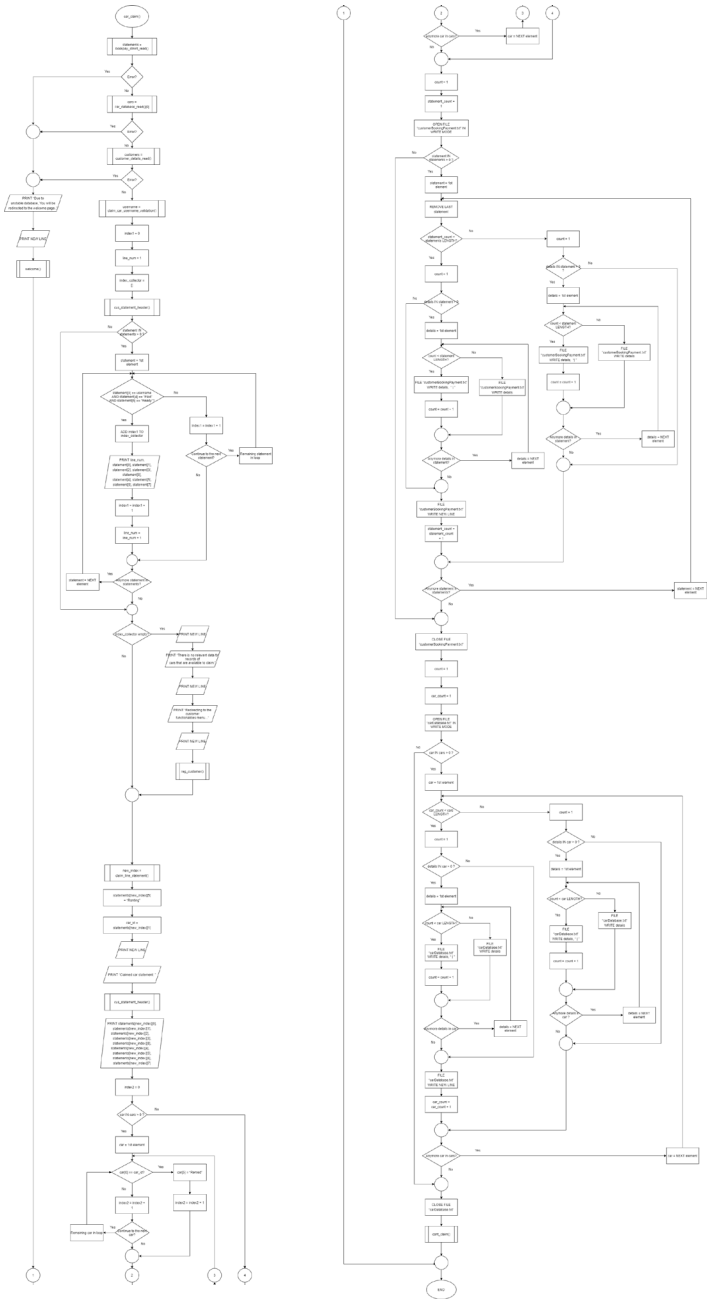



- Continue top up?

```
FUNCTION cont_top_up()
    PRINT NEW LINE
    PRINT "Do you wish to top up more?"
    PRINT NEW LINE
    PRINT "   [YES] or [NO]"
    PRINT NEW LINE
    PRINT "Note: Selecting [NO] will redirect you back to the functionalities main menu."
    PRINT "Option => "
    READ cont_or_not
    IF (cont_or_not == "YES") THEN
        PRINT "Redirecting back to the top up section...."
        CALL top_up()
    ELSE
        IF (cont_or_not == "NO") THEN
            PRINT NEW LINE
            PRINT "Redirecting to the customer functionalities main menu..."
            PRINT NEW LINE
            CALL reg_customer()
        ELSE
            PRINT "Invalid input, please enter either [YES] or [NO]."
            CALL cont_top_up()
        ENDIF
    ENDIF
ENDFUNCTION
```

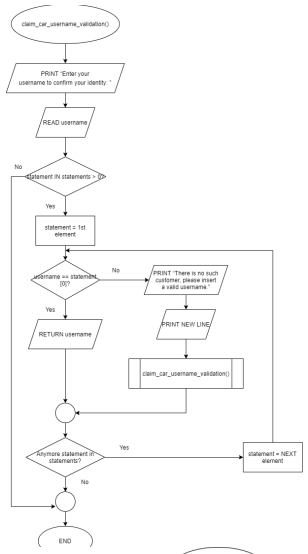
## f. Claim car that is ready to be rented

```
FUNCTION car_claim()
    TRY
        statements = CALL bookpay_stmnt_read()
        cars = CALL car_database_read()[0]
        customers = CALL customer_details_read()
    EXCEPT ERROR THEN
        PRINT "Due to unstable database, You will be redirected to the welcome page.."
        PRINT NEW LINE
        CALL welcome()
    ENDTRY
    username = CALL claim_car_username_validation()
    index1 = 0
    line_num = 1
    index_collector = []
    CALL cus_statement_header()
    FOR EACH statement IN statements
        IF (statement[0] == username) AND (statement[4] == "Paid") AND (statement[5] == "Ready") THEN
            ADD index1 TO index_collector
            PRINT line_num, statement[0], statement[1], statement[2], statement[3], statement[8], statement[4], statement[5], statement[6], statement[7]
            index1 = index1 + 1
            line_num = line_num + 1
        ELSE
            index1 = index1 + 1
            NEXT statement
        ENDIF
    ENDFOR
    IF index_collector EMPTY THEN
        PRINT NEW LINE
        PRINT "There is no relevant data for records of cars that are available to claim."
        PRINT NEW LINE
        PRINT "Redirecting to the customer functionalities menu..."
        PRINT NEW LINE
        CALL reg_customer()
    ENDIF
    new_index = claim_line_statement()
    statements[new_index][5] = "Renting"
    car_id = statements[new_index][1]
    PRINT NEW LINE
    PRINT "Claimed car statement: "
    CALL cus_statement_header()
    PRINT statements[new_index][0], statements[new_index][1], statements[new_index][2], statements[new_index][3], statements[new_index][8], statements[new_index][4], statements[new_index][5], statements[new_index][6], statements[new_index][7]
    FOR EACH car IN cars
        IF (car[0] == car_id) THEN
            car[5] = "Rented"
        ENDIF
    ENDFOR
    count = 1
    statement_count = 1
    OPEN FILE "customerBookingPayment.txt" IN WRITE MODE
        FOR EACH statement IN statements
            REMOVE LAST statement
            IF (statement_count < statements LENGTH) THEN
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE "customerBookingPayment.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "customerBookingPayment.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "customerBookingPayment.txt" WRITE NEW LINE
                statement_count = statement_count + 1
            ELSE
                count = 1
                FOR EACH details IN statement
                    IF (count < statement LENGTH) THEN
                        FILE "customerBookingPayment.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "customerBookingPayment.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE "customerBookingPayment.txt"
    count = 1
    car_count = 1
    OPEN FILE "carDatabase.txt" IN WRITE MODE
        FOR EACH car IN cars
            IF (car_count < cars LENGTH) THEN
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
                FILE "carDatabase.txt" WRITE NEW LINE
                car_count = car_count + 1
            ELSE
                count = 1
                FOR EACH details IN car
                    IF (count < car LENGTH) THEN
                        FILE "carDatabase.txt" WRITE details, " | "
                        count = count + 1
                    ELSE
                        FILE "carDatabase.txt" WRITE details
                    ENDIF
                ENDFOR
            ENDIF
        ENDFOR
    CLOSE FILE "carDatabase.txt"
    CALL cont_claim()
ENDFUNCTION
```



- Claim car username validation

```
FUNCTION claim_car_username_validation()
        PRINT “Enter your username to confirm your identity: ”
        READ username
        FOR EACH customer IN customers
                IF (username == customer[0]) THEN
                        RETURN username
                ELSE
                        PRINT “There is no such customer, please insert a valid username.”
                        PRINT NEW LINE
                        CALL claim_car_username_validation()
                ENDIF
        ENDFOR
ENDFUNCTION
```

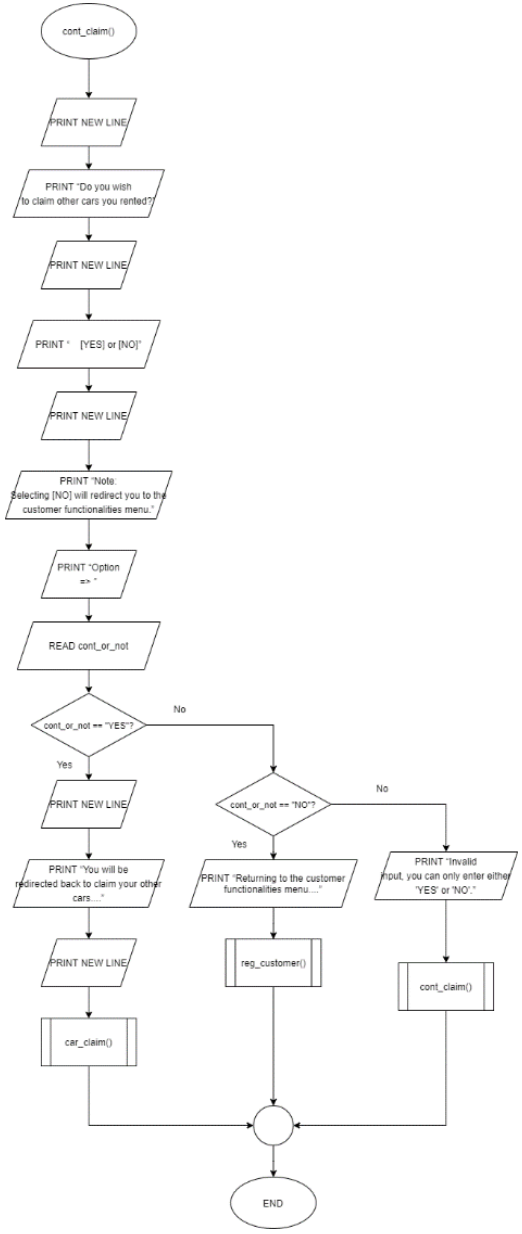



- Claim line statement

```
FUNCTION claim_line_statement()
        TRY
                PRINT NEW LINE
                PRINT “Which statement you would like to claim your car (line of statement): ”
                READ claim_car
                IF (claim_car <= 0) OR (claim_car > index_collector LENGTH) THEN
                        PRINT “Invalid choice, choose from available line statement.”
                        CALL claim_line_statement()
                ELSE
                        index_value = index_converter(claim_car)
                        new_index = index_collector[index_value]
                        RETURN new_index
                ENDIF
        EXCEPT VALUE ERROR THEN
                PRINT “Invalid input, please insert numeric value…”
                CALL claim_line_statement()
        EXCEPT INDEX ERROR THEN
                PRINT NEW LINE
                PRINT “There is no relevant line statement that appear to be claimed.”
                CALL claim_line_statement()
        ENDTRY
ENDFUNCTION
```

- Continue claiming?

```
FUNCTION cont_claim()
        PRINT NEW LINE
        PRINT “Do you wish to claim other cars you rented?”
        PRINT NEW LINE
        PRINT “    [YES] or [NO]”
        PRINT NEW LINE
        PRINT “Note: Selecting [NO] will redirect you to the customer functionalities menu.”
        PRINT “Option => ”
        READ cont_or_not
        IF (cont_or_not == "YES") THEN
                PRINT NEW LINE
                PRINT “You will be redirected back to claim your other cars....”
                PRINT NEW LINE
                CALL car_claim()
        ELSE
                IF (cont_or_not == "NO") THEN
                        PRINT “Returning to the customer functionalities menu....”
                        CALL reg_customer()
                ELSE
                        PRINT “Invalid input, you can only enter either 'YES' or 'NO'.”
                        CALL cont_claim()
                ENDIF
        ENDIF
ENDFUNCTION
```



## 3.0 Program source code explanation

### 3.1 General users' interface

#### a. Welcome & Role

```
def welcome():
    print("\n", decoration(), "Welcome to the Online Car Rental System(OCRS) by Super
Car Rental Services(SCRS)",
          decoration())

    # Menu
    print("""
Enter the number that best describe you.

    Admin : 1
    Customer : 2
    """)
    login_system()
```

The function *welcome()* is used to define the welcome page, followed by the welcome banner decorated with ten * each side by calling the *decoration()* function. A menu about selecting roles will be displayed to the users, 1 for admins while 2 for customers. *login_system()* will then be operated.

```
def login_system():
    # Ask for users' role
    def role_validation():
        try:
            role = int(input("Role => "))
            return role

        # Exclude non numeric value
        except ValueError:
            print("\nInvalid input, please insert a numeric value..\n")
            return role_validation()

    role = role_validation()
    # Administrator pass
    if role == 1:
        administrator_login()

    # Customer pass
    elif role == 2:
        customer_interface()

    # Error if neither 1 nor 2 are entered
    else:
        print("\nInvalid choice, please choose again.\n")
        return login_system()
```

The function *login_system()*, the integer that the user had entered will be stored in the *role* variable in a sub-function *role_validation()*. Inserting 1, users will be redirected to the administrator login screen by calling *administrator_login()* function. While inserting 2, they will be redirected to the customer main menu by calling *customer_interface()* function. Input other than 1 or 2 are not allowed, an error

message will appear and they will be given the option to reenter their choice again by calling *login_system()* again.

### b. Exit system

```
def exit_system():
    # Menu
    print("""
Are you sure you want to exit the Online Car Rental System?

    [YES] or [NO]

Note: Selecting [NO] will navigate you back to the welcome page.""")

    # Asking for exit confirmation
    confirmation = input("Option => ").upper()

    # Terminate the program
    if confirmation == "YES":
        print("\n", decoration(), "Exit", decoration(), "\n")
        exit()

    # Direct to the welcome page
    elif confirmation == "NO":
        print("You will be redirect to the welcome page shortly...\n")
        welcome()

    # Error when either "YES" or "NO" are not selected
    else:
        print("Invalid input, please enter YES or NO.")
        return exit_system()
```

The function *exit_system()* displays a confirmation message with 'YES' or 'NO' option will appear. The user's choice will be stored in the *confirmation* variable, if they have selected "YES", the program will terminate after the exit note displayed along with calling the *decoration()* function on both side of the note. If users have selected "NO", they will be sent back to the welcome page by calling the *welcome()* function with a message saying "You will be redirected to the welcome page shortly…" is displayed to provide users an idea of what the system is currently doing. Other than the two choices, an error message will be displayed and the user will be asked the same question again by calling the *exit_system()* function again.

## 3.2 File handling

- **Read data**

```
def car_database_read():
    # Extract car data to a list
    try:
        index = 0
        cars = []
        index_collector = []
        with open("carDatabase.txt", 'r') as car_file:
            for line in car_file:
                index_collector.append(index)
                row = line.strip().split(" | ")
                cars.append(row)
                index += 1

    except:
        print("\nDatabase is corrupted..")

    all_car_details = [cars, index_collector]

    return all_car_details
```

```
def bookpay_stmnt_read():
    # Extract customer booking / payment statement to a list
    try:
        statements = []
        with open("customerBookingPayment.txt", 'r') as statement_file:
            for line in statement_file:
                row = line.strip().split(" | ")
                statements.append(row)

        for statement in statements:
            total = int(statement[2]) * int(statement[3])
            statement.append(total)

    except:
        print("\nDatabase is corrupted..")

    return statements
```

```
def customer_details_read():
    # Extract customer information to a list
    try:
        customers = []
        with open("customerDetails.txt", 'r') as customers_file:
            for line in customers_file:
                row = line.strip().split(" | ")
                customers.append(row)

    except:
        print("\nDatabase is corrupted..")

    return customers
```

Three functions are defined to read data from three database file. Respectively:

- *car_database_read()* ➔ "carDatabase.txt"
- *bookpay_stmnt_read()* ➔ "customerBookingPayment.txt"
- *customer_details_read()* ➔ "customerDetails.txt"

Each line is being stripped, the data details in the line is being split by the separator " | ". These data are appended to list type variables like *cars*, *statements* and *customers* then returned to each function. If the file is not available, an error message is displayed. The difference is that *car_database_read()* return both *cars* and *index_collector* (index for each car) lists in *all_car_details*. While in *bookpay_stmnt_read()*, running through each statement, the function calculates the multiplication of price and days as *total* appended to each record in the *statements*.

```
# Extract customer booking / payment statement to a list
 try:
     statements = bookpay_stmnt_read()

# Extract car data to a database that store a list

     cars = car_database_read()[0]
     cars_index = car_database_read()[1]

# Extract customer details to variable that store a list

     customers = customer_details_read()
```

To utilize them, the functions are being called in variables *statements, cars, cars_index* and *customers* to store the lists. Then, the variables are being used to access the file data. The operations are being placed in the try clause to avoid file error.

| | |
|---|---|
| `# No file identified`<br>`except:`<br>`  print("Access the database system to check database issue.\n")`<br>`  maintenance_database_access()`<br>**Admins** | When Error,<br>"Access the database system" message<br>Execute *maintenance_database_access()* |
| `# No file spotted`<br>`except:`<br>`  print("Due to unstable database, You will be redirected to the welcome page..\n")`<br>`  welcome()`<br>**Visitors Customers** | When Error<br>"Redirected to welcome page" message<br>Execute *welcome()* |

- **Write data to file**

```
# Update data in text files
count1 = 1
with open('customerBookingPayment.txt', 'w') as mark_completed:
    stmnt_count = 1
    for statement in statements:
        statement.pop()
        if stmnt_count < len(statements):
            count = 1
            for details in statement:
                if count < len(statement):
                    mark_completed.write(f"{details} | ")
                    count += 1
                else:
                    mark_completed.write(details)
            mark_completed.write("\n")
            stmnt_count += 1
        else:
            count = 1
            for details in statement:
                if count < len(statement):
                    mark_completed.write(f"{details} | ")
                    count += 1
                else:
                    mark_completed.write(details)
```

"customerBookingPayment.txt" file

```python
# Transfer new data to the text file
with open('carDatabase.txt', 'w') as modified:
    car_count = 1
    for car in cars:
        if car_count < len(cars):
            count = 1
            for details in car:
                if count < len(car):
                    modified.write(f"{details} | ")
                    count += 1
                else:
                    modified.write(details)
            modified.write("\n")
            car_count += 1
        else:
            count = 1
            for details in car:
                if count < len(car):
                    modified.write(f"{details} | ")
                    count += 1
                else:
                    modified.write(details)
```

“carDatabase.txt” file

```python
# Update data in text files
count = 1
with open('customerDetails.txt', 'w') as modified:
    customer_count = 1
    for customer in customers:
        if customer_count < len(customers):
            count = 1
            for details in customer:
                if count < len(customer):
                    modified.write(f"{details} | ")
                    count += 1
                else:
                    modified.write(details)
            modified.write("\n")
            customer_count += 1
        else:
            count = 1
            for details in customer:
                if count < len(customer):
                    modified.write(f"{details} | ")
                    count += 1
                else:
                    modified.write(details)
```

“customerDetails.txt” file

The data in database file is being replaced by the latest information. For every records in *statements / cars / customers*, all of the details are being added to the file by splitting each of the detail with a “ | “ symbol. The last detail of every record is added without the symbol. Between every record, a new line is being written into the file to separate each car. For the last record, no new line is written into the file. The difference in these codes is that for “customerBookingPayment.txt” file, every last detail in the booking statement is being remove as the total amount of price is not required to stored in the database file. The functions that include these blocks are *admin_return_rent()*, *pay_car()* and *top_up()*.

## 3.3 Menu redirection / Navigation

- Multiple-choice menu

```
# 1. Customer landing page
def customer_interface():
    # Menu
    print("""
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.
""")
    # Input validation
    try:
        option = int(input("Option  => "))

        # Execute system based on option
        if option == 1:
            rent_car_details()
            cont_main_menu()
        elif option == 2:
            cus_reg_header()
        elif option == 3:
            registered_login()
        elif option == 4:
            exit_system()

        # Error if integers 1 to 4 are not entered, customers can try e
ntering again
        else:
            print("Invalid input, please insert valid value (1 to 4).")
            return customer_interface()

    # Non numeric value validation
    except ValueError:
        print("Invalid input, please insert a valid numeric value.")
        return customer_interface()
```

The multiple-choice menu executes different tasks based on users' input for number in a specific range. For example, *customer_interface()* displays a menu to the users and options are requested and stored in *option* variable. Selecting 1, users trigger *rent_car_details()* that view all cars that are available for rent and then *cont_main_menu()* is automated for users' redirection preference. Selecting 2, users trigger *cus_reg_header()* and can register as the car rental system customers. Selecting 3, users trigger *registered_login()* function and being directed to the customer login page. While option 4, users execute *exit_system()* and leave the system. Any other option will be displayed with error message and their option ids requested again. The other functions that implement this code style are *administrator_system(), admin_display()* and *reg_customer()*.

- YES or NO menu

```
    def cont_top_up():
        # Menu
        print("""
Do you wish to top up more?

    [YES] or [NO]

Note: Selecting [NO] will redirect you back to the functionalities main
 menu.""")
        cont_or_not = input("Option => ").upper()

        # Execute choices made by customers
        if cont_or_not == "YES":
            print("Redirecting back to the top up section....")
            top_up()
        elif cont_or_not == "NO":
            print("\nRedirecting to the customer functionalities main m
enu...\n")
            reg_customer()

        # Invalid input and request for customers' option again
        else:
            print("Invalid input, please enter either [YES] or [NO].")
            return cont_top_up()
```

The YES/NO menu only accepts YES and NO choice. It is also used in *exit_system()*, *admin_add_car()* and *cont_pay()*. In *cont_top_up()*, selecting "YES" will be redirected to the top up system by calling the *top_up()* function. Selecting "NO", users are redirected to customer functionalities page in *reg_customer()*. Other options result in error message and option is requested again.

- 1 or 2 Menu

```
def display_redirect():
    # Ask for admins' preference in redirection
    def redirect_validation():
        try:
            print("""
Do you wish to return to the display menu or administrator's main scree
n?

1: Display Menu
2: Administrator Main Screen
""")
            option = int(input("Option=> "))
            return option

        # Exclude non numeric value
        except ValueError:
            print("\nInvalid input, please insert a numeric value..")
            return redirect_validation()

    option = redirect_validation()

    # Return to the display menu
    if option == 1:
        print("\nYou will be redirected to the display menu...")
        admin_display()

    # Return to administrator functionalities main screen
    elif option == 2:
        print("\nYou are returning to the administrator main screen....
")
        administrator_system()

    # Invalid input, ask for redirection option again
    else:
        print("\nInvalid input, please key in 1 or 2.\n")
        return display_redirect()
```

YES/NO menu is commonly used to continue an action or terminate an action. In *display _redirect()*, it stores users' option in *redirect_validation()*. 1 to display menu, 2 to administrator main

screen, other options result in error message and choice is requested again. The functions that implement this code style are *search_choice_validation(), menu_direct_validation()* and *payment_type()*.

## 3.4 Empty print

```
# No data spotted
    if not emp_spotter:
        print(
            "\nThere is no relevant data for records of cars on that particular sta
tus.\n")

        # Admins' option on redirection
        display_redirect()
```

The system will check whether *emp_spotter, index_collector* or *new_index* is empty to make sure that there is no relevant data to display. If these lists are empty, the system will display there is no relevant data and will trigger certain function to navigate to the other functionality. The functions that use this code block are *dis_rent_car() cus_book_search()* and *rental_hist()*.

## 3.5 Input validation

### 3.5.1 Type validation

```
    def search_type_validation():
        print("\nOptions: [Username] [Car ID] [Days]\n")
        data_type = input(
            "What is the data type you would like to seek for: ").lower
()
        if data_type == "username":
            index = 0
        elif data_type == "car id":
            index = 1
        elif data_type == "days":
            index = 3

        else:
            print("Invalid input, please choose from the option.\n")
            return search_type_validation()
        return index
```

Type validation requests users to insert a data type for modify or search etc. In *search_type_validation(),* users can only choose username, car id and days and each of them return a specific index value. Other options will result in error message and choice is requested again. Related functions are *modify_type_validation(), paysearch_type_validation()* and *cusmodify_type_validation()*.

### 3.5.2 Line validation

```
    def modify_line_validation():
        try:
            data_line = int(
                input("\nWhich line of data you want to perform the mod
ification: "))
            car_index_convert = index_converter(data_line)
            car_index = cars_index[car_index_convert]

            if data_line <= 0:
                print("Invalid input, please try again.")
                return modify_line_validation()

            else:
                return car_index

        # Exclude non numeric value
        except ValueError:
            print("\nInvalid input, please insert a numeric value..")
            return modify_line_validation()

        # Exclude non existent records
        except IndexError:
            print("\nCar line is out of range.")
            return modify_line_validation()
```

Line validation function is used to exclude unnecessary errors and invalid choice. For example, when the option is lower than 0 and nonnumeric value are result in error message and requested to choose again. It is being used in *modify_line_validation()*, *return_line_validation(), select_car_validation()*, *pay_line_validation(), claim_car_statement().*

### 3.5.3 Username validation

```
    def username_validation():
        username = input("Enter your credential username: ")

        for customer in customers:
            if username == customer[0]:
                return username

        # Username not found in the database
        else:
            print("There is no such customer, please insert a valid use
rname.\n")
            return username_validation()
```

Username validation is used to check whether the username users insert is unavailable. The username is being compared to run through the usernames in customer details. Invalid choice results in error message and is requested again. The related functions are *booking_username_validation(), claim_car_username_validation()* and *check_balance_validation()*.

## 3.6 Admin

### a. Administrator login page

```
def administrator_login():
    # request for admins username and password
    administrator_username = "SCRSLL001"
    administrator_password = "SCRSOCRSLaiLim"

    username = input("\nEnter your username: ")
    password = input("Enter your password: ")

    # username and password validation
    if username == administrator_username and password == administrator_pa
ssword:
        administrator_system()
    else:
        print("Invalid credential, please check your username and password
.")
        return administrator_login()
```

The *administrator_login()* function will be triggered when users select the integer 1 (Admin) in the welcome menu. Admins are requested to enter their username and password. If the *username* and *password* match the predefined *administrator_username* and *administrator_password* the admin will be granted access into the system by calling the *administrator_system()* function, or else an error message will appear, and the admin will be able to try logging in again by calling *administrator_login()* again.

### b. Add cars to be rented out

```
def admin_add_car():
.
.
.
# Auto generate the new car id
last_car_id = cars[-1][0].split("R")
new_car_id_num = int(last_car_id[1]) + 1

# Add car while admins choose 'yes'
if add_or_not == "YES":
    new_car_details = []
    new_car_id = 'R' + f'{new_car_id_num:03d}'
    print("Car ID: ", new_car_id)
    new_car_brand = input("Enter car brand: ")
    new_car_model = input("Enter car model: ")
    new_car_plate = input("Enter car plate: ")
    new_year = input("Enter car manufacture year: ")

    def new_status_validation():
        new_status = input(
            "Enter status [Open / Rented / X / Booked]: ").capitalize()
        status = ['Open', 'Rented', 'X', 'Booked']
        if new_status not in status:
            print("Invalid status, Please enter a valid status.\n")
            return new_status_validation()
        else:
            return new_status
    new_status = new_status_validation()
.
.
.
new_price = new_price_validation()
new_car_details.extend([new_car_id, new_car_brand,
                        new_car_model, new_car_plate, new_year, new_status,
                               str(new_price)])
        # Append the new car data into carDatabase text file
        try:
            count = 1
            with open('carDatabase.txt', "a") as admin_add_details:
                admin_add_details.write("\n")
                for detail in new_car_details:
                    if count < len(new_car_details):
                        count += 1
                        admin_add_details.write(f"{detail} | ")
                    else:
                        admin_add_details.write(detail)
            print("\nInsert completed... Processing...\n")

        # No file identified
        except:
            print("Access the database system to check database
progress\n")
            maintenance_database_access()

        # Ask for admins choice on inserting more cars or stop adding
        return admin_add_car()
.
.
.
```

*admin_add_cars()* is used to add new cars into the car rental system. It will auto generate a new subsequent car id, the system extracts the car id at the last row of the text file, the car id is then separated from “R”, the rest integer will be added by one. When admins select “YES”, the car id will be display

in the “R000” format and admins are requested to enter the new car brand, model, plate, year, status and price. The status only allows ‘Open’, ‘Rented’, ‘X’ and ‘Booked’, other inputs will result in an error message. Besides, the newly keyed details will be added into the list, with the newly keyed in price as a string. The carDatabase.txt file will be opened in the append mode as the *admin_add_details* variable, however, if the file is not available, admins will be given a message relating to checking database progress, they will be redirected to the *maintenance_database_access()* function to solve the issue occurring within the database. If the file opening is successful, the data available in the *new_car_details* list will be added into the text file according to the number of details within with the format of “details |”, whereas for the last detail, the “|” will be taken away. After adding a new car, admins will be returned to the add car menu by calling the *admin_add_car()* function

### c. Modify car details

```
def admin_modify():
    # Display all cars data
    view_cars()
.
.
.
    car_index = modify_line_validation()
    print("\nIMPORTANT!!! Note that uniqueID can't be modified. "
          "\nIf the car had been removed from the system place X in ren
t status.\n")
.
.
.
.
       word_and_index = [origin_word, index]
       return word_and_index
    word_and_index = data_type_validation()
    origin_word = word_and_index[0]
    replace_index = word_and_index[1]
    print("Origin data: ", origin_word)

    # Replace data
    replace_word = input("Replaced by:  ")
    cars[car_index][replace_index] = replace_word

    # Display updated data
    print("\nDisplaying updated data.\n")
    car_header()
    print("   {:<8}{:<11}{:<11}{:<10}{:<7}{:<9}{:<6}"
          .format(cars[car_index][0], cars[car_index][1], cars[car_inde
x][2], cars[car_index][3], cars[car_index][4], cars[car_index][5], cars
[car_index][6]))
```

*admin_modify()* is used for admins to modify the car details. Admins are requested to select a line of car displayed to modify. The *word_and_index* variable extracts *origin_word* and *replace_index*. Then, the initial data of the selected car (*origin_word*) is displayed. Admins can insert the new data to replace the older data. The updated data will be displayed for users to see what they have changed, with the *car_header()* function called to display the headers of each attribute. Then the updated data is being uploaded o replace the data in “carDatabase.txt”.

### d. Display records

#### i. View all car data

```
# 1. View all car data
def view_cars():
.
.
.
    # Display all car data
    car_header()
    line_num = 1
    for car in cars:
        print("{:<4}{:<7}{:<12}{:<11}{:<10}{:<6}{:<8}{:<5}"
              .format(line_num, car[0], car[1], car[2], car[3], car[4],
 car[5], car[6]))
        line_num += 1
```

The *view_cars()* function displays all of the cars available in the database by going through a loop within the car database.

#### ii. Display car with different status

```
def dis_rent_car():
.
.
.
    # Menu
    print("\nCategories available [Open] [Rented] [X] [Booked]\n")

    # Request status
    status = input(
        "Status => ").capitalize()

    available_status = ["Open", "Rented", "X", "Booked"]

    # Display cars with matching status
    if status in available_status:
        index = 0
        emp_spotter = []
        line_number = 1
        car_header()
        for car in cars:
            if status == car[5]:
                emp_spotter.append(index)
                print("{:<4}{:<7}{:<12}{:<11}{:<10}{:<6}{:<8}{:<5}"
                      .format(line_number, car[0], car[1], car[2], car[3],
car[4], car[5], car[6]))
                index += 1
                line_number += 1
            else:
                index += 1
                continue
    else:
        print("\nInvalid type of status, please choose again.")
        return dis_rent_car()


    # Admins' option on redirection
    display_redirect()
```

The *dis_rent_car()* displays all of the cars based on a specific status insert by the admins. The status allowed are only ‘Open’, ‘Rented’, ‘X’ and ‘Booked’. Then, the system will display the cars according to the status. After that, *display_redirect()* function is called for admins to choose which menu they wish to be redirected to.

### iii. Customer Booking Statement

```
def dis_cus_booking():
    .
    .
    .
    # Set reservation only specify on booking status
    reservation = ['In Queue', 'Ready']

    # Display matching customer booking statement
    index = 0
    line_number = 1
    emp_spotter = []
    cus_book_header()
    for statement in statements:
        if reservation[0] == statement[5] or reservation[1] == statemen
t[5]:
            emp_spotter.append(index)
            print("{:<4}{:<11}{:<8}{:<7}{:<8}RM{:<9}{:<8}{:<12}{:<10}"
                  .format(line_number, statement[0], statement[1], stat
ement[2], statement[3], statement[8], statement[4], statement[5], state
ment[7]))
            index += 1
            line_number += 1
        else:
            index += 1
            continue
    .
    .
    .

    # Admins' option on redirection
    display_redirect()
```

*dis_cus_booking()* displays active customer booking statement. Running through data in *statements* list, the system retrieves all records with “In Queue” and “Ready” reservation. The index for the respective records is added into the *emp_spotter* and all the records are display along with the incrementing *line_num* starting from 1. In the meantime, the *index* and *line_number* are added 1. If the condition does not match, the *index* is added 1 and the loop will continue to the next iteration. After that, *display_redirect()* function is called for admins to choose which menu they wish to be redirected to.

### iv. Customer Payment Statement

```
def dis_cus_pay():
.
.
.
    def date_request_validation():
        # Menu instruction
        print("\nDisplay rent car between the start date and end date.\
n")
        try:
            # request start date
            start_date = input(
                "Insert the start date in DD/MM/YYYY Format: ").split("
/")
            start_day, start_month, start_year = int(
                start_date[0]), int(start_date[1]), int(start_date[2])
            start_date = [start_year, start_month, start_day]

            # request end date
            end_date = input(
                "Insert the end date in DD/MM/YYYY Format: ").split("/"
)
            end_day, end_month, end_year = int(
                end_date[0]), int(end_date[1]), int(end_date[2])
            end_date = [end_year, end_month, end_day]

            if start_month not in range(1, 13) or end_month not in rang
e(1, 13):
                print("\nInvalid date input, please insert a valid date
.\n")
                return date_request_validation()

            else:
                dates = [start_date, end_date]
                return dates

        except:
            print("\nInvalid input, please insert a date in DD/MM/YYYY
Format.")
            return date_request_validation()

    dates = date_request_validation()
    start_date = dates[0]
    end_date = dates[1]
```

```
# Set statement status only specify on 'Paid'
status = 'Paid'

# Display matching customer payment statement
index = 0
line_number = 1
emp_spotter = []
cus_pay_header()
for statement in statements:

    # make a valid date
    try:
        date = statement[7].split("/")
        date_year, date_month, date_day = int(date[2]), int(
            date[1]), int(date[0])
        date = [date_year, date_month, date_day]

    # Skip N/A value
    except:
        pass
      if status == statement[4] and start_date <= date <= end_date:
          emp_spotter.append(index)
          print("{:<4}{:<11}{:<8}{:<7}{:<8}RM{:<9}{:<8}{:<16}{:<10}"
                .format(line_number, statement[0], statement[1], stat
ement[2], statement[3], statement[8], statement[4], statement[6], state
ment[7]))
          index += 1
          line_number += 1
      else:
          index += 1
          continue
```

dis_cus_pay() displays all successfully paid rental record. Admins are requested to insert the *start_date* and *end_date* in DD/MM/YYYY format to display customer payment for a specific time duration. Running through the records in statements with the 'Paid' status, the rent date of the statement is being checked whether it is between the *start_date* and *end_date*. If it is, the records are being displayed with an auto-incremented *line_num* and the *index* of the corresponding statement is appended to the *emp_spotter*. After that, *display_redirect()* function is called for admins to choose which menu they wish to be redirected to.

## e. Search Record

```
def search_redirect():
```

```
def cus_book_search():
.
.
.
type_index = search_type_validation()
.
.
.
```

```
def cus_pay_search():
.
.
.
type_index = paysearch_type_validation()
.
.
.
```

*search_redirect()* function is the menu for *cus_book_search()* and *cus_pay_search()* which respectively search records in customer booking and customer payment.

```
def search_type_validation():
        print("\nOptions: [Username] [Car ID] [Days]\n")
        data_type = input(
            "What is the data type you would like to seek for: ").lower
()
        if data_type == "username":
            index = 0
        elif data_type == "car id":
            index = 1
        elif data_type == "days":
            index = 3

        else:
            print("Invalid input, please choose from the option.\n")
            return search_type_validation()
        return index
```

```
    search_phrase = input("Enter keyword to search: ")
    reservation = ["In Queue", "Ready"]

    # Display related records based on search_phrase in customer bookin
g statement
    index = 0
    line_num = 1
    emp_spotter = []
    cus_book_header()
    for statement in statements:
        if search_phrase == statement[type_index] and statement[5] in r
eservation and statement[4] == "Paid":
            emp_spotter.append(index)
            print("{:<4}{:<10}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<12}{:<10}"
                  .format(line_num, statement[0], statement[1], stateme
nt[2], statement[3], statement[8], statement[4], s
tatement[5], statement[7]))
            index += 1
            line_num += 1
        else:
            index += 1
            continue
```

Admins are required to enter a data type and a keyword to search, a loop is implemented to go through the statement in statements, this is applicable for the *cus_book_search()* and the *cus_pay_search()* function.

| *cus_book_search()* | *cus_pay_search(* |
|---|---|
| • The *search_phrase* is equivalent to the desired data type of the statement<br>• Reservation status (*statement[5]*) is either "In Queue" or "Ready"<br>• Payment status (*statement[4]*) is "Paid" | • The *search_phrase* is equivalent to the desired data type of the statement<br>• Payment status (*statement[4]*) is "Paid" |

The system will display the relevant statements and the index is added into the *emp_spotter* list. The *index* and *line_num* variable will be added by 1, if no statements met the criteria above, the *index* will increment by 1 and the loop continues to the next iteration.

### f. Return a rented car

```
def admin_return_rent():
.
.
.
   # Display data that required to be return to the system
    index1 = 0
    line_num = 1
    index_collector = []
    cus_statement_header()
    for statement in statements:
        if statement[4] == 'Paid' and statement[5] == 'Renting':
            index_collector.append(index1)
            print("{:<4}{:<11}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:
<10}"
                  .format(line_num, statement[0], statement[1], stateme
nt[2], statement[3], statement[8], statement[4], statement[5], statemen
t[6], statement[7]))
            index1 += 1
            line_num += 1
        else:
            index1 += 1
            continue
.
.
.
    def return_line_validation():
.
.
.
   new_index = return_line_validation()
    # Change reservation to completed
    statements[new_index][5] = 'Completed'
    car_id = statements[new_index][1]
    # Find for cars that status should be changed to 'Open'
    for car in cars:
        if car[0] == car_id:
            # Set car status to 'Open'
            car[5] = 'Open'
    # Display updated data
    print("\nReturned rent statement: ")
    cus_statement_header()
    print("   {:<11}{:<8}{:<7}{:<8}RM{:<8}{:<9}{:<13}{:<16}{:<10}"
          .format(statements[new_index][0], statements[new_index][1], s
tatements[new_index][2], statements[new_index][3], statements[new_index
][8], statements[new_index][4], statements[new_index][5], statements[ne
w_index][6], statements[new_index][7]))
```

In the *admin_return_rent()* function, the system goes through the statement in statements, statements that matches the following criteria:

- Payment status (statement[4]) is “Paid”
- Reservation status (statement[5]) is “Renting”

The collected *index1* will be appended into the *index_collector* and prints out statements relating to the yet to be returned cars. If the statement does not meet the above conditions, the loop continues to the next iteration. After changing the reservation status to ‘Completed’, the system needs to find relevant cars to change the status to “Open”. The if statement is used to determine whether the value in the *car_id* matches *car[0]*, if it does, the status of the car (*car[5]*) will be equivalent to “Open”. Then, the system will display the returned rent statement which is the new updated data of the database relating to the process.

## 3.7 Customer

### a. Customer registration

```
def customer_registration():
    # Request for customers' username and password for registration
    username = input("Enter your username: ")
    password = input("Enter your password: ")
    .
    .
    .
    # Extract all the username
    username_holder = []
    for customer in customers:
        username_holder.append(customer[0])

    # Usernames are in the database / usernames must be unique
    if username in username_holder:
        print("\nThis username had been used, please use another userna
me.\n")
        return customer_registration()

    # Obtaining new customers' details
    else:
        print(
            "\nPlease fill in the following details to provide more inf
ormation to rent a car / cars.\n")
        address = input("Address: ")
        contact_number = input("Contact number: ")
        # Every newly registered customer will have RM 0 as their balance
        balance = 0

        # Registration Completed
        print(f"""Your balance is now: RM{balance}
You can recharge it from the customer functionalities page.

{decoration()} Thank you for registering {decoration()}
""")

        new_customer_details = []
        new_customer_details.extend(
            [username, password, address, contact_number, balance])

    try:
        # Inserting new users data into customerDetails.txt as a new re
ord
        count = 1
        with open('customerDetails.txt', "a") as new_customer:
            new_customer.write("\n")
            for detail in new_customer_details:
                if count < len(new_customer_details):
                    new_customer.write(f"{detail} | ")
                    count += 1
                else:
                    new_customer.write(str(detail))

    # No file identified
    except:
        print("Access the database system to check database issue.\n")
        maintenance_database_access()

    print("\nYou can login to the system now. Start renting your car! \
0001F601 \n")

    # Automatically redirect to customer landing page
    customer_interface()
```

*customer_registration()* is a function utilized for visitors to register their account to the system. Visitors are requested to fill in their new username and password. *username_holder* stores the existing username, which if exact usernames are detected, the username is not available, and visitors are requested to insert a different username. If the username is not found, the customers can proceed to enter details like address and contact number to complete the registration process. Every newly registered customer will have a balance of RM0. All details of the newly registered customer are added into the *new_customer_details* list. The customerDetails.txt file is opened in the append mode as the *new_customer* variable to insert new customers data into the file as a new record. Every details in *new_customer_detail* is being added to the file by splitting with a " | " symbol. The last detail is added without the symbol.

### 3.7.1 Customer log in

```
def registered_login():
    # Log in attempts starts counting,  after 3 attempts customer will
be terminate out of the login system
    for time in range(1, 4):
        print("\nAttempt: ", time, "\n")
        # Request for customers' username and password
        username = input("Enter your username: ")
        password = input("Enter your password: ")
.
.
.
       # Username and password validation based on customers' input
        for customer in customers:
            if username == customer[0] and password == customer[1]:
                print("\n", decoration(), " Welcome to the OCRS, ",
                      username, decoration(), "\n")
                reg_customer()
                break

        # Customers who keyed in the wrong username or password will be
 asked to try again
        else:
            print("\nInvalid username or password, please try again..."
)
.
.
.
            def registration_inquiry():
.
.
.
                    def register_account_ask():
.
.
.
```

*registered_login()* function is used for customers to login. Counting in three attempts, users are requested to insert their username and password. If the username and password is correct, users can gain access to the customer functionalities through *reg_customer()* function. However, when the incorrect username or password is entered, an error message will be shown and users proceed to the next attempt and in the third attempt if it does not match users' input, users will be redirected out of the login system.

## 3.8 Registered customers' functionalities

### a. Display all available to rent cars

```
# View detail of cars to be rented out.
def dis_all_rent():
    rent_car_details()

    # Call function to execute
    cont_book_car()
```

```
def rent_car_details():
.
.
.
    # Displaying cars based on the "open" status
    available_rent_car = []
    index_holder = []
    index = 0
    line_num = 1
    status = "Open"
    car_header()
    for car in cars:
        if status == car[5]:
            available_rent_car.append(car)
            index_holder.append(index)
            print("{:<4}{:<7}{:<12}{:<11}{:<10}{:<6}{:<8}{:<5}"
                  .format(line_num, car[0], car[1], car[2], car[3], car
[4], car[5], car[6]))
            index += 1
            line_num += 1
        else:
            index += 1
            continue

    car_and_index = [available_rent_car, index_holder]
 return car_and_index
```

*dis_all_rent()* calls *rent_car_details()* function and display all available rent cars. The system identifies cars with the "Open" status to be displayed. The car is appended into the *available_rent_car* list while the *index* is appended into the *index_holder* list. The information of the cars is shown with an auto-increment *line_num*. If the car does not meet the condition, the *index* will increase by 1 and the loop continues, else *index* and *line_num* is added by 1 Then, the *available_rent_car* and *index_holder* values are stored in the *car_and_index* variables and returned.

**b. Modify personal details**

```
def modify_details():
.
.
.
   # Customers' username validation
    index = 0
    for customer in customers:
        if customer[0] == username:
            cus_index = index
            print(f"""
username: {customer[0]}
password: {customer[1]}
address: {customer[2]}
contact number: {customer[3]}
""")
            break
        index += 1
.
.
.
    def cusmodify_type_validation():
.
.
.
    modify_index = cusmodify_type_validation()
    old_data = customers[cus_index][modify_index]
    print("Original data: ", old_data)
    new_data = input("Replace with: ")
    customers[cus_index][modify_index] = new_data

    print(f"""
Modified details:
username: {customers[cus_index][0]}
password: {customers[cus_index][1]}
address: {customers[cus_index][2]}
contact number: {customers[cus_index][3]}
""")
```

*modify_details()* function allows customers to alter the initial username, password, address and contact number is displayed, which includes the original record that they would want to edit. After inputting the updated data, the data will be updated by finding their specific column index (*modify_index*) and row index (*cus_index*), then their modified details will be shown.

### c. View personal rental history

```
def rental_hist():
.
.
.
    def username_validation():
.
.
.
    username = username_validation()
.
.
.
   # username validation and display statement
    index = 0
    line_num = 1
    emp_spotter = []
    cus_statement_header()
    for statement in statements:
        if username == statement[0]:
            emp_spotter.append(index)
            print("{:<4}{:<11}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:
<10}".format(
                  line_num, statement[0], statement[1], statement[2], sta
tement[3], statement[8], statement[4], statement[5], statement[6], stat
ement[7]))
            index += 1
            line_num += 1
            continue
        else:
            index += 1
            continue
```

*rental_hist()* display all rental history Username validation is done before displaying the customers' rental history, the system goes through the statements to find the matching username to display the rental history after adding the found *index* into the *emp_spotter* list, the *index* and *line_num* variable will increase by 1 and the loop continues to the next iteration, or else the index will increase by 1 and the loop still continues as well.

### d. Book a car

```
def book_car():
    # Display details on available cars and transfer the statements dat
a to this function
    car_and_index = rent_car_details()
    available_rent_car = car_and_index[0]
    available_rent_index = car_and_index[1]
.
.
.
    new_index = select_car_validation()
.
.
.
    username = booking_username_validation()
    select_car = cars[new_index][0]

    # Extract car price
    for car in cars:
        if car[0] == select_car:
            price = car[6]
            break
   # Ask to book for how many days
    def days_validation():
    days = days_validation()
.
.
.
    new_booking_details = []
    new_booking_details.extend([username, select_car, price, days])
```

```
    try:
        with open("customerBookingPayment.txt", 'a') as new_booking:
            new_booking.write("\n")
            for detail in new_booking_details:
                new_booking.write(f"{detail} | ")
            new_booking.write("Pending | N/A | N/A | N/A")

    # No file spotted
    except:
        print("Due to unstable database, You will be redirected to the
welcome page..\n")
        welcome()

    # Automatically redirect to customer functionalities page
    print("\nYour booking is requested...\n\nNote: Your booking is not
confirm yet.\n"
          "You can make your payment by choosing option 5 to confirm yo
ur booking.\n"
          "\nYou will be redirected to the customer functionalities pag
e.")
    return reg_customer()
```

*book_car()* is used for customers to book the car. The detail on available cars is displayed by calling *rent_car_details()*, the statements are transferred to the *available_rent_car* and *available_rent_index* variable. The booking details made within this process which includes the *username, select_car, price* and *days* will be added into the *new_booking_details* list. The detail within the list is added into the text file by opening the customerBookingPayment.txt in append mode. Each of the details entered will be written within the text file with the "|" separator, behind the newly added statements, more attributes are also added in the form of "Pending | N/A | N/A | N/A".

**e. Pay to confirm booking**

```
def pay_car():
.
.
.
    # Enter username to confirm booking
    username = input(
        "\nEnter your username to pay for your booking confirmation: ")

    index = 0
    new_index = []
    line_num = 1
    cus_statement_header()
    for statement in statements:
        # Display booking that need to be paid
        if username == statement[0] and statement[4] == 'Pending' and s
tatement[6] == 'N/A':
            new_index.append(index)
            print("{:<4}{:<10}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:
<10}".format(
                line_num, statement[0], statement[1], statement[2], sta
tement[3], statement[8], statement[4], statement[5], statement[6], stat
ement[7]))
            line_num += 1
            index += 1

        # Display pay with balance statement / Redirected to pay with b
alance
        elif username == statement[0] and statement[4] == 'Pending' and
 statement[6] == 'balance':
            new_index.append(index)
            print("{:<4}{:<10}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:
<10}\t\t--
uncompleted payment due to insufficient balance before".format(
                line_num, statement[0], statement[1], statement[2], sta
tement[3], statement[8], statement[4], statement[5], statement[6], stat
ement[7]))
            line_num += 1
            index += 1
        else:
            index += 1
            continue
.
.
.
    def cont_pay():
.
.
.
```

*pay_car()* is used for customers to confirm a booking. Customers will need to select a car to book and enter their username to confirm their booking and make payment. The statement that meets the following conditions will be displayed along with line_num after appending the *index* into the *new_index* list.

- The username is found
- The payment status (statement[4]) is pending
- The payment method (statement[6]) is N/A (for Bookings)

- The payment method (statement[6]) is balance (pending balance payment)

The *line_num* and *index* variable will increment by 1.

```
    def payment_method_request():
        # Menu
        print("""What would you like to use to pay the booking?

[credit card] [balance]""")

        # Request customers to select their preferred payment method
        payment_method = input("Option => ").lower()

        # Update data
        statements[pay_index][6] = payment_method

        # Credit card payment method
        if payment_method == 'credit card':
            print(
                "\nNote that your private information in this area will
 not be stored in our cookies.\n")
            credit_card_num = input("Enter your credit card number: ")
            cvv = input("Enter your credit card's CVV: ")
            expiry_date = input("Enter your credit card's expiry date:
")

            # Display payment details
            print(f"""
Transaction completed...

{decoration()} Payment details {decoration()}

Paid amount: RM{statements[pay_index][8]}
Credit card: {credit_card_num}
    """)

            date = input("Please insert your desired rent date (DD/MM/Y
YYY): ")
            print(
                "\nDate had been recorded and please be patient and wai
t for the preparation process.\n")
```

The *payment_method_request()* function determines which way customers would like to pay their booking by selecting their desired payment method. The *payment_method* present will be updated the chosen methods.

**Selecting credit card:** They need to enter their credit card number, CVV and the card's expiry date. After inserting the card's details, the payment details will be displayed, customers also need to enter their desired rent date to start renting the car.

```
        # Redirect to balance payment page
        elif payment_method == 'balance':
            index1 = 0
            for customer in customers:
                if customer[0] == username:
                    new_index1 = index1
                    balance = int(customer[4])
                    break
                index1 += 1

            # Calculating the total amount customers should pay
            total = statements[pay_index][8]

            # Display how much the customer should pay and car ID
            print(f"""
Your initial balance: RM{balance}
You have to pay: RM{total}
Car ID: {car_id}
""")
```

**Selecting balance:** The system starts looping the customers to find the matching *username* to identify the *balance* of the customer, the loop will break. The *total* variable calculates the total rent amount customers should pay. After that, the system will display the initial balance, the amount that the customer must pay and car id.

```
            # Deduct from balance if it is more than total
            if balance >= total:
                new_balance = balance - total

                # Display remaining balance
                print(
                    "You had paid the booking confirmation.\nTransactio
n completed. Your current balance is RM", new_balance)

                date = input(
                    "Please insert your desired rent date (DD/MM/YYYY):
 ")
                print(
                    "\nDate had been recorded and please be patient and
 wait for the preparation process\n")

                # Change data to paid situation
                customers[new_index1][4] = str(new_balance)
                statements[pay_index][4] = 'Paid'
                statements[pay_index][5] = 'In Queue'
                statements[pay_index][6] = 'balance'
                statements[pay_index][7] = date

                for car in cars:
                    if car[0] == car_id:
                        car[5] = 'Booked'
```

**Balance >= Total:** The balance will be deducted from the total and the confirmation statement and the current balance is displayed. Customers are requested to insert desired date to start renting the car. The initial data available is updated to their respective statuses within the paid situation. The system starts to find the matching car id to change the car status to “Booked”.

```
# Insufficient balance to pay will be automatically redirected to the t
op up screen
            elif balance < total:
                print(f"""
Your balance is insufficient...
Your current balance: RM{balance}

You will need to top up before paying your booking confirmation.

Redirecting to top up system....
""")
                top_up_header()

        # Invalid input, reboot car payment page
        else:
            print("Invalid input, please choose from the available opti
on")
            return payment_method_request()

    payment_method_request()
```

**Balance < Total:** Customers with insufficient balance will be redirected to the top up system.

## 4.0 Additional features source code explanation

### 4.1 Admins

#### a. Database access

```
def maintenance_database_access():
    # Accessing database by verifying username and password
    maintenance_username = "SCRSLL001"
    maintenance_password = "SCRSOCRSLaiLim"

    username = input("Enter username: ")
    password = input("Enter password: ")

    # Validation of username and password
    if username == maintenance_username and password == maintenance_passwo
rd:
        print(f"\n{decoration()} Welcome administrator {decoration()}\n")
```

*maintenance_database_access()* function is triggered once any of the database text file is unreadable. Admins can access to create the file by entering username and password that match pre-

defined variables *maintenance_username* and *maintenance_password*. When the passcode matches, the system will display a header message.

```
        def admin_create_file():
            # Options on new database if it is not created
            try:
                # Menu
                print("""What database you would like to create

1: Car Database
2: Customer information Database
3: Customer Booking / Payment Database
""")
                create_file = int(input("Option => "))

                # Execute based on the choices entered
                if create_file == 1:
                    create_car = open("carDatabase.txt", "w")
                    print(
                        "\nDatabase created, you will be redirected to
the add car functionalities.")
                    create_car.close()
                    admin_add_car()
                elif create_file == 2:
                    create_user = open("customerDetails.txt", "w")
                    print(
                        "\nDatabase created, you will be redirected to
the administrator main screen.")
                    create_user.close()
                    administrator_system()
                elif create_file == 3:
                    create_booking = open("customerBookingPayment.txt",
 "w")
                    print(
                        "\nDatabase created, you will be redirected to
the administrator main screen.")
                    create_booking.close()
                    administrator_system()

                # Error message on inputing the shown value
                else:
                    print(
                        "Please insert a number from 1 to 3.\n")
                    return admin_create_file()

            # Exclude non numeric value
            except ValueError:
                print("\nPlease insert a numeric value.\n")
                return admin_create_file()

        admin_create_file()
```

Then, the nested function *admin_create_file()* is executed and admins will see a create file menu which they can choose to rebuild database text file by entering 1, 2 or 3.

**Selecting 1** will rebuild the *carDatabase.txt* file and will be directed to the add car function.
**Selecting 2** will rebuild *customerDetails.txt* file.
**Selecting 3** will rebuild *customerBookingPayment.txt* file.

Other numbers and non-numeric value will get an error message and admins will be requested to select their option again.

**b. Display dataset (**Registered customers' username)

```
def dis_cus_username():
.
.
.
  # Store all customers' username in a list
    customers_names = []
    for customer in customers:
        customers_names.append(customer[0])

    # Arrange customers' username alphabetically
    customers_names.sort()
    # Display all customers' username
    line_num = 1
    print('\n\tUsername')
    for name in customers_names:
        print(f"{line_num}\t{name}", end='\n')
        line_num += 1

    print("\nThis is all of the registered customers' username.")

    # Admins' option on redirection
    display_redirect()
```

*dis_cus_username()* is defined to display all customers' username in the system. *customers_names* variable stores every customer username (*customer[0]*) extracted from *customers*. The usernames in *customers_names* are being arranged alphabetically in an ascending manner. A header

"Username" is printed which is followed by all usernames stored in *customer_names* printed with *line_num* starting from 1 which is added up by 1 each time and *name*. After displaying all usernames, "This is all of …" message is printed. Then, *display_redirect()* is being called.

### c. Mark car as ready

```
def admin_mark_ready():
.
.
.
# Display data that required to be mark as 'Ready'
index1 = 0
line_num = 1
index_collector = []
cus_statement_header()
for statement in statements:
    if statement[4] == 'Paid' and statement[5] == 'In Queue':
        index_collector.append(index1)
        print("{:<4}{:<11}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:<10}"
              .format(line_num, statement[0], statement[1], statement[2],
statement[3], statement[8], statement[4],
                      statement[5], statement[6], statement[7]))
        index1 += 1
        line_num += 1
    else:
        index1 += 1
        continue
.
.
.
new_index = mark_ready_validation()
# Set reservation as Ready
statements[new_index][5] = 'Ready'

# Display returned rent car
print("\nMarked ready statement: ")
cus_statement_header()
print("   {:<11}{:<8}{:<7}{:<8}RM{:<8}{:<9}{:<13}{:<16}{:<10}"
      .format(statements[new_index][0], statements[new_index][1],
statements[new_index][2],
              statements[new_index][3], statements[new_index][8],
statements[new_index][4],
              statements[new_index][5], statements[new_index][6],
statements[new_index][7]))
```

*admin_mark_ready()* is used for admins in marking a car as ready in the customer booking. *cus_statement_header()* function is being called to display related attributes on the customer booking payment. By running through the records in *statements*, the system will seek for the statement with 'Paid' status (*statement[4]*) and 'In Queue' reservation (*statement[5]*). These records' *index1* is added to *index_collector* and each of the respective records is being displayed to the admins with *line_num* and statement details. For every related records, *index1* and *line_num* are being added 1. While non-related records only i*ndex1* is added 1. After admins choose a valid statement to mark ready, the system displays the marked ready statement and call cus_statement_header() function.

### d. Analytics Dashboard

```
def analytics_dashboard():
.
.
.
    # Retrieve all customer payment total amount
    payment_amount = []
    for statement in statements:
        if statement[4] == 'Paid':
            payment_amount.append(statement[8])

    # Obtain highest / lowest / total payment completed in the system
    max_paid = max(payment_amount)
    min_paid = min(payment_amount)
    total_paid = sum(payment_amount)
    # Display analytics information in a dashboard format
    print(f"""
{decoration()} OCRS Data Analytics Dashboard {decoration()}

Total cars:                         {len(cars)} cars
Total customers:                    {len(customers)} customers
Total booking / payment records:    {len(statements)} statements
Total sales / profits:              RM {total_paid}
Highest sales / profits:            RM {max_paid}
Lowest sales / profits:             RM {min_paid}

Keep up your great work! \U0001F44D

Returning to administrator system....""")
    administrator_system()
```

*analytics_dashboard()* function is a functionality for admins to display the data visualization in the car rental system. For every statement in *statements*, whenever the record's status is 'Paid', the statement total pay amount (*statement[8]*) is appended to *payment_amount* list. The *max_paid* variable stores the max value in *payment_amount*, *min_paid* stores the min value in *payment_amount* and *total_paid* stores the sum of every values in *payment_amount*. Then, the analytics board is displayed to the admins with page banner, total cars, total customers etc. After displaying the analytics dashboard, admins are automatically redirected to admin functionalities page by triggering *administrator_system()*.

## 4.2 Registered Customer

### a. Top up your balance

```
def top_up():
.
.
.
    username = check_balance_validation()
    # Display current balance based on username
    index = 0
    for customer in customers:
        if customer[0] == username:
            cus_index = index
            balance = int(customer[4])
            print("Your current balance: ", str(balance))
            break
        else:
            index += 1
            continue

    # Error message if username does not exist and request username aga
in
    else:
        print("Invalid username, please try again.")
        return top_up()
                # Bank options to proceed payment in FPX
                def bank_options():
.
.
.
                bank_option = bank_options()
.
.
.
    payment_method = payment_type()

    # Request top up amount
    def top_up_amount():
.
.
.
    top_up_value = top_up_amount()
    balance += top_up_value
    customers[cus_index][4] = str(balance)
    print("Top up success, current balance: RM",
          customers[cus_index][4])
```

*top_up()* is used for customers to top up their balance in the system. Customers are required to insert their username and the balance is being retrieved from the *customers.* If the username is not

detected, customers will receive an error message and username is requested again. Then, *top_up_value* variable is then triggered by calling the *top_up_amount()* function. It will ask for customers' input in the top up amount they would like to recharge which is being added to the *balance*. A top up success message is displayed with the updated balance and is modified in the text file.

### b. Claim car

```
def car_claim():
.
.
.
        def claim_car_username_validation():
.
.
.
username = claim_car_username_validation()

# Display information based on username and fits the requirement to claim
index1 = 0
line_num = 1
index_collector = []
cus_statement_header()
for statement in statements:
    if statement[0] == username and statement[4] == 'Paid' and statement[5]
== 'Ready':
        index_collector.append(index1)

print("{:<4}{:<10}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<15}{:<10}".format(
            line_num, statement[0], statement[1], statement[2],
statement[3], statement[8], statement[4],
            statement[5], statement[6], statement[7]))
        index1 += 1
        line_num += 1
    else:
        index1 += 1
        continue
```

```
  def claim_line_statement():
.
.
.
  new_index = claim_line_statement()
   # Change status
   statements[new_index][5] = 'Renting'
   car_id = statements[new_index][1]

   # Display claimed car statement
   print("\nClaimed car statement: ")
   cus_statement_header()
   print("    {:<10}{:<8}{:<7}{:<8}RM{:<9}{:<9}{:<13}{:<16}{:<10}".for
mat(
       statements[new_index][0], statements[new_index][1], statements[
new_index][2], statements[new_index][3], statements[new_index][8], stat
ements[new_index][4], statements[new_index][5], statements[new_index][6
], statements[new_index][7]))

   # Change car status to rented
   for car in cars:
       if car[0] == car_id:
           car[5] = 'Rented'
.
.
.
```

*car_claim()* function is used by customers to claim a car in the ready status. Customers enter their username then *cus_statement_header()* is being called to display all the customer booking / payment attributes displayed in a row. Running through *statements,* if the records consist of the username, status is "Paid" and reservation is "Ready", the statement is displayed along with an incrementing *line_num* starting from 1. The *index1* of the statement is being appended to the *index_collector* list. After displayed, *index1* and *line_num* are being added up by 1. When the record does not match it, only *index1* is being added 1 and the loop continues to the next iteration. Customers will choose a statement to be claimed and if the statement is valid, the status of the record is modified to "Renting" and the *car_id* is defined to retrieve the corresponding car in the rent statement. Then, the system calls *cus_statement_header()* function and the claimed car statement will be displayed to the customers. By running through the car records in *cars*, if the car id matched the *car_id*, the specific car status (*car[5]*) is being changed to 'Rented'.

## 5.0 Sample input/output explanation

### 5.1 Welcome Page

```
********** Welcome to the Online Car Rental System(OCRS) by Super Car Rental Services(SCRS) **********

Enter the number that best describe you.

    Admin : 1
    Customer : 2

Role => one

Invalid input, please insert a numeric value..

Role => 3

Invalid choice, please choose again.

Role => 1

Enter your username:
```

```
********** Welcome to the Online Car Rental System(OCRS) by Super Car Rental Services(SCRS) **********

Enter the number that best describe you.

    Admin : 1
    Customer : 2

Role => 2
```

When the online car system program is executed, the welcome page will be displayed. Users can only enter 1 for ➔Admins and 2 for ➔ Customers. Other options will receive an error message and option is requested again.

### 5.2 Admins

#### 5.2.1 Admins Login

```
Enter your username: SCRS001LL
Enter your password: SCRSOCRSLaiLim
Invalid credential, please check your username and password.
```

Once admins **select 1** in the welcome page, they are required to enter the correct username and password. If username and password are invalid, an error message will appear, and the admins will be given opportunity to re-insert their username and password.

```
Enter your username: SCRSLL001
Enter your password: SCRSOCRSLailim

********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action:
```

```
8: Exit the system.

Select your action: 0
Invalid input, please choose options in range of 1 to 8.

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action: eight

Invalid input, please insert a numeric value..
```

By entering the correct username and password, admins can get access to the functionalities board with a welcome banner and functionalities menu provided for the admins to choose from. This menu does not accept any input other than the number 1 to 8. As shown above, number 0 and text “eight” is displayed with error message and input is requested again after each error.

### 5.2.2 Add Car

```
********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action: 1
```

When admins **choose 1**, admins can add new car records to the car rental system.

```
Insert car data:
[YES] to continue
[NO] to stop and display all data. then redirected to admin main screen.

Option => y
Invalid input, please try again.

Insert car data:
[YES] to continue
```

```
Insert car data:
[YES] to continue
[NO] to stop and display all data. then redirected to admin main screen.

Option => yes
Car ID:  R093
Enter car brand: Toyota
Enter car model: Harrier
Enter car plate: HAY88
Enter car manufacture year: 2020
Enter status [Open / Rented / X / Booked]: Open
Enter price per day (Only numeric data): RM190

Invalid input, please insert a numeric value..

Enter price per day (Only numeric data): 190

Insert completed... Processing...


Insert car data:
```

Selecting options other than “YES” and “NO” will results in error message. When admins enter **“YES”**, admins will get an auto-incremented car ID. Then admins will need to insert the details of the car like car brand, car model, car plate, manufacture year, status and price. The price only accepts numeric value, the example above “RM190” is unacceptable whereas entering 190 proceeds to complete insertion. After that, admins can decide to choose yes or no again to insert a new car data.

**Changes in text file**

```
92   R092 | DC | Super | DCS11 | 2020 | Open | 90
93   + R093 | Toyota | Harrier | HAY88 | 2020 | Open | 190
```

The new car details are added to the end of “carDatabase.txt” file.

```
Insert car data:
[YES] to continue
[NO] to stop and display all data. then redirected to admin main screen.

Option => no

Displaying all data...

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R001   Perodua     Axia       MEH0145   2021  Booked  50
2   R002   BMW         X1         HEHE114   2020  Rented  100
94  R094   Toyota      Wish       WWF445    2018  X       90

 **********  Back to administrator main screen.  **********

 ********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action:
```

If admins select **"NO"**, the system will display all the car data. After displaying all the cars, the system will proceed to return to the administrator main screen again. Which admins can select their actions once again.

### 5.2.3 Modify car

```
********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action: 2

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R001   Perodua     Axia       MEH0145   2021  Booked  50
2   R002   BMW         X1         HEHE114   2020  Rented  100
3   R003   Perodua     Kancil     DAMN520   1998  Open    10
93  R093   Toyota      Harrier    HAY88     2020  Open    190
94  R094   Toyota      Wish       WWF445    2018  X       90

Which line of data you want to perform the modification:
```

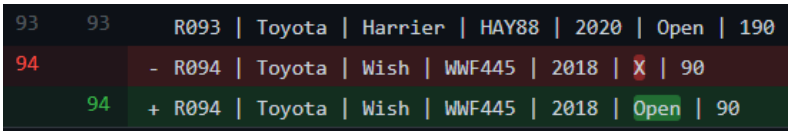
When admins **select 2**, admins will get all the car data display in an arranged manner. By looking at the list of the cars, admins can choose one of the car details to modify the details in it. To change a car details, admins must enter the desired line of data displayed by the system.

```
Which line of data you want to perform the modification: 100

Car line is out of range.

Which line of data you want to perform the modification: ninety four

Invalid input, please insert a numeric value..

Which line of data you want to perform the modification: 94

IMPORTANT!!! Note that uniqueID can't be modified.
If the car had been removed from the system place X in rent status.

Modify data type (e.g. Car Model, Price): status
Origin data:  X
Replaced by:  Open

Displaying updated data.


   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
   R094    Toyota     Wish       WWF445     2018  Open    90

Continue modifying?

    [YES] or [NO]

Note: Selecting [NO] will redirect you back to the administrator main menu.
Option =>
```

The line of statement the admins can select is only the lines in the available range from the 1st displayed car to the last car. For instance, 100 and "ninety-four" above face error as the available last

car is lesser than 100 and only allowed numeric value. While 94 is a valid input as it is in the range, admins can proceed to modify the details. Admins must choose the data type they want to modify like status, car plate etc. After selecting a data type, the system will display the original word and request the admins to enter the replace word. After these actions, the car is modified, and the line of car details will be display with the modified version again. After that, admins select **"YES"** to **continue modification** or select **"NO"** to return to the **administrator main screen**.

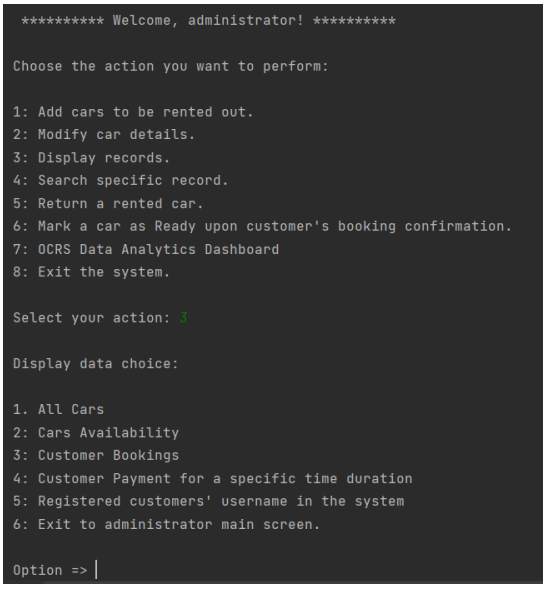
**Changes in text file**




The specific detail of the cars is being updated in the "carDatabase.txt" file.

Inserting other input than "YES" or "NO" is not allowed. Users can select "YES" to continue modify car details and select "NO" will be directly return to the administrator main screen.

### 5.2.4 Display data




To display records, admins need to **select 3** in the administrator main screen. Then, a menu of the available data choice is being displayed. Admins can select numbers in range 1 to 5 to display different data and select 6 to exit data display and return to the functionalities page.

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => one

Invalid input, please insert a numeric value..


Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 0
Invalid input. Please select choices in range 1 to 6.

Display data choice:
```

Selecting options other than number 1 to 6, admins will get an error message and is required to insert their choice on data type display again. As shown above, selecting “one” and 0 result in error message and each of it is followed with choice option request once again.

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 1

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R001   Perodua      Axia       MEH0145   2021  Booked  50
```

**Selecting 1** will display all the car details in the system.

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 2

Categories available [Open] [Rented] [X] [Booked]

Status => $poilt

Invalid type of status, please choose again.

Categories available [Open] [Rented] [X] [Booked]

Status =>
```

**Select 2** to display cars based on status availability. Selecting status other than “Open”, “Rented”, “X” and “Booked” will result in error message and is required to choose again.

```
Categories available [Open] [Rented] [X] [Booked]

Status => Open

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R003   Perodua     Kancil     DAMN520   1998  Open    10
2   R005   Audi        A4 Sedan   RW214     2002  Open    100
3   R008   Volks       Bently     SUK123    2012  Open    25
4   R009   Proton      Waja       ANGY1     2002  Open    4
5   R012   Toyota      Accord     TYT121    1990  Open    200
6   R013   Ford        Ranger XL  FRXL12    2000  Open    125
7   R015   Toyota      Mercedez   TMYC129   2011  Open    280
8   R022   Perodua     Psiva      PSV155    1980  Open    80
9   R024   Zebra       Animal     AGH991    2012  Open    200
10  R031   Toyota      Hype       HYI098    2019  Open    129
```

"Open" status

```
Categories available [Open] [Rented] [X] [Booked]

Status => Rented

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R002   BMW         X1         HEHE114   2020  Rented  100
2   R004   Perodua     Myvi       BUM0599   2000  Rented  60
3   R006   Mercedez    Benz       R092      2012  Rented  200
```

"Rented " status

```
Categories available [Open] [Rented] [X] [Booked]

Status => X

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R017   Proton      X70        UWU3445   2020  X       50
2   R023   Volks       VV1        TIY985    2000  X       20
3   R056   Jaguar      XF         VVIP123   2018  X       300
4   R059   Krock       Rock       KRO099    2019  X       250
5   R088   Thrax       388        THR8475   2013  X       80
```

"X" status

```
Categories available [Open] [Rented] [X] [Booked]

Status => Booked

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R001   Perodua     Axia       MEH0145   2021  Booked  50
2   R010   Proton      Wira       wee123    1998  Booked  5
3   R014   Geely       Lotus      LL129     2012  Booked  28s
```

"Booked" status

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 3

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Requested Rent Date
1   Love      R001    50     3     RM150        Paid    Ready        20/03/2021
2   Kari      R011    300    8     RM2400       Paid    Ready        31/05/2021
3   Kari      R006    200    1     RM200        Paid    Ready        01/05/2021
```

**Selecting 3** can display all active customer booking statement.

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 4

Display rent car between the start date and end date.

Insert the start date in DD/MM/YYYY Format: 01/13/2020
Insert the end date in DD/MM/YYYY Format: 01/02/2021

Invalid date input, please insert a valid date.
```

```
Insert the start date in DD/MM/YYYY Format: 2019-09-88

Invalid input, please insert a date in DD/MM/YYYY Format.

Display rent car between the start date and end date.

Insert the start date in DD/MM/YYYY Format:
```

**Selecting 4** can display customer payment for a specific time duration. Start date and end date are requested to display the car rent date between that timeframe. The date does not accept month value more than 12 or smaller than 1. If admins insert the date which is not in DD/MM/YYYY format such as "2019-09-88" in the example, admins will get an error message and the date will be requested again.

```
Insert the start date in DD/MM/YYYY Format: 01/04/2021
Insert the end date in DD/MM/YYYY Format: 20/04/2021

   Username  Car ID  Price  Days  Total Amount  Status  Payment Method  Requested Rent Date
1   Cheryl     R005    100    3       RM300      Paid    balance         01/04/2021
2   Cheryl     R039    200    5       RM1000     Paid    credit card     03/04/2021
3   Love       R040    30     5       RM150      Paid    balance         10/04/2021
4   Kaeya      R057    200    5       RM1000     Paid    credit card     02/04/2021
```

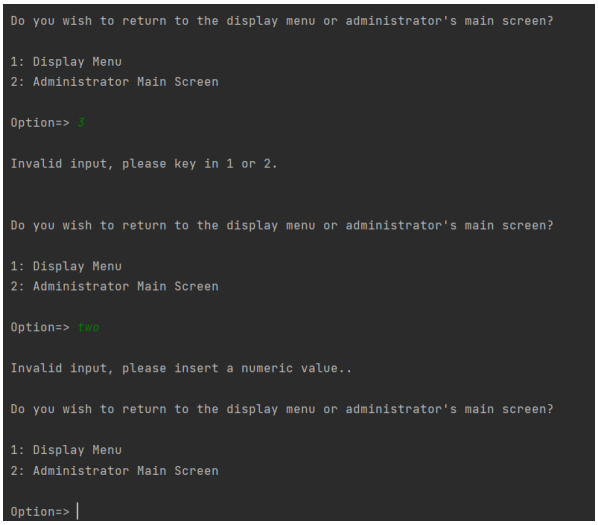
By inserting a valid start and end date, the system will display the customer booking / payment statement requested rent date which is between the start and end. For instance, all booking / payment statements with requested rent date between "01-04-2021" and "20-04-2021" are displayed as requested by admins.

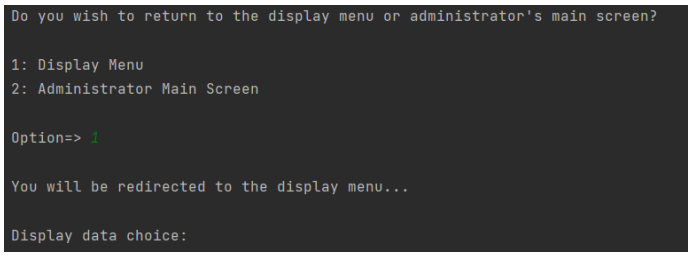
```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 5

    Username
1   Alex
2   Anthony
3   Ayaka
4   Banana
5   Cherry
```

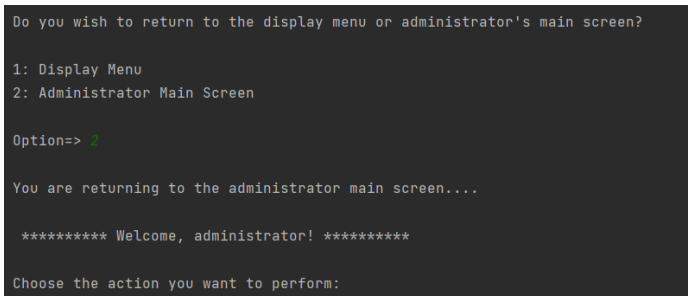
**Selecting 5** will display all the customer username arranged alphabetically.

```
Display data choice:

1. All Cars
2: Cars Availability
3: Customer Bookings
4: Customer Payment for a specific time duration
5: Registered customers' username in the system
6: Exit to administrator main screen.

Option => 6

 ********** Welcome, administrator! **********

Choose the action you want to perform:
```

**Selecting 6**, admins will return to administrator main screen.

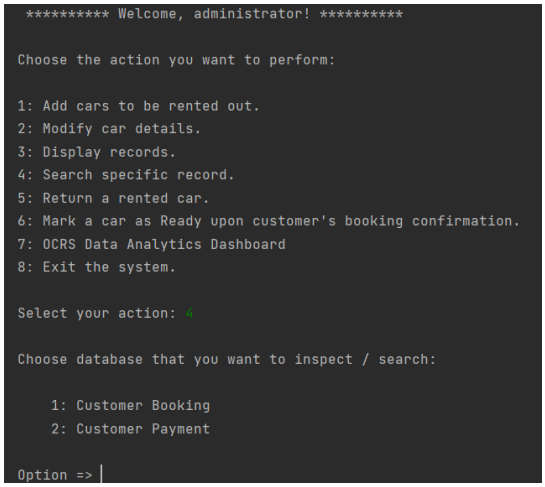



After displaying the data (from option 1 to 5), admins will be shown with a redirection menu either they want to return to the display menu or return to the main screen. Selecting options other than 1 or 2 will results in an error message and option for redirection is requested again. **Selecting 1** admins will be return to the display menu again to display more dataset. When admins **select 2**, admins are redirected to admin main screen which they can reselect their actions again.

### 5.2.5 Search record




In the functionalities menu, admins **select 4** to search for specific records in either customer booking or customer payment. After selecting 4 in the menu, admins will be displayed with a database option menu, 1 for Customer Booking and 2 for Customer Payment.

```
Choose database that you want to inspect / search:

    1: Customer Booking
    2: Customer Payment

Option => one

Invalid input, please insert a numeric value..


Choose database that you want to inspect / search:

    1: Customer Booking
    2: Customer Payment

Option => 3
Invalid input, choose only 1 or 2.


Choose database that you want to inspect / search:
```

On the search menu, admins can only select 1 or 2, other options will result in error message and admins will need to select their option again.

- Customer Booking

```
Choose database that you want to inspect / search:

    1: Customer Booking
    2: Customer Payment

Option => 1

Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: price
Invalid input, please choose from the option.
```

By **selecting 1**, admins can search either username, car ID or days in the customer booking records. The data types that users can search are username, car ID and days. Selecting other data types will result in an error message and the data type is requested again.

```
Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: username
Enter keyword to search: Vandyck

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Requested Rent Date
1   Vandyck   R038    160    6       RM960      Paid     In Queue    11/11/2021
2   Vandyck   R049    20     5       RM100      Paid     In Queue    12/07/2021

Do you want to return to the search menu or administrator main screen?
```

Data type: Username

Keyword: Vandyck

```
What is the data type you would like to seek for: car ID
Enter keyword to search: R001

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Requested Rent Date
1   Love      R001    50     3       RM150      Paid     Ready       20/03/2021

Do you want to return to the search menu or administrator main screen?
```

Data type: Car ID

Keyword: R001

```
What is the data type you would like to seek for: Days
Enter keyword to search: 5

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Requested Rent Date
1   Love      R040    30     5       RM150      Paid     Ready       10/04/2021
2   Vandyck   R049    20     5       RM100      Paid     In Queue    12/07/2021
3   Cheryl    R039    250    5       RM1250     Paid     Ready       15/05/2021
4   Cheryl    R062    100    5       RM500      Paid     Ready       04/05/2021
```

Data type: Days

Keyword: 5

After every valid search results, admins will be asked to choose their desired redirection.

```
Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: car ID
Enter keyword to search: R009

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Requested Rent Date

There is no relevant data for records of customer booking statement based on your search keyword.

Continue searching?

    [YES] or [NO]

Note: Selecting [NO] will navigate you back to administrator main screen.

Option =>
```

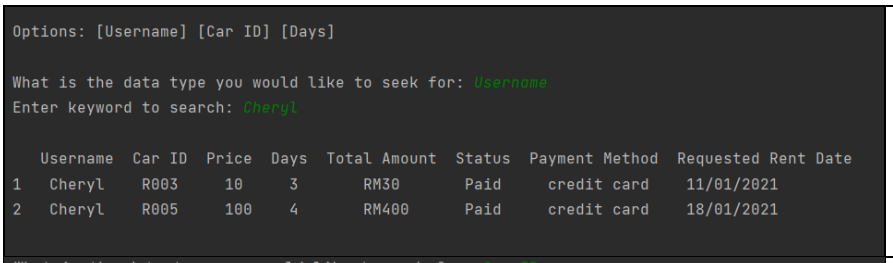
When there are no relevant data on the specific keyword, admins will get a "There is no relevant data …" message and will select **"YES"** continue searching or select **"NO"** to return to administrator main screen. An incorrect input will provide an error message, users are requested to choose again.

- Customer Payment

```
Choose database that you want to inspect / search:

    1: Customer Booking
    2: Customer Payment

Option => 2

Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: Username
```

```
Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: Total amount
Invalid input, please choose from the option.
```

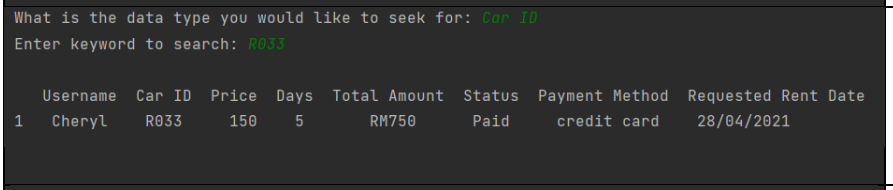
**Selecting 2**, admins can search in customer payment statements. Then, admins will receive options to choose the data type to search for a keyword. The system only accepts data types like username, car ID and days. Selecting options other than available data types will result in error message and the data type input is requested again.



```
Options: [Username] [Car ID] [Days]

What is the data type you would like to seek for: Username
Enter keyword to search: Cheryl

   Username  Car ID  Price  Days  Total Amount  Status  Payment Method  Requested Rent Date
1   Cheryl    R003     10     3       RM30       Paid     credit card    11/01/2021
2   Cheryl    R005     100    4       RM400      Paid     credit card    18/01/2021
```

Data type: Username

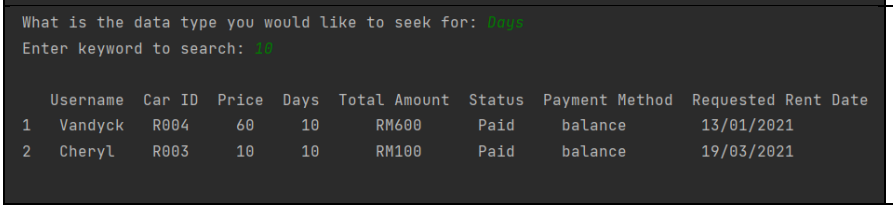
Keyword: Cheryl



```
What is the data type you would like to seek for: Car ID
Enter keyword to search: R033

   Username  Car ID  Price  Days  Total Amount  Status  Payment Method  Requested Rent Date
1   Cheryl    R033     150    5       RM750      Paid     credit card    28/04/2021
```

Data type: Car ID

Keyword: R033

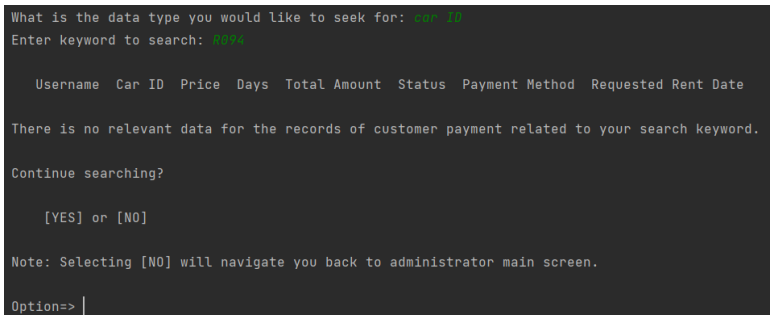


```
What is the data type you would like to seek for: Days
Enter keyword to search: 10

   Username  Car ID  Price  Days  Total Amount  Status  Payment Method  Requested Rent Date
1   Vandyck   R004     60     10      RM600      Paid     balance        13/01/2021
2   Cheryl    R003     10     10      RM100      Paid     balance        19/03/2021
```

Data type: Days

Keyword: 10

After every valid search results, admins will be asked to choose their desired redirection.



```
What is the data type you would like to seek for: car ID
Enter keyword to search: R094

   Username  Car ID  Price  Days  Total Amount  Status  Payment Method  Requested Rent Date

There is no relevant data for the records of customer payment related to your search keyword.

Continue searching?

    [YES] or [NO]

Note: Selecting [NO] will navigate you back to administrator main screen.

Option=>
```

When there are no relevant data on the specific search phrase, admins will be displayed with "There is no relevant data …" message. Then, admins can choose **"YES"** to continue their search for booking / payment statement again or **"NO"** to return to administrator main screen. Selecting other options, admins will receive error message and the option is requested again.

- Admins search redirection

```
Do you want to return to the search menu or administrator main screen?

    1: Search menu
    2: Administrator Main screen

Option => one

Invalid input, please insert a numeric value..


Do you want to return to the search menu or administrator main screen?

    1: Search menu
    2: Administrator Main screen

Option => 0

Invalid input, please select 1 or 2.


Do you want to return to the search menu or administrator main screen?
```

The redirection menu provides choice 1 for Search Menu and 2 for Administrator Main Screen. Options other than 1 or 2 will result in error message and admins' choice is requested again.

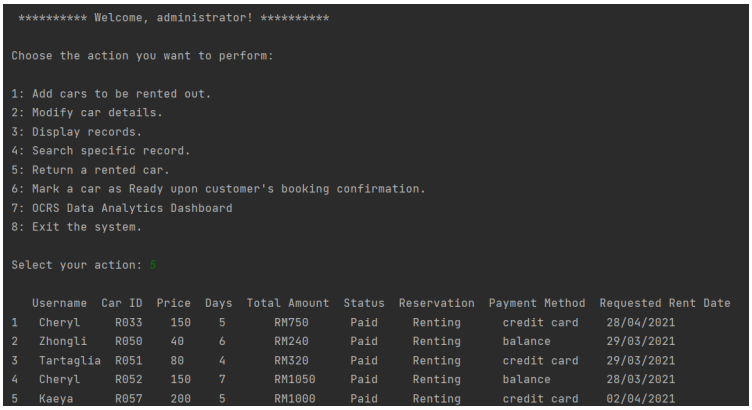
```
Do you want to return to the search menu or administrator main screen?

    1: Search menu
    2: Administrator Main screen

Option => 1

You will be redirected to the search menu shortly....


Choose database that you want to inspect / search:

    1: Customer Booking
```

**Selecting 1** in the redirection menu, admins can continue searching for input by selecting the desired database.

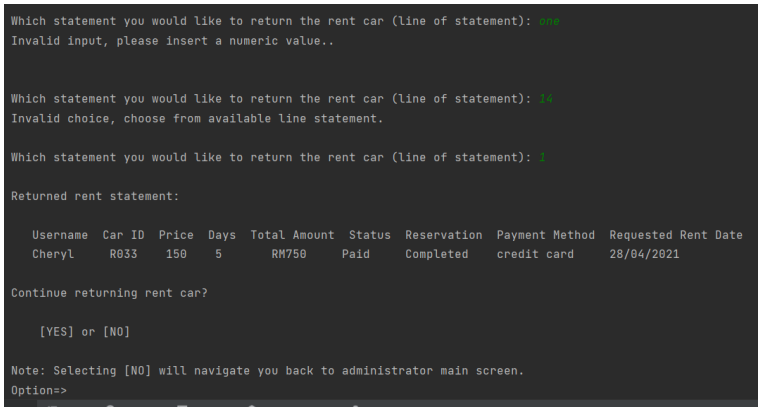
```
Do you want to return to the search menu or administrator main screen?

    1: Search menu
    2: Administrator Main screen

Option => 2

You will be redirected to the administrator main screen shortly...


********** Welcome, administrator! **********
```

**Selecting 2,** admins will be returned to the administrator main screen.

### 5.2.6 Return rented car

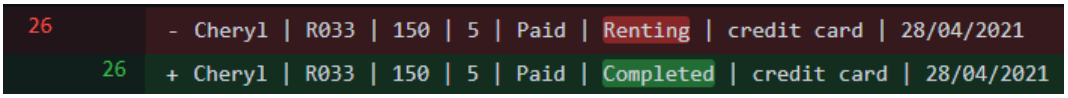



**Selecting 5** in the functionalities page, admins can return a rented car back to the system. This action is required when the customers' rent duration had ended for a particular request. After choosing 5, admins will receive a list of rental statements that are required to be mark ready.

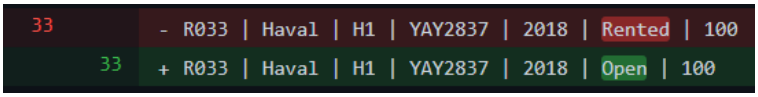



Then, from the list of statements, admins can select from the range of available line number. Non-numeric value and number out of available range will be result in error message and their option is requested again. While choosing a valid line, admins will receive a returned rent statement on their selection with updated data. After that, admins will need to choose **"YES"** to continue return rented car or **"NO"** returning to the administrator main screen.

**Changes in database file**

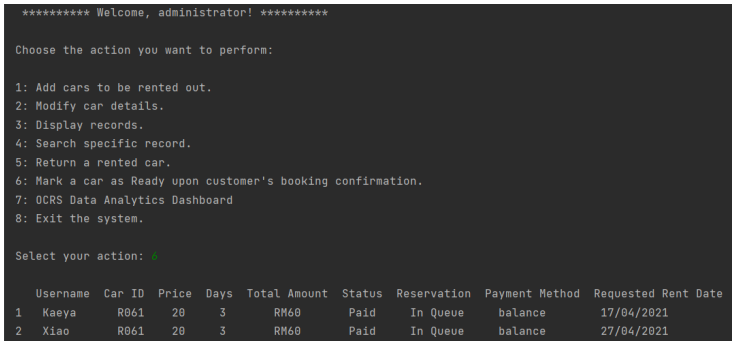


Customer booking / payment statement status changed from ‘Renting’ to ‘Completed’ in “cusBookingPayment.txt” file.

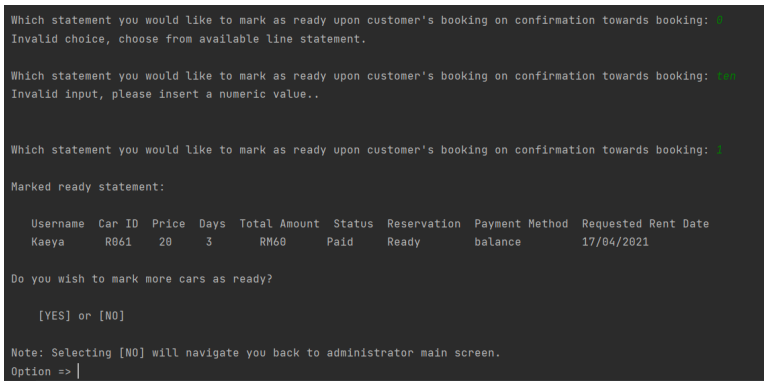



Car status is being changed from ‘Rented’ to ‘Open’ in “carDatabase.txt” file.

### 5.2.7 Mark ready




By **selecting 6** in the admins’ functionalities page, admins can mark a car as ready whenever a car is prepared to be claim upon customers’ booking confirmation. This functionality enables admins to arrange the sequence for which customer will get to rent the car first.




Then, admins can choose from valid statements to mark the specific car as ready for the customers’ statement. Entering a non-numeric value or options not available in the range of statement list, admins will be displayed with an error message and line statement request is requested again. When admins select a valid statement, admins will see a marked ready statement which is updated based on

their choice. Then, admins must select **"YES"** to navigate for marking more cars as ready or **"NO"** return to the administrator main screen. Selecting other options, admins will receive error message and will need to choose the option again.

**Changes in database file**

```
52      - Kaeya | R061 | 20 | 3 | Paid | In Queue | balance | 17/04/2021
52      + Kaeya | R061 | 20 | 3 | Paid | Ready | balance | 17/04/2021
```

Customer booking / payment statement status is being changed from 'In Queue' to 'Ready' in "customerBookingPayment.txt" file.

### 5.2.8 Data Analytics Dashboard

```
********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action: 7

********** OCRS Data Analytics Dashboard **********

Total cars:                         94 cars
Total customers:                    32 customers
Total booking / payment records:    83 statements
Total sales / profits:              RM 52461
Highest sales / profits:            RM 8100
Lowest sales / profits:             RM 20

Keep up your great work! 👍

Returning to administrator system....

********** Welcome, administrator! **********
```

Admins **select 7** in functionalities main screen to inspect the car rental system data analytics dashboard. The dashboard is decorated with a banner and admins can see the total cars, total customers, total rental statement (booking / payment), total profits, highest profits and lowest profits. After displaying all these data, admins will get a "Keep up your great work!" :thumbsup: emoji message and automatically return to the admin main screen.

## 5.3 Normal Customers / Visitors

```
********** Welcome to the Online Car Rental System(OCRS) by Super Car Rental Services(SCRS) **********

Enter the number that best describe you.

    Admin : 1
    Customer : 2

Role => 2

As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  =>
```

After **selecting 2** in the welcome page, customer will be displayed with the normal customers (visitors) interface. Visitors have three unique action steps and one exit system action.

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => one
Invalid input, please insert a valid numeric value.

As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => 5
Invalid input, please insert valid value (1 to 4).

As a visitor, select your action.
```

Options other than number from 1 to 4, customers will get an error message and are required to key in their choice again.

### 5.3.1 View all available to rent cars

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => 1

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R003   Perodua     Kancil     DAMN520   1998  Open    10
2   R005   Audi        A4 Sedan   RW214     2002  Open    100
```

```
Do you wish to go back to the main menu or login for more functionalities?

1: Main menu
2: Proceed to log in
3: Proceed to registration (For customer with no account in the server)
4: Exit the customer system

Option =>
```

**Selecting 1** will view all cars with ‘Open’ status which is available for rent. After all cars displayed, customers will get to see a redirection menu.

### 5.3.2 Customer Registration

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => 2

  ********** Welcome to the registration page **********

 Enter your username: Cheryl
 Enter your password: weeyi123

 This username had been used, please use another username.
```

**Selecting 2**, visitors can register to be a customer in the system to access more functionalities. To register, visitors will need to insert their desired username and password. If the username had been used, the visitors will receive an error message and need to use another username to create their account.

```
Enter your username: Tommy
Enter your password: youth

Please fill in the following details to provide more information to rent a car / cars.

Address: Besut
Contact number: 0192564353
Your balance is now: RM0
You can recharge it from the customer functionalities page.

********** Thank you for registering **********


You can login to the system now. Start renting your car! 😁


As a visitor, select your action.
```

After inserting a valid username and password, visitors will need to fill in more details like address and phone number to complete their registration. After filling in these details, visitors will see information about their balance, recharge notice and complete registration message. Then, automatically visitors are redirected to the visitor menu page and can now login as a registered customer.

**Changes in database file**

```
31  Valerie | night | Bukit Bintang | 0193467521 | 0
32  Lara | Craft | Klang | 0198790543 | 0
33  + Tommy | youth | Besut | 0192564353 | 0
```

The newly added / registered customer details are appended to the “customerDetails.txt” file.

### 5.3.3 Customer Login

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => 3

Attempt:  1
```

**Selecting 3** in the visitors' menu, customers can login with their registered account to access more features in the car rental system.

```
Attempt:  1

Enter your username: Hang Tuah
Enter your password: history

Invalid username or password, please try again...

Do you have an account?
[YES] or [NO]

Option => yes
Please try again.
```

To login the system, customers will need to insert their username and password. Customers are given three attempts to enter their credential username and password. In every attempt, customers are required to insert their username and password. If the username and password does not match or invalid, customers will be asked whether they have an account. If they choose **"YES"** the attempts with continue to attempt 2 and this will go on till attempt 3 when the wrong credentials are keyed in.

```
You had exceeded the limit to log in your account..
Please try again after 1 minute..
Redirecting to customer landing page....


As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  =>
```

After three attempts, customers will be removed from the login transaction and return to the visitor page.

```
Do you have an account?
[YES] or [NO]

Option => no

Do you wish to register a new account?

    [YES] or [NO]
```

While if customers select **"NO"** in the option of "Do you have an account?", customers will be given choice to decide whether they wish to register a new account or not.

```
Do you wish to register a new account?

    [YES] or [NO]

Note: Selecting [NO] will redirect you to customer landing page.
Option => yes
Redirecting to the customer registration page...


********** Welcome to the registration page **********

Enter your username:
```

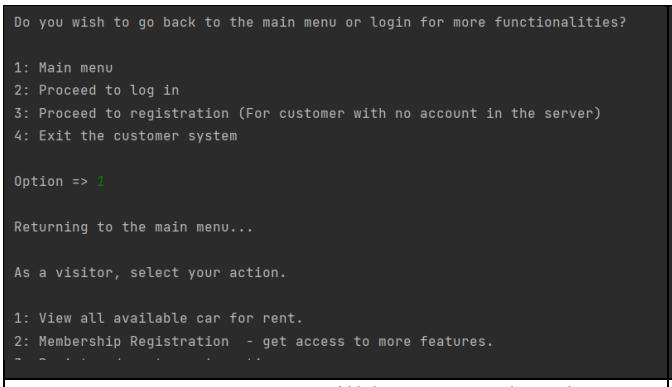
```
Note: Selecting [NO] will redirect you to customer landing page.
Option => no

Returning to the customer landing page..


As a visitor, select your action.

1: View all available car for rent.
```

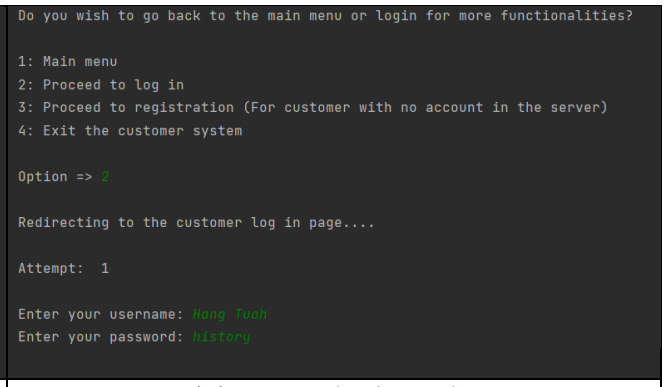
Selecting **"YES"** on wish to register, customers will be redirected to the customer registration page. Selecting **"NO"**, customers are redirected to the customer landing page.

### 5.3.4 Visitors' Redirection Menu

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => one
Invalid input, please insert a valid numeric value.

Do you wish to go back to the main menu or login for more functionalities?

1: Main menu
2: Proceed to log in
3: Proceed to registration (For customer with no account in the server)
4: Exit the customer system

Option => 0

Invalid input, please select either 1, 2, 3 or 4.

Do you wish to go back to the main menu or login for more functionalities?
```

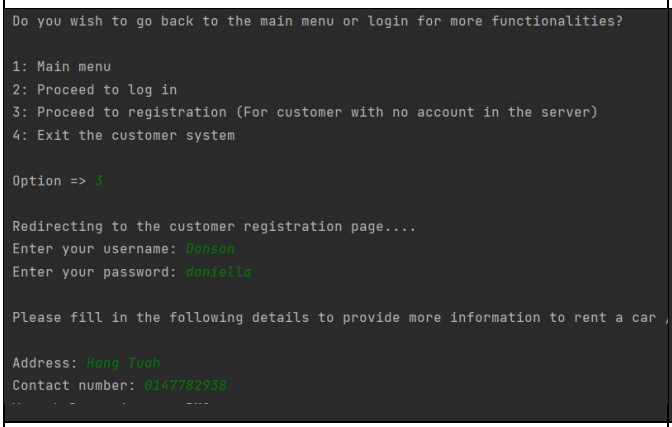
In the visitors' redirection menu, visitors can choose from number 1 to 4. Options other than number in range 1 to 4 will receive an error message and the option is requested again.

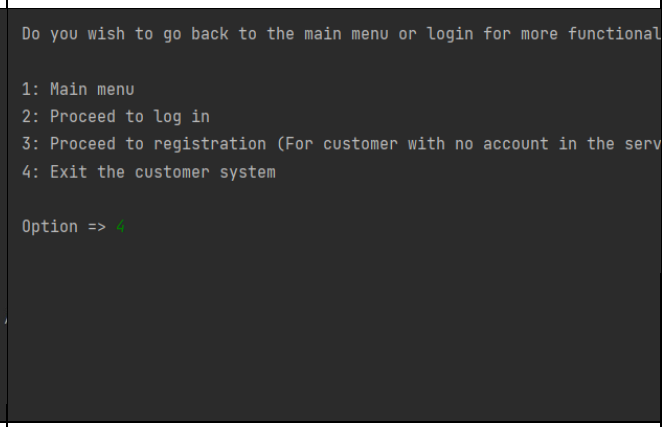



**Selecting 1**, customers will be returned to the visitors' menu.

**Selecting 2**, visitors can login to the system to access the registered customers' functionalities.

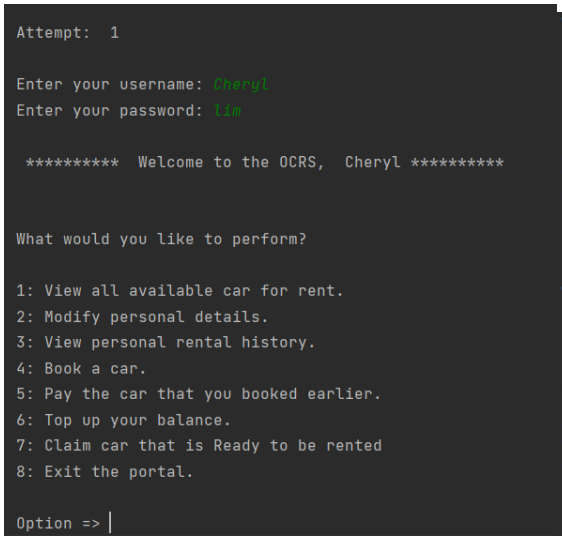


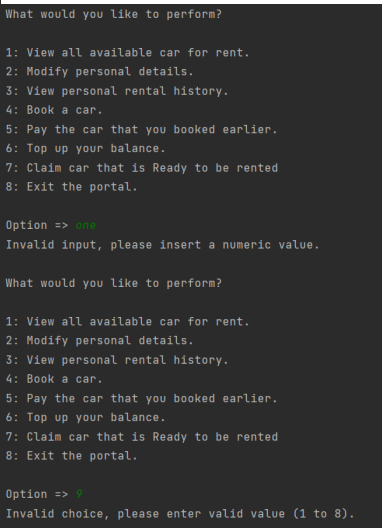




**Selecting 3**, visitors can register an account in the car rental system.

**Selecting 4** to exit the system.

## 5.4 Registered Customers



After inserting customers' valid credential username and password, customers will be provided with a customer functionalities menu. The menu contains seven unique actions and one exit system choice. The menu does not accept any nonnumeric value and number other than 1 to 8. When wrong inputs are inserted, customers are requested to insert their choices again after getting an error message.

### 5.4.1 View all available car for rent

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 1

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R003   Perodua     Kancil     DAMN520   1998  Open    10
2   R005   Audi        A4 Sedan   RW214     2002  Open    100
3   R008   Volks       Bently     SUK123    2012  Open    25
```

**Selecting 1** will display all the cars that are available for rent with status of 'Open'.

```
Do you want to proceed to book a car or return to the functionalities menu?

    1: Booking
    2: Customers functionalities menu

Option => 1

Redirecting to booking page...

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R003   Perodua     Kancil     DAMN520   1998  Open    10
2   R005   Audi        A4 Sedan   RW214     2002  Open    100
3   R008   Volks       Bently     SUK123    2012  Open    25
```

```
Do you want to proceed to book a car or return to the functionalities menu?

    1: Booking
    2: Customers functionalities menu

Option => 2

Returning to main menu...

What would you like to perform?

1: View all available car for rent.
```

```
Do you want to proceed to book a car or return to the functionalities menu?

    1: Booking
    2: Customers functionalities menu

Option => 0

Invalid input, please select either 1 or 2.


Do you want to proceed to book a car or return to the functionalities menu?

    1: Booking
    2: Customers functionalities menu

Option => two

Invalid input, please insert a numeric value.


Do you want to proceed to book a car or return to the functionalities menu?
```

After all cars displayed, customers can choose to proceed on booking a car or return to customers functionalities menu. Selecting options other than 1 or 2 will receive error message and requested to

reselect their option again. **Selecting 1** will proceed to customer booking. **Selecting 2** will return customer to functionalities menu.

### 5.4.2 Modify personal details

```
**********  Welcome to the OCRS,  Vandyck **********


What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 2
Your username is requested to check your credential for profile modification: Vandyck

username: Vandyck
password: lai
address: Johor Bahru
contact number: 0129989852
```

**Selecting 2,** customers can modify their personal details in the system. Customers will need to insert their username and the details on the customers' username is displayed after.

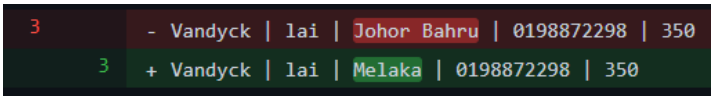
```
IMPORTANT! Note that username is immutable.
Options: [ Password ,  Address ,  Contact Number ]

Which type of data you would like to change: username

Invalid choice, username is unable to modify, please choose from available options.

IMPORTANT! Note that username is immutable.
Options: [ Password ,  Address ,  Contact Number ]

Which type of data you would like to change: bookings

Invalid input, please choose from the choices given

IMPORTANT! Note that username is immutable.
Options: [ Password ,  Address ,  Contact Number ]

Which type of data you would like to change:
```

Then, customers are allowed to change their password, address and contact number. Options other than available choices will result in error message and username is immutable in the system, customers are prohibited to modify their username.

```
IMPORTANT! Note that username is immutable.
Options: [ Password ,  Address ,  Contact Number ]

Which type of data you would like to change: address
Original data:  Johor Bahru
Replace with: Melaka

Modified details:
username: Vandyck
password: lai
address: Melaka
contact number: 0198872298


Continue modifying?

    [YES] or [NO]

Note: Selecting [NO] will navigate you back to the customer functionalities page
Option =>
```

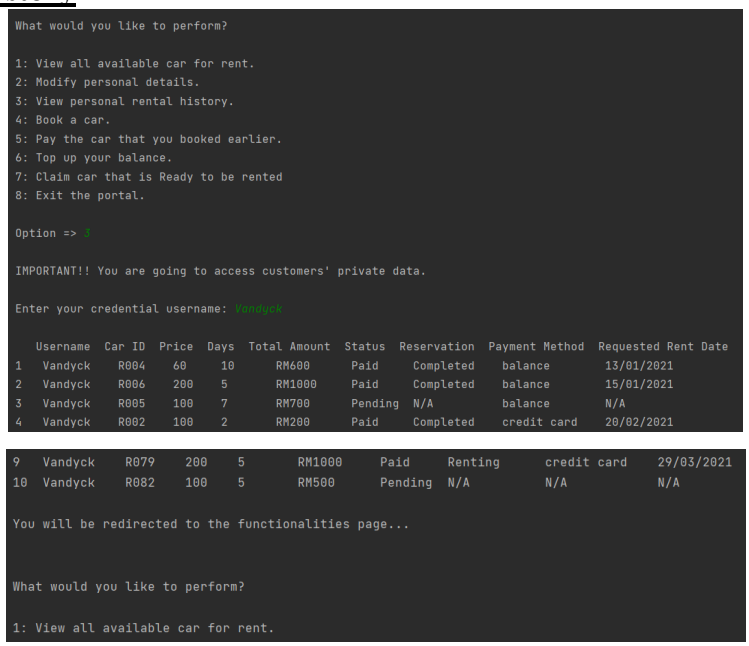
When customers choose from password, address and contact number, customers will be displayed with original data and are required to insert their desired replace information. Then after replaced, customers will be displayed with modified details. Customers will need to decide on **"YES"** for continue modification or **"NO"** to navigate back to customer functionalities menu.

**Changes in database file**

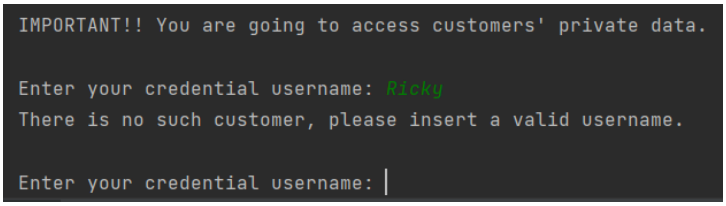



The address detail of customer "Vandyck" had been changed from 'Johor Bahru' to 'Melaka' in "customerDetails.txt" file.

### 5.4.3 View rental history




After **selecting 3** in the functionalities page, customers will be asked to insert their username to display all rental history. Then, automatically customers will be redirected to functionalities page.



When customers insert a username that does not registered in the system, they will get “There is no such customer…” message.

```
IMPORTANT!! You are going to access customers' private data.

Enter your credential username: Fiver

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Payment Method  Requested Rent Date

The rental history is empty, start to rent a car! 😊

You will be redirected to the functionalities page...


What would you like to perform?
```

When the username is registered but there is no rental history, customers will get “The rental history is empty” message. Automatically, customers will be returned to the functionalities page.

### 5.4.4 Book car

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 4

   Car ID  Car Brand  Car Model  Car Plate  Year  Status  Price
1   R003   Perodua     Kancil     DAMN520   1998  Open    10
2   R005   Audi        A4 Sedan   RW214     2002  Open    100
3   R008   Volks       Bently     SUK123    2012  Open    25
4   R009   Proton      Waja       ANGY1     2002  Open    4
```

**Selecting 4**, customers can book a car in the system. After the selection, customers will be displayed with all the available to rent cars. Then, customers can select one of the car by choosing the line of statement to book the car.

```
Select the car (line of statement) you would like to book: one

Invalid input, please insert a numeric value within the available car range.


Select the car (line of statement) you would like to book: 39
Invalid choice, choose from available line statement.
```

```
Select the car (line of statement) you would like to book: 1
Enter your username to request your booking: Vandyck
How many day(s) you would like to book it: 4

Your booking is requested...

Note: Your booking is not confirm yet.
You can make your payment by choosing option 5 to confirm your booking.

You will be redirected to the customer functionalities page.

What would you like to perform?
```

Non-numeric value and number out of the available range will result in error message and the line statement is requested again. When customers select a valid line of car, customers are requested to fill in their username and choose how many days they want to rent the car. After that, customers will receive messages of booking is request but not confirm yet and redirection to customer functionalities. Then, customers will see the functionalities page and they can choose option 5 to pay their booking confirmation.

**Changes in database file**

```
83  Zhongli | R092 | 90 | 8 | Pending | N/A | N/A | N/A
84  + Vandyck | R003 | 10 | 4 | Pending | N/A | N/A | N/A
```

The newly requested booking is added to the end of "customerBookingPayment.txt" file.

### 5.4.5 Pay car to confirm booking

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 5
```

**Selecting 5** in the functionalities page, customers can pay their car to confirm their booking.

```
Enter your username to pay for your booking confirmation: Vandyck

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Payment Method  Requested Rent Date
1  Vandyck   R005    100    7     RM700         Pending  N/A         balance         N/A          --uncompleted payment due to insufficient balance before
2  Vandyck   R038    160    6     RM960         Pending  N/A         N/A             N/A
3  Vandyck   R035    55     9     RM495         Pending  N/A         N/A             N/A
4  Vandyck   R049    20     5     RM100         Pending  N/A         N/A             N/A
5  Vandyck   R082    100    5     RM500         Pending  N/A         N/A             N/A
6  Vandyck   R003    10     4     RM40          Pending  N/A         N/A             N/A

Which booking statement (line) you would like to pay:
```

To access the booking system, customers are required to insert their username to display all the booking the customers currently have in their account that are pending for payment. From the available list, customers can select one of the booking statements to be paid by entering the line of statement correspond to the booking record.

```
Which booking statement (line) you would like to pay: 0
Invalid choice, choose from available line statement.

Which booking statement (line) you would like to pay: two

Invalid input, please insert numeric value.


Which booking statement (line) you would like to pay:
```

Selecting options other than the valid range of line statement, customers will receive an error message and the option is required again.

```
Which booking statement (line) you would like to pay: 2

IMPORTANT!!
Note:
• you will need to make full payment to confirm your booking.
• you can overwrite your previous payment method if you had not paid the booking

What would you like to use to pay the booking?

[credit card] [balance]
Option => cash
Invalid input, please choose from the available option
What would you like to use to pay the booking?
```

By selecting a valid choice of line, customers can choose a payment type either **credit card** or **balance** to make their payment. Selecting other option will receive an error message and payment type is requested again.

```
[credit card] [balance]
Option => credit card

Note that your private information in this area will not be stored in our cookies.

Enter your credit card number: 178928938921
Enter your credit card's CVV: 122
Enter your credit card's expiry date: 10/28

Transaction completed...

********** Payment details **********

Paid amount: RM960
Credit card: 178928938921

Please insert your desired rent date (DD/MM/YYYY): 11/11/2021

Date had been recorded and please be patient and wait for the preparation process.


Redirecting to customer functionalities page...


What would you like to perform?

1: View all available car for rent.
```

Selecting **"credit card"**, customers are required to insert credit card number, cvv and expiry date. Then, the transaction is completed, and customers are required to choose their desired rent date in DD/MM/YYY. Automatically, customers are redirected to customer functionalities page.

**Changes in database file**

```
27      - Vandvck | R038 | 160 | 6 | Pending | N/A | N/A | N/A
27      + Vandyck | R038 | 160 | 6 | Paid | In Queue | credit card | 11/11/2021
```

The booking statement is now “Paid” and reservation “In Queue”, payment method “credit card”, desired rent date “11/11/2021”. This record is updated in “customerBookingPayment.txt” file.

```
38      - R038 | Alfa | 159 | 0008463 | 2013 | Open | 160
38      + R038 | Alfa | 159 | 0008463 | 2013 | Booked | 160
```

The car status is changed from ‘Open’ to ‘Booked’ in “carDatabase.txt” file.

```
[credit card] [balance]
Option => balance

Your initial balance: RM20
You have to pay: RM495
Car ID: R035


Your balance is insufficient...
Your current balance: RM20

You will need to top up before paying your booking confirmation.

Redirecting to top up system....


********** Welcome to the top up system. **********

Enter your username to check your current balance:
```

```
[credit card] [balance]
Option => balance

Your initial balance: RM450
You have to pay: RM100
Car ID: R049

You had paid the booking confirmation.
Transaction completed. Your current balance is RM 350
Please insert your desired rent date (DD/MM/YYYY): 12/07/2021

Date had been recorded and please be patient and wait for the preparation process


Redirecting to customer functionalities page...


What would you like to perform?
```

Selecting **“balance”**, if the customers’ balance is lesser than the total amount required to paid, customers will get “You balance is insufficient” message. Automatically, customers will be redirected to the balance top up system. When the balance is sufficient to pay the booking statement, customers can insert their desired rent date upon their success transaction. After that, customers will be redirected to customer functionalities page.

**Changes in database file**

```
49      - R049 | Proton | Iriz | IWI2938 | 2021 | Open | 20
49      + R049 | Proton | Iriz | IWI2938 | 2021 | Booked | 20
```

The car status is changed from ‘Open’ to ‘Booked’ in “carDatabase.txt” file.

```
41      - Vandyck | R049 | 20 | 5 | Pending | N/A | N/A | N/A
41      + Vandyck | R049 | 20 | 5 | Paid | In Queue | balance | 12/07/2021
```

Statement’s status is modified to ‘Paid’, reservation ‘In Queue’, payment method ‘balance’ and desired rent date ‘12/07/2021’ in “customerBookingPayment.txt” file.

### 5.4.6 Top up balance

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 6

 ********** Welcome to the top up system. **********
```

**Selecting 6** in the functionalities page, customers can top up their balance in the system and are displayed with the "Welcome to the top up system" banner after selection.

```
 ********** Welcome to the top up system. **********

Enter your username to check your current balance: Vandyck
Your current balance:  20

Which payment method do you prefer to top up your balance with?

    1: Credit/debit card
    2: FPX online banking

Option => one

Invalid input, please insert numeric value (1 or 2).


Which payment method do you prefer to top up your balance with?

    1: Credit/debit card
    2: FPX online banking

Option => 0
Invalid input, please enter either 1 or 2.

Which payment method do you prefer to top up your balance with?
```

After the banner, customers are required to insert their username to access their private data like balance. Customers are displayed with their current balance in the system, then they can decide on which payment method to top up their balance. Selecting options other than 1 or 2, customers will get error message and they are required to choose their option again.

- **Credit / debit card top up**

```
Which payment method do you prefer to top up your balance with?

    1: Credit/debit card
    2: FPX online banking

Option => 1

 ****************************
Top up with: Credit/debit card
 ****************************
Enter your credit/debit card number: 19792979001
Enter your credit/debit card's CVV: 345
Enter your credit/debit card's expiry date: 01/22
How much do you want to top up: 30
Top up success, current balance: RM 50
```

**Selecting 1 i**n the payment method, customers will see “Top up with credit / debit card” banner. Customers are requested to insert the card number, cvv, card expiry date and top up value. Then, they will receive a “Top up success…” message.

**Changes in database file**

```
3       - Vandyck | lai | Melaka | 0198872298 | 20
3       + Vandyck | lai | Melaka | 0198872298 | 50
```

The balance is updated from 20 to 50 in “customerDetails.txt” file.

- **FPX Online banking top up**

```
Which payment method do you prefer to top up your balance with?

    1: Credit/debit card
    2: FPX online banking

Option => 2

 ****************************
Top up with: FPX online banking
 ****************************

Select your merchant:

    1: Maybank
    2: Public Bank
    3: Ambank
    4: RHB Bank
    5: CIMB Bank

Option => 6

Select your merchant:

    1: Maybank
    2: Public Bank
    3: Ambank
    4: RHB Bank
    5: CIMB Bank

Option => 6
Invalid input, please select in range 1 to 5.

Select your merchant:

    1: Maybank
    2: Public Bank
    3: Ambank
    4: RHB Bank
    5: CIMB Bank

Option => one

Invalid input, please insert numeric value.


Select your merchant:

 ****************************
Top up with: FPX online banking
 ****************************

Select your merchant:

    1: Maybank
    2: Public Bank
    3: Ambank
    4: RHB Bank
    5: CIMB Bank

Option => 1
Enter your bank account number: 1980298193081
Enter your bank account password: vandyklai

Top up with 1980298193081
How much do you want to top up: 60
Top up success, current balance: RM 110

Do you wish to top up more?
```

**Selecting 2** for FPX online banking, customers will be displayed with “Top up with: FPX online banking” banner and are required to choose a merchant to top up their balance. Select options other than number from 1 to 5, customers will receive an error message and are requested to insert their option

again. By selecting a valid merchant, customers are proceeded to insert their bank account number and the password. After that, customers can insert a top up value which is then being displayed with the top up balance.

**Changes in database file**

```
3   - Vandyck | lai | Melaka | 0198872298 | 50
3   + Vandyck | lai | Melaka | 0198872298 | 110
```

The balance of customer “Vandyck” had been updated from 50 to 110 in “customerDetails.txt” file.

```
Do you wish to top up more?

    [YES] or [NO]

Note: Selecting [NO] will redirect you back to the functionalities main menu.
Option => yup
Invalid input, please enter either [YES] or [NO].

Do you wish to top up more?

    [YES] or [NO]

Note: Selecting [NO] will redirect you back to the functionalities main menu.
Option => yes
Redirecting back to the top up section....

Enter your username to check your current balance: Vandyck
Your current balance:  50

Which payment method do you prefer to top up your balance with?
Do you wish to top up more?

    [YES] or [NO]

Note: Selecting [NO] will redirect you back to the functionalities main menu.
Option => no

Redirecting to the customer functionalities main menu...


What would you like to perform?
```

After a successful top up attempt, customers will need to decide on their redirection. **“YES”** to continue top up, **“NO”** to return to the functionalities menu. Other options will result in error message and the option is requested again.

### 5.4.7 Claim car

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 7
```

**Selecting 7** in the functionalities page, customers can claim a car that had been marked as ready by the admins upon their booking confirmation.

```
Enter your username to confirm your identity: Cheryl

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Payment Method  Requested Rent Date
1   Cheryl    R048    100    4       RM400      Paid     Ready        balance        26/04/2021
2   Cheryl    R039    250    5       RM1250     Paid     Ready        balance        15/05/2021
3   Cheryl    R062    100    5       RM500      Paid     Ready        credit card    04/05/2021
4   Cheryl    R067    100    5       RM500      Paid     Ready        balance        19/04/2021
5   Cheryl    R069    90     10      RM900      Paid     Ready        credit card    21/04/2021
```

```
Which statement you would like to claim your car (line of statement): one
Invalid input, please insert numeric value..

Which statement you would like to claim your car (line of statement): 9
Invalid choice, choose from available line statement.

Which statement you would like to claim your car (line of statement): 1

Claimed car statement:

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Payment Method  Requested Rent Date
    Cheryl    R048    100    4       RM400      Paid     Renting      balance        26/04/2021

Do you wish to claim other cars you rented?

    [YES] or [NO]

Note: Selecting [NO] will redirect you to the customer functionalities menu.
Option =>
```

Customers will get a list of rental cars that are ready to be claimed after inserting their username. After that, customers are required to insert a valid line statement selection to claim the car. Selection out of range and non-numeric value will face error message and are requested to choose again. If it is valid, customer will receive claimed car statement notice and need to decide on their redirection, **"YES"** to continue claiming cars or **"NO"** to return to the customer functionalities menu.

**Changes in database file**

```
40      - Cheryl | R048 | 100 | 4 | Paid | Ready | balance | 26/04/2021
        40 + Cheryl | R048 | 100 | 4 | Paid | Renting | balance | 26/04/2021
```

Statement is being checked as "Renting" in "customerBookingPayment.txt" file.

```
48      - R048 | Toyota | Yaris | OPO2836 | 2020 | Booked | 100
48      + R048 | Toyota | Yaris | OPO2836 | 2020 | Rented | 100
```

Car is being recorded “Rented” in “carDatabase.txt” file.

```
Enter your username to confirm your identity: Vandyck

   Username  Car ID  Price  Days  Total Amount  Status  Reservation  Payment Method  Requested Rent Date

There is no relevant data for records of cars that are available to claim.
Redirecting to the customer functionalities menu...


What would you like to perform?

1: View all available car for rent.
```

When there are relevant data, customers “There is no relevant data” message and will be redirected to the customer functionalities menu.

## 5.5 Exit System

```
********** Welcome, administrator! **********

Choose the action you want to perform:

1: Add cars to be rented out.
2: Modify car details.
3: Display records.
4: Search specific record.
5: Return a rented car.
6: Mark a car as Ready upon customer's booking confirmation.
7: OCRS Data Analytics Dashboard
8: Exit the system.

Select your action: 8
```

Admin

```
What would you like to perform?

1: View all available car for rent.
2: Modify personal details.
3: View personal rental history.
4: Book a car.
5: Pay the car that you booked earlier.
6: Top up your balance.
7: Claim car that is Ready to be rented
8: Exit the portal.

Option => 8
```

Custome

```
As a visitor, select your action.

1: View all available car for rent.
2: Membership Registration  - get access to more features.
3: Registered customers' section.
4: Exit the system.

Option  => 4
```

Visito

To exit the system, admins and registered customers can **select 8** in the functionalities menu. **Selecting 4,** visitors can exit the car rental system. After choosing the exit system option on users’ respective menus, users will need to confirm their action on either ending the program or returning to the welcome page as sometimes users might access exit system by accident.

## 6.0 Conclusion

In conclusion, to develop a well function online car rental system, it is required to make some assumptions on the system. After having those assumptions, a list of pseudocodes and flowcharts should be clarified to visualize the flow of the program works. The process of distributing the pseudocodes and flowcharts involve many logical thinking and code sequence arrangement. Using the pseudocodes and flowcharts as guidelines, the Python code had been programmed step by step and it is being tested, debug until the code functionalizes according to the program assumption. To make the online car rental system more realistic, the Python code is being integrated with three text files that act as databases to store car details, customer details and customer booking & payment statement. The execution of the entire program serves the car rental admins and the customers with respective functionalities. The program is a menu driven system that users can perform their actions based on their preferences on the options. The system had been preserved error free with the setup of multiple validations.

## 7.0 References

Algoquin College, 2003. *Flowcharting.* [Online]
Available at: http://elearning.algonquincollege.com/coursemat/mcintyb/dat2219d/lectures/27-Flowcharting.htm
[Accessed 7 June 2021].

VB Forums, 2004. *flowcharts [RESOLVED].* [Online]
Available at: https://www.vbforums.com/showthread.php?292920-flowcharts-RESOLVED
[Accessed 7 June 2021].

Subarmaniyan, S 2021, *Pseudocode and Flowchart for Function*, lecture notes, AAPP010-4-2 Programming With Python (PWP), Asia Pacific University of Technology & Innovation.

Subarmaniyan, S 2021, *Pseudocode and Flowchart for List and Enhanced For Loop*, AAPP010-4-2 Programming With Python (PWP), Asia Pacific University of Technology & Innovation.

Subarmaniyan, S 2021, *Algorithms- Control- Iteration*, AAPP010-4-2 Programming With Python (PWP), Asia Pacific University of Technology & Innovation.