DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-116
Course Title: Object Oriented Programming

Complex Engineering Problem

FE Batch 2024, Spring Semester 2025
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1		
S2		
S3		

CRITERIA AND SCALES				Marks Obtained		
				S1	S2	S3
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3)						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization?						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: How friendly is the application interface? (CPA-1, CPA-3)						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 4: How does the student performed individually and as a team member? (CPA-2, CPA-3)						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 5: Does the report adhere to the given format and requirements?						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

Contents

1 Introduction: 3
1.1 Problem Description: 3
1.2 Project Scope: 3
2 Features: 3
2.1 Object-Oriented Programming Features: 3
2.2 Distinguishing Features of Project: 4
3 Flow of Project: 4
3.1 Program Flow: 4
3.2 Class Diagram: 6
4 Technical Implementation: 7
4.1 Class Structure and OOP concepts: 7
4.2 Data Handling with Text Files: 7
4.3 Libraries Used: 7
5 Challenges Faced in The Project: 8
5.1 System Design & Architecture: 8
5.2 Data Management and File Operation: 8
5.3 Exception Handling: 8
5.4 Ensuring Consistent Performance Everywhere: 8
5.5 Business Rules: 8
5.6 User Interface: 8
6 New Python Concepts Learned: 8
6.1 Modular Programming: 8
6.2 File Handling and Display: 8
6.3 Python Libraries: 8
6.4 Debugging Techniques: 8
6.5 Data Management: 9
6.6 OOP Concepts: 9
7 Individual Contributions: **Error! Bookmark not defined.**
7.1 Usman Rasheed Siddiqui (CS-24038): **Error! Bookmark not defined.**
7.2 Huzaifa Hanif (CS-24039): **Error! Bookmark not defined.**
8 Potential Future Expansions: 9
9 Test Cases: 10
9.1.1 Test Case 1: 10
9.1.2 Test Case 2: 11
9.1.3 Test Case 3: 12

1 Introduction:

1.1 Problem Description:

The objective of this project is to develop a Car Rental System using Python. A system that allows a user to rent different types of cars according to the car's availability for a period and return it. The Car Rental System will allow users to create an account, rent a car, return it, update their information and view car specifications within the application.

This system must allow users to:

- ✓ Create an account with basic information
- ✓ Login into their respective account
- ✓ Rent a car based on availability (Not more than one)
- ✓ Review rental history and update balance

The system should:

- ✓ Provide user with a proper interface
- ✓ Store and display list of different cars with their rental price per day and other features
- ✓ Make online payment from user's balance
- ✓ Maintain rental history

The system must allow admin to:

- ✓ Print a report about all customers and their current rentals
- ✓ Add a car and remove a car from the system
- ✓ Print a report about all cars that are currently reserved

1.2 Project Scope:

This project demonstrates the application of programming concepts that were learnt during the course. It emphasizes:

- ➢ Modular Programming
- ➢ File handling for different data storage
- ➢ An interactive CLI for users to perform respective actions

2 Features:

2.1 Object-Oriented Programming Features:

The following Object-Oriented Features were implemented:

- ✓ Inheritance
- ✓ Association (Composition)
- ✓ Method Overriding
- ✓ Abstract Class

- ✓ Exception Handling

2.2 Distinguishing Features of Project:

Key features that distinguish this project include:

- ✓ **Exception Handling Class:** A separate Exception class “Exceptions” for user-defined exception classes.
- ✓ **Penalty System:** A proper penalty system if user does not return the car on time.
- ✓ **Checking Status:** Allows user to check his/her status i.e. rented car, balance, rented car name etc.
- ✓ **Rental History:** Allows user to see a specific car’s rental history.
- ✓ **Receipt Generation:** Generates a receipt for user after he/she rents a car.
- ✓ **View Specific Car:** Allows user to quickly view detailed information of a specific available car without having to hassle through the entire table.
- ✓ **Feedback:** Allows user to give feedback that can be viewed by the administrator.
- ✓ **Removing Specific Car:** Allows admin to remove a specific car from the system using its ID.
- ✓ **Quitting Without Any Problem:** Allows users and admin to quit any action safely without generating any errors.

3 Flow of Project:

The Car Rental System is designed using a modular approach and a flexible design, enabling it to be expanded and adapt to future needs.

3.1 Program Flow:

- ➔ **Start Application:** The application starts using the main.py.
- ➔ **Main Interface:** The main interface shows up having options to Login, Create Account, Login as Administrator and Exit.

The user can select login to login into his/her account, create an account and if the user is an admin (in our case there is only one admin account) he can use Login as Administrator.

- ➔ **Create an Account:** The user can create an account using some basic information like name, email, password, and address. Once an account is created the user then needs to login to his account to get full access.

- ➔ **Login:** Customer/Admin can have 3 attempts to login to their accounts otherwise the program will return back to the main menu.

- o **As Customer:**

The user can login to his/her account using email and password. Once logged in he/she is shown with the Customer Interface.

- ▪ **Customer Interface:** Customer Interface contains the following options:
  1. *Rent a Car:* To rent a Car

2. *Return Car:* To return a car
3. *Check Status:* To check his status
4. *Update Balance:* To update his balance
5. *Update Your Information:* To update his information like address and password
6. *View All Available Cars:* To view all the available cars in the system
7. *View Specific Available Car:* To check details of a specific available car
8. *View Your Rental History:* Review his/her own rental history
9. *View A Car's Rental History:* Review a specific car’s rental history
10. *Have any complain? Feel free to write it in your feedback!:* Give feedback
11. *Exit:* To exit to main menu

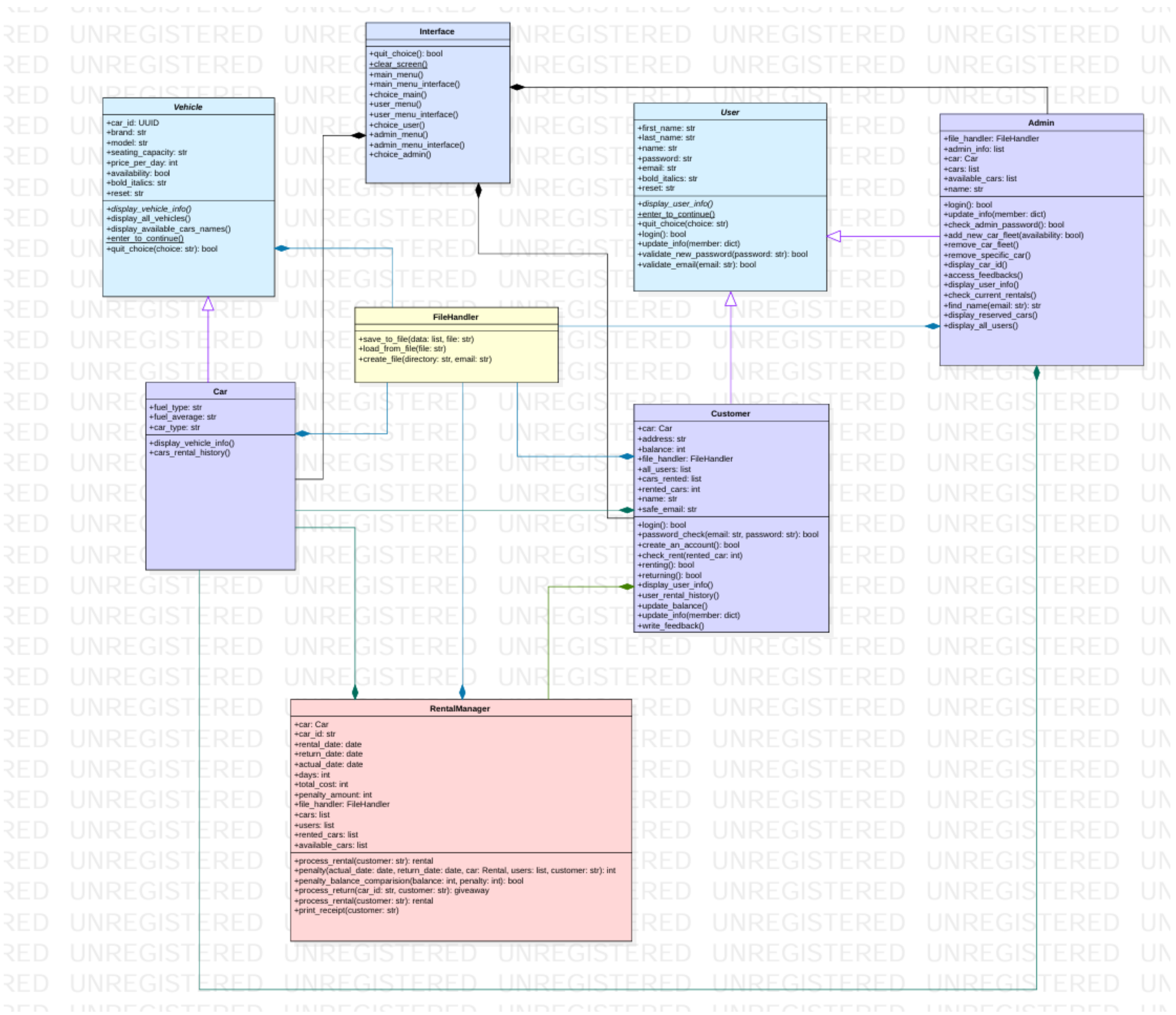
- **As Administrator:**

The administrator can login using his email address and password. Once logged in he is shown with the Admin Interface.

- **Admin Interface:** Admin Interface contains the following options:
  1. *Add Car Fleet:* To add a whole car fleet
  2. *Remove Car Fleet:* To remove a whole car fleet
  3. *Remove Specific Car:* To remove a specific car from system
  4. *Check All Customers:* To generate a report of all customers
  5. *Check Specific Customer Rental History:* To check a specific customer’s rental history
  6. *Check Current Rentals:* To check current rentals and customers
  7. *Currently Reserved Cars:* To check all the currently reserved cars
  8. *View All Available Cars:* To view all available cars
  9. *View All Available Car IDs:* To view all available car’s IDs.
  10. *View Specific Car:* To view specific car
  11. *Access Feedbacks:* To access feedback from users
  12. *Update Password:* To update password
  13. *Exit:* To exit to main menu

➔ **File Handler:** A file handler handles the saving, creating and loading from file.
➔ **Exceptions:** The exceptions module handles all the user-defined exceptions

3.2 Class Diagram:



- **Abstract Classes:**
  - *User and Vehicle are abstract classes.*
    - They provide a blueprint for derived classes without being instantiated themselves.
- **Inheritance:**
  - *Admin and Customer inherit from User; Car inherits from Vehicle.*
    - Admin and Customer are specific types of User; Car is specific type of Vehicle.
- **Composition:**
  - *Interface composes Car, Customer, and Admin.*
    - To directly handle users, admin and cars during system operations.

  - *FileHandler is composed in Car, Customer, Admin, Vehicle, and RentalManager.*
    - To manage saving and loading data for these classes.
  - *Car is composed in RentalManager, Admin, and Customer.*

    - Because they need to access and work with cars for rentals and management otherwise, cars have no meaning in the system.

4 Technical Implementation:

This heading includes how different concepts of Python were implemented to achieve a fully functional Car Rental System.

4.1 Class Structure and OOP concepts:

This system is built using Python OOP principles. It uses:

- **Abstract Classes:** to define shared attributes and define common features.
- **Inheritance:** to allow child classes to reuse or extend from parent classes.
- **Composition:** to give classes ownership of others where direct interaction is needed.
- **Method Overriding:** allows sub classes to customize or extend parent class's methods while keeping a common interface.

4.2 Data Handling with Text Files:

This system uses plain text files for data storage. This is how it is implemented:

- Renting and Returning Car: Car and customer data are updated and stored during the rental and returning process.
- Adding and Removing Cars: Cars are added or removed from the txt files dynamically through admin actions.
- Updating Balance and other Information: The system locates and rewrites specific entries to modify data accurately.
- Displaying Data: Reads and displays text files to present cars, users and other relevant information.

4.3 Libraries Used:

Several built-in libraries support the system functionality:

- **uuid:** Generates a unique ID for every car
- **re:** Validates user inputs (e.g. password, and email).
- **time:** Adds delays for user interface feedback and a proper user experience.
- **os:** Helps find, create, read and write to appropriate files in the system. Also helps to clear the screen of every new functionality.

5 Challenges Faced in The Project:

While coding, we encountered several challenges. Most of them are listed below:

5.1 System Design & Architecture:

- Defining class structures with proper encapsulation
- Managing module imports and interaction between different classes and files without errors.

5.2 Data Management and File Operation:

- Implementing changes to data within the files during different operations without data corruption.
- Loading data from files without data corruption.

5.3 Exception Handling:

- Validating user input to prevent crashes from invalid entries.

5.4 Ensuring Consistent Performance Everywhere:

- Ensuring that the project performs as per the requirements and the way it is made on any system.

5.5 Business Rules:

- Figuring out how much to charge for late car returns.
- Making rental receipts and car lists look neat and organized.

5.6 User Interface:

- Designing an intuitive menu system that contains all functionalities with ease of use.
- Formatting output displays for clarity and proper presentation.

6 New Python Concepts Learned:

6.1 Modular Programming:

- Learned to organize code into separate modules for better structure
- Understood how to import and reuse functions across different files

6.2 File Handling and Display:

- Gained experience in reading/writing/creating files and formatting outputs clearly.
- Practiced displaying data in user-friendly tables and layouts

6.3 Python Libraries:

- Used re for input validation and os for file path operation and clear screen.
- Applied time for date calculation and assigning and uuid to generate unique IDs.

6.4 Debugging Techniques:

- Learned to properly isolate and fix bugs in larger programs.

6.5 Data Management:

- Developed skills in maintaining data consistency across multiple operations.

6.6 OOP Concepts:

- Increased understanding of classes, inheritance and encapsulation.

7 Individual Contributions:

7.1 Usman Rasheed Siddiqui (CS-24038):

- ➔ Build the Vehicle, User, Customer, Rental Manager, File Handler class
- ➔ Implemented text file handling for data preservation, managing renting, returning, adding, removing cars etc.
- ➔ Implemented modular programming for neat coding and ease of understanding.
- ➔ Made ReadMe file.

7.2 Huzaifa Hanif (CS-24039):

- ➔ Build the Car, Admin and Interface Classes and integrated them into the modular system
- ➔ Contributed to the project documentation, including writing key sections of the report and ensuring proper formatting and organization of the project details.

8 Potential Future Expansions:

Several features can enhance the functionality of the system. These include:

- ➔ **Hiring a Driver:**
  Introduce an option for users to hire a professional driver along with the vehicle.

- ➔ **Account Deletion:**
  Allows users to permanently delete their account and associated data.

- ➔ **Early Return Refunds:**
  Providing partial refunds if users return the vehicle before the scheduled return date.

- ➔ **Categorizing Vehicles:**
  Expanding the vehicle options by adding categories like Sedan, SUV and Pick-Up.

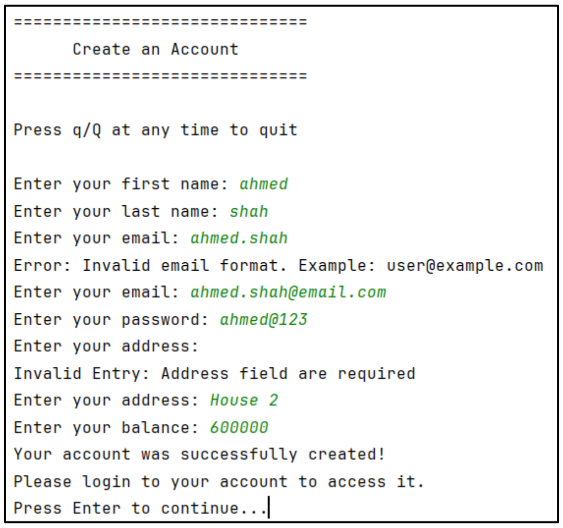
- ➔ **Multiple Rental Durations:**
  Provide various rental duration options, which include hourly, daily and weekly rentals.

- ➔ **Admin Review for Late Return Reason:**
  Add a process where admin can review and confirm a valid reason for delayed vehicle return.

9 Test Cases:

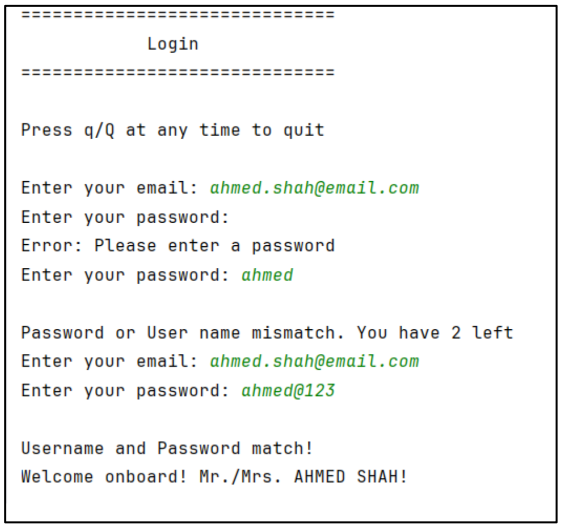
9.1.1 Test Case 1:

- ✓ Objective: Verify that the system allows customers to create accounts and then login



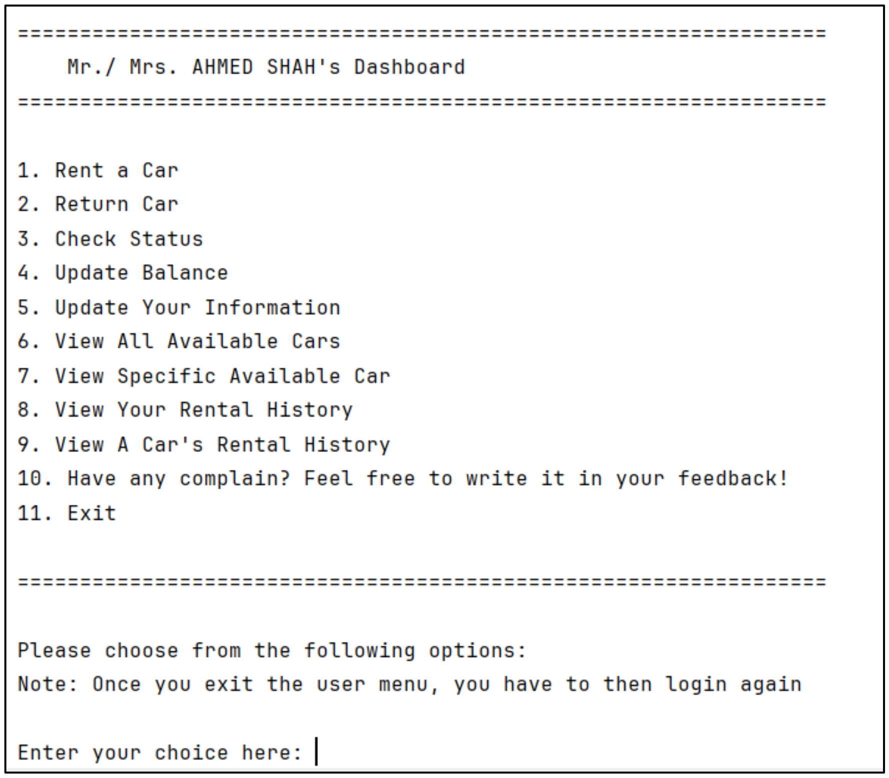
```
==============================
      Create an Account
==============================

Press q/Q at any time to quit

Enter your first name: ahmed
Enter your last name: shah
Enter your email: ahmed.shah
Error: Invalid email format. Example: user@example.com
Enter your email: ahmed.shah@email.com
Enter your password: ahmed@123
Enter your address:
Invalid Entry: Address field are required
Enter your address: House 2
Enter your balance: 600000
Your account was successfully created!
Please login to your account to access it.
Press Enter to continue...
```

Figure 2: Creating an Account



```
==============================
            Login
==============================

Press q/Q at any time to quit

Enter your email: ahmed.shah@email.com
Enter your password:
Error: Please enter a password
Enter your password: ahmed

Password or User name mismatch. You have 2 left
Enter your email: ahmed.shah@email.com
Enter your password: ahmed@123

Username and Password match!
Welcome onboard! Mr./Mrs. AHMED SHAH!
```

Figure 3: Login to Created Account

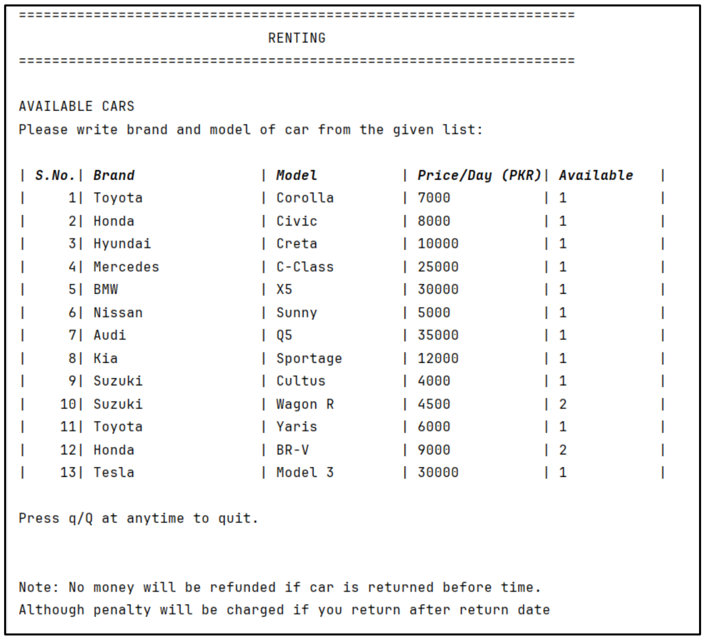


```
=================================================================
    Mr./ Mrs. AHMED SHAH's Dashboard
=================================================================

1. Rent a Car
2. Return Car
3. Check Status
4. Update Balance
5. Update Your Information
6. View All Available Cars
7. View Specific Available Car
8. View Your Rental History
9. View A Car's Rental History
10. Have any complain? Feel free to write it in your feedback!
11. Exit

=================================================================

Please choose from the following options:
Note: Once you exit the user menu, you have to then login again

Enter your choice here:
```

Figure 1: Customer's Dashboard After Successful Login

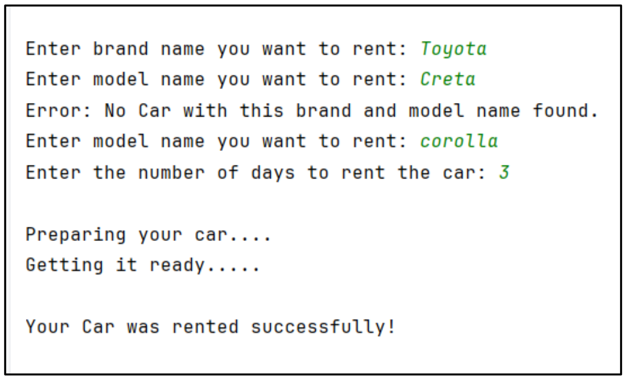
9.1.2 Test Case 2:

- ✓ Objective: Verify that the system allows customers to rent a car.



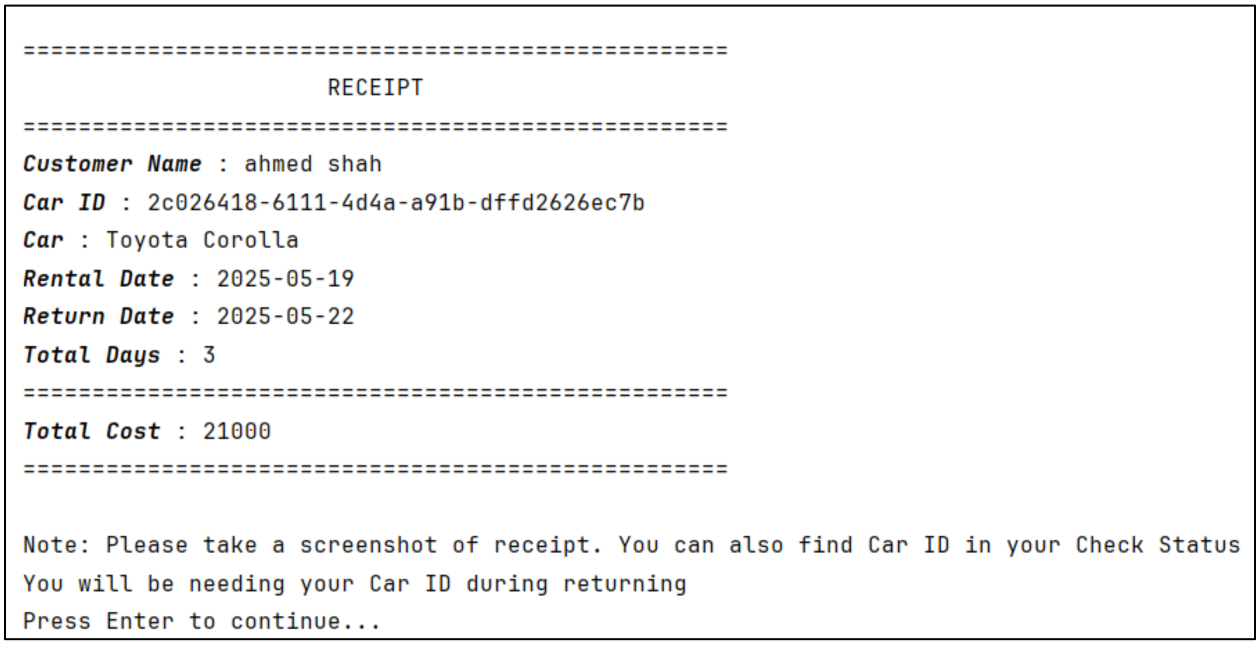
```
=====================================================================
                              RENTING
=====================================================================

AVAILABLE CARS
Please write brand and model of car from the given list:

| S.No.| Brand              | Model       | Price/Day (PKR)| Available    |
|     1| Toyota             | Corolla     | 7000           | 1            |
|     2| Honda              | Civic       | 8000           | 1            |
|     3| Hyundai            | Creta       | 10000          | 1            |
|     4| Mercedes           | C-Class     | 25000          | 1            |
|     5| BMW                | X5          | 30000          | 1            |
|     6| Nissan             | Sunny       | 5000           | 1            |
|     7| Audi               | Q5          | 35000          | 1            |
|     8| Kia                | Sportage    | 12000          | 1            |
|     9| Suzuki             | Cultus      | 4000           | 1            |
|    10| Suzuki             | Wagon R     | 4500           | 2            |
|    11| Toyota             | Yaris       | 6000           | 1            |
|    12| Honda              | BR-V        | 9000           | 2            |
|    13| Tesla              | Model 3     | 30000          | 1            |

Press q/Q at anytime to quit.


Note: No money will be refunded if car is returned before time.
Although penalty will be charged if you return after return date
```

Figure 5: Renting Interface



```
Enter brand name you want to rent: Toyota
Enter model name you want to rent: Creta
Error: No Car with this brand and model name found.
Enter model name you want to rent: corolla
Enter the number of days to rent the car: 3

Preparing your car....
Getting it ready.....

Your Car was rented successfully!
```

Figure 4: User Inputs

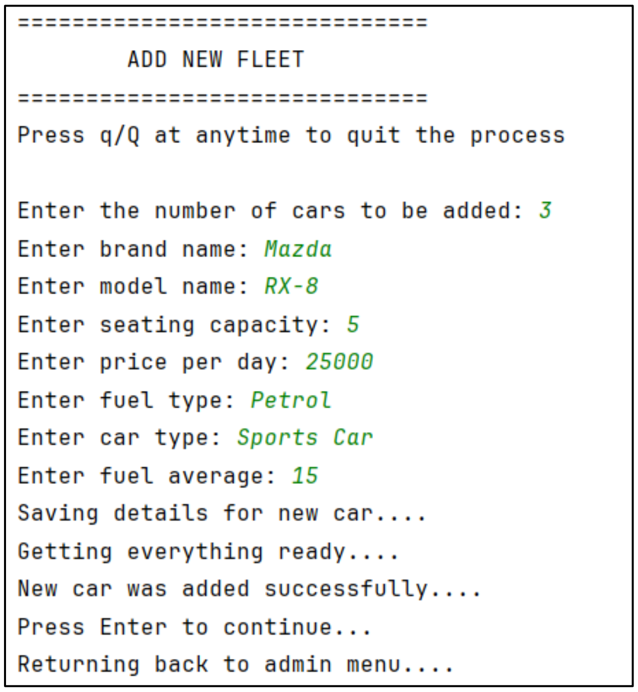


```
===================================================
                      RECEIPT
===================================================
Customer Name : ahmed shah
Car ID : 2c026418-6111-4d4a-a91b-dffd2626ec7b
Car : Toyota Corolla
Rental Date : 2025-05-19
Return Date : 2025-05-22
Total Days : 3
===================================================
Total Cost : 21000
===================================================

Note: Please take a screenshot of receipt. You can also find Car ID in your Check Status
You will be needing your Car ID during returning
Press Enter to continue...
```

Figure 6: Receipt Generation

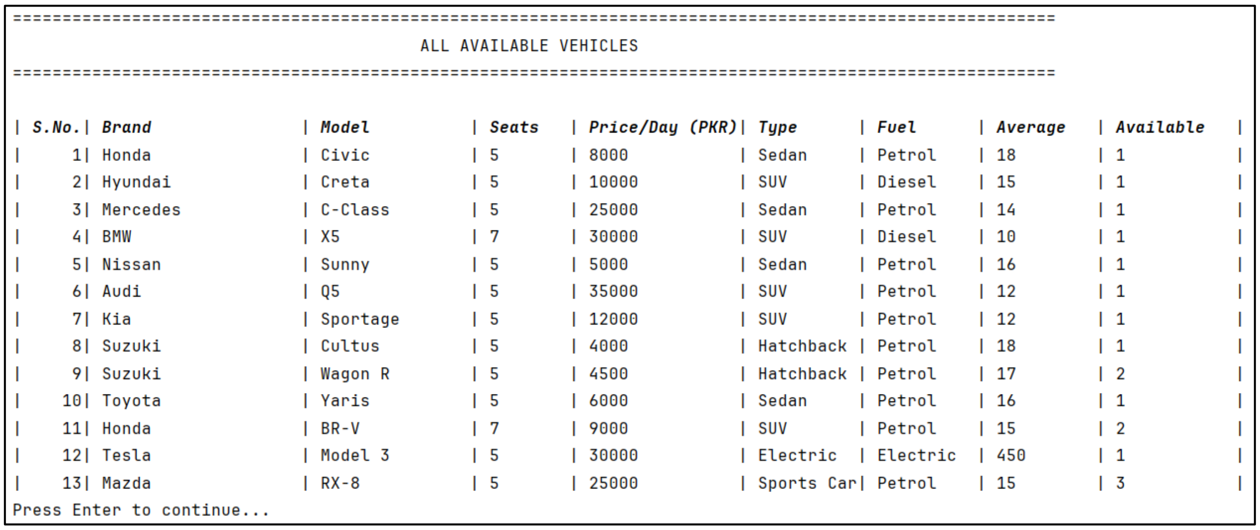
9.1.3 Test Case 3:

- ✓ Objective: Verify that the system allows admin to add a car fleet.



```
==============================
        ADD NEW FLEET
==============================
Press q/Q at anytime to quit the process

Enter the number of cars to be added: 3
Enter brand name: Mazda
Enter model name: RX-8
Enter seating capacity: 5
Enter price per day: 25000
Enter fuel type: Petrol
Enter car type: Sports Car
Enter fuel average: 15
Saving details for new car....
Getting everything ready....
New car was added successfully....
Press Enter to continue...
Returning back to admin menu....
```

Figure 8: Interface to add a new car fleet



```
==========================================================================================================
                                         ALL AVAILABLE VEHICLES
==========================================================================================================

| S.No.| Brand                | Model           | Seats     | Price/Day (PKR)| Type       | Fuel       | Average     | Available     |
|     1| Honda                | Civic           | 5         | 8000           | Sedan      | Petrol     | 18          | 1             |
|     2| Hyundai              | Creta           | 5         | 10000          | SUV        | Diesel     | 15          | 1             |
|     3| Mercedes             | C-Class         | 5         | 25000          | Sedan      | Petrol     | 14          | 1             |
|     4| BMW                  | X5              | 7         | 30000          | SUV        | Diesel     | 10          | 1             |
|     5| Nissan               | Sunny           | 5         | 5000           | Sedan      | Petrol     | 16          | 1             |
|     6| Audi                 | Q5              | 5         | 35000          | SUV        | Petrol     | 12          | 1             |
|     7| Kia                  | Sportage        | 5         | 12000          | SUV        | Petrol     | 12          | 1             |
|     8| Suzuki               | Cultus          | 5         | 4000           | Hatchback  | Petrol     | 18          | 1             |
|     9| Suzuki               | Wagon R         | 5         | 4500           | Hatchback  | Petrol     | 17          | 2             |
|    10| Toyota               | Yaris           | 5         | 6000           | Sedan      | Petrol     | 16          | 1             |
|    11| Honda                | BR-V            | 7         | 9000           | SUV        | Petrol     | 15          | 2             |
|    12| Tesla                | Model 3         | 5         | 30000          | Electric   | Electric   | 450         | 1             |
|    13| Mazda                | RX-8            | 5         | 25000          | Sports Car| Petrol     | 15          | 3             |
Press Enter to continue...
```

Figure 7: Evidence of the New Fleet Added