University Of Mauritius

Faculty of Information, Communication and Digital Technologies

Car Rental System

Prepared by :

BEEDASSY Nirvana Luxmi – 2413850

NARAIN Isha – 2413288

Course : BSc (Hons) Computer Science – level 1

Module : ICDT1201Y Computer Programming

Lecturer Name : Mr. Selvanaden Sathan

Date : 10th March 2025

Table of Contents

1.0 Introduction

1.1 The Problem

Car rental businesses often struggle with manual record-keeping, leading to errors in tracking vehicles, calculating rental costs, and maintaining rental history. Without a centralized system, issues like overbooking, misplaced records, and processing delays occur. An automated solution is needed for smoother operations and improved customer experience.

1.2 Proposed Solution

This Car Rental System provides a digital platform for managing rentals, customer transactions, and vehicle inventory. Built using Python's OOP principles, it is modular and scalable. Data is stored in text files for persistence, and the user-friendly interface allows efficient rental operations, reducing administrative workload.

1.3 Scope

The Car Rental System includes the following functionalities:

- **Car Management**: Add, remove, and list cars.
- **Customer Management**: Register and remove customers.
- **Rental Transactions**: Rent cars, calculate costs, and process returns.
- **Rental History**: Track rental and return dates.
- **Data Persistence**: Store data in text files.

1.4 Distribution of Task

To efficiently complete the project, tasks were divided among group members as follows:

- **Beedassy Nirvana Luxmi (2413850):** Designed and implemented the core classes, including Vehicle, Car, Person, and Customer, as well as data handling functionalities.
- **Narain Isha (2413288):** Developed the rental transaction process, implemented file handling for data storage, and conducted testing to ensure system reliability.

2.0 Solution Design

2.1 Pseudocode

```
BEGIN
    LOAD data from text files (cars.txt, customers.txt, rental_history.txt)
    REPEAT
        OUTPUT  " Menu Options : "
            1.  Add car
            2. Remove car
            3. List cars
            4. Add customers
            5. Remove customers
            6. List customers
            7. Rent car
            8. Return car
            9. View Rental History
            10. Exit
        OUTPUT " Enter user choice : "
        INPUT User choice
        IF User choice = 1 THEN
            OUTPUT " Enter Car Details : "
            INPUT car model, year, color, daily rate
            VALIDATE inputs
            CREATE a new car object
            ADD car to the list
            SAVE updated car list to file
        IF User choice = 2 THEN
            DISPLAY list of cars
            INPUT car index to remove
```

VALIDATE index

REMOVE car from list

SAVE updated car list to file

IF User choice = 3 THEN

DISPLAY list of available cars

IF User choice = 4 THEN

OUTPUT “ Enter Customer Details : ”

INPUT customer name and contact number

VALIDATE inputs

CREATE a new customer object

ADD customer to the list

SAVE updated customer list to file

IF User choice = 5 THEN

DISPLAY list of customers

INPUT customer name to remove

VALIDATE name

REMOVE customer from list

SAVE updated customer list to file

IF User choice = 6 THEN

DISPLAY list of registered customers

IF User choice = 7 THEN

DISPLAY list of available cars

INPUT customer name and car index

VALIDATE inputs

CHECK if car is available

CALCULATE rental cost based on days

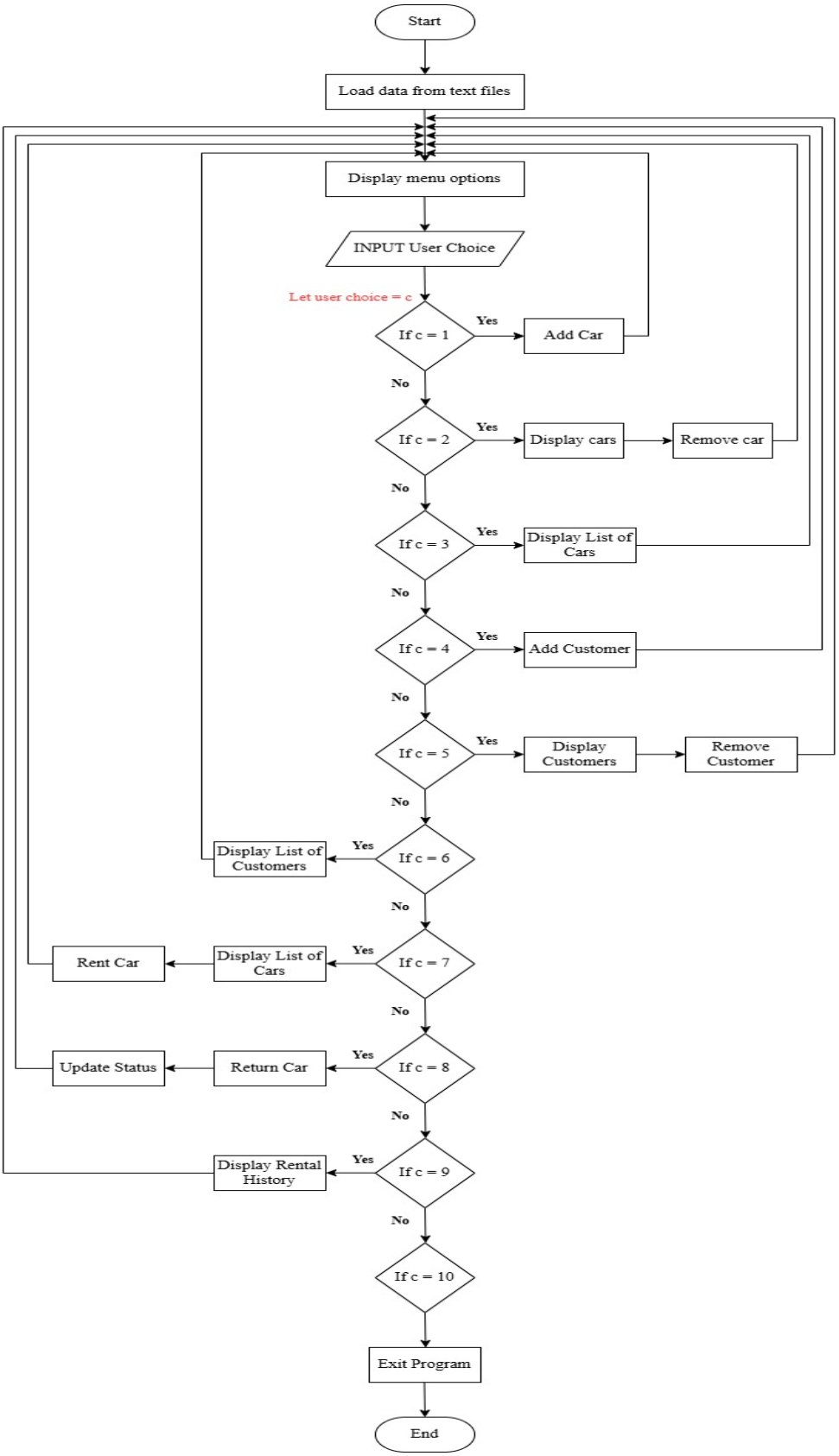
UPDATE car status to rented

SAVE rental transaction to history file

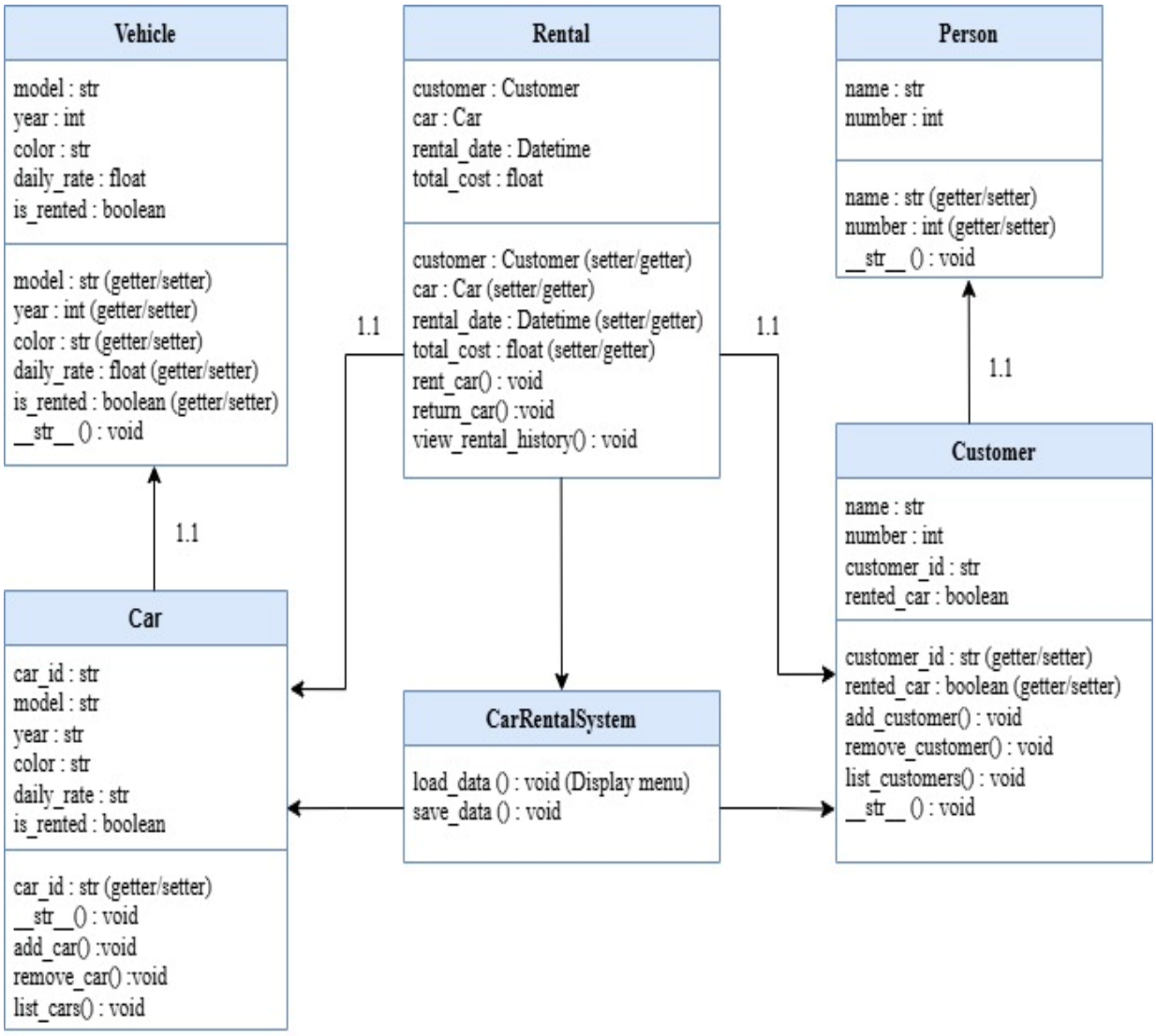
```
        IF User choice = 8 THEN
            INPUT customer name
            FIND rented car associated with customer
            UPDATE car status to available
            RECORD return date in history file
            SAVE updated data
        IF User choice = 9 THEN
            DISPLAY Rental History
        IF User choice = 10 THEN
            OUTPUT " Exiting…"
            SAVE all data
            TERMINATE program
        ELSE
            OUTPUT " Invalid choice, please try again "
END
```

2.2 Flowchart and UML

Flowchart :



UML :


Vehicle
model : str
year : int
color : str
daily_rate : float
is_rented : boolean
model : str (getter/setter)
year : int (getter/setter)
color : str (getter/setter)
daily_rate : float (getter/setter)
is_rented : boolean (getter/setter)
__str__ () : void
Rental
customer : Customer
car : Car
rental_date : Datetime
total_cost : float
customer : Customer (setter/getter)
car : Car (setter/getter)
rental_date : Datetime (setter/getter)
total_cost : float (setter/getter)
rent_car() : void
return_car() :void
view_rental_history() : void
Person
name : str
number : int
name : str (getter/setter)
number : int (getter/setter)
__str__ () : void
1.1
1.1
1.1
1.1
Car
car_id : str
model : str
year : str
color : str
daily_rate : str
is_rented : boolean
car_id : str (getter/setter)
__str__() : void
add_car() :void
remove_car() :void
list_cars() : void
CarRentalSystem
load_data () : void (Display menu)
save_data () : void
Customer
name : str
number : int
customer_id : str
rented_car : boolean
customer_id : str (getter/setter)
rented_car : boolean (getter/setter)
add_customer() : void
remove_customer() : void
list_customers() : void
__str__ () : void

3.0 Implementation and Testing

3.1 System requirements

The Car Rental System is developed using Python 3.13.2 and utilizes text file handling for data storage. No external databases or additional dependencies are required, making it lightweight and easy to deploy. The system runs on any operating system that supports Python.

3.2 Implementation details

The system is implemented using Python's Object-Oriented Programming (OOP) paradigm, ensuring modularity and code reusability. It consists of three main components: Vehicle Management, Customer Management, Rental Management and CarRentalSystem Management.

1. Vehicle Management

This module handles the addition, removal, and display of cars available for rental. It includes the following:

- **Vehicle Class (Base Class):**
  - Stores general vehicle attributes such as model, year, color, daily rental rate, and rental status.
  - Implements getter and setter methods to ensure data encapsulation.
  - Keeps track of the total number of vehicles using a class variable.
- **Car Class (Subclass):**
  - Extends the Vehicle class by adding a unique car ID.
  - Ensures that each car in the system has a distinct identifier.
  - Provides an overridden string representation to enhance readability
- **Methods:**
  - add_car(): Allows administrators to add new cars while ensuring uniqueness in ID format.
  - remove_car(): Enables the removal of a car from the system.
  - list_cars(): Displays all available cars along with their details.

2. Customer Management

This module manages customer details, including their personal information and rental history. It includes:

- **Person Class (Base Class):**
    - Stores general personal information such as name and contact number.
    - Implements getter and setter methods to enforce data integrity.
- **Customer Class (Subclass):**
    - Extends the Person class by adding a unique customer ID and tracking the currently rented car.
    - Ensures that customer details are correctly stored and retrieved.
    - Includes validation to prevent duplicate customer IDs.
- **Methods:**
    - add_customer(): Allows new customers to be added with validation checks for ID uniqueness and proper format.
    - remove_customer(): Removes a customer from the system if they have no active rentals.
    - list_customers(): Displays a list of all registered customers.

3. Rental Management

This module handles the core functionality of renting and returning cars, ensuring proper record-keeping and pricing calculations.

- **Methods:**
    - rent_car(): Checks if car is available for rent,validates customer identity and calculates total cost based on the daily rate and duration.
    - return_car(): Verifies customer's rental status, modifies the rental history to record the return date.

4. CarRentalSystem Management

This module is the core component of the car rental system. It maintains data management in the application.

- Methods:
    - load_data(): Load car and customer data from text files when the program starts.It also reads cars.txt to populate cars list with Car objects and customers.txt to populate the customers list with Customer objects.
    - save_data(): Save car and customer data to text files when the program exits. It writes the cars list to cars.txt and customers list to customers.txt .

3.3 Test Plan and Scenarios

Scenario 1 : Test car and customer creation

Adding a car to the system:

```
--------Car Rental System--------
1. Add Car
2. Remove Car
3. List Cars
4. Add Customer
5. Remove Customer
6. List Customers
7. Rent Car
8. Return Car
9. View Rental History
10. Exit
Enter your choice : 1

Enter car ID: V001
Enter the car model: Honda Jazz
Enter the car year: 2024
Enter the car color: White
Enter daily rental price: 1500
Car added successfully!
```

Adding a customer to the system:

```
--------Car Rental System--------
1. Add Car
2. Remove Car
3. List Cars
4. Add Customer
5. Remove Customer
6. List Customers
7. Rent Car
8. Return Car
9. View Rental History
10. Exit
Enter your choice : 4

Enter customer ID: C001
Enter customer name: John Doe
Enter customer number: 12345678
Customer added successfully!
```

Displaying available cars:

```
Available Cars:

ID: V001 - Honda Jazz (2024) - White - Rs1500.0/day - Available
```

Displaying customers

```
Customers:
ID: C001 - John Doe - 12345678 - No Car
```

Validation:

```
Enter car ID: 1234
Error: Car ID must be in the format V followed by 3 digits (e.g., V001).
Enter car ID: V002
Enter the car model: BMW
Enter the car year: 2030
Error: Year cannot be in the future. Current year is 2025.
Enter the car year: 2020
Enter the car color: Black
Enter daily rental price: w
Error: Daily rental price must be a valid number.
Enter daily rental price: 10000
Car added successfully!
```

```
Enter customer ID: 1234
Error: Customer ID must be in the format C followed by 3 digits (e.g., C001).
Enter customer ID: C002
Enter customer name: 123
Error: Customer name cannot be empty or contain digits.
Enter customer name: Tom
Enter customer number: 123456789
Error: Customer number must be exactly 8 digits.
Enter customer number: 23456789
Customer added successfully!
```

Scenario 2 : Rent and return car

Display available cars:

```
Available Cars:

ID: V001 - Honda Jazz (2020) - Grey - Rs1500.0/day - Available
ID: V002 - Toyota Yaris (2019) - White - Rs1300.0/day - Available
ID: V003 - Suzuki Swift (2023) - Black - Rs1800.0/day - Available
ID: V004 - BMW M5 (2025) - Red - Rs10000.0/day - Available
ID: V005 - Audi A5 (2015) - Blue - Rs3000.0/day - Available
```

Display customers:

```
Customers:
ID: C001 - John Doe - 12345678 - No Car
ID: C002 - Tom Smith - 23456789 - No Car
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - No Car
```

Renting a car :

```
  Khushi Bee has rented Audi A5 for 4 days.
  Total cost: Rs12000.0
  Rental Date: 10/03/2025
  Expected Return Date: 14/03/2025
```

Customers list updated:

```
Customers:
ID: C001 - John Doe - 12345678 - No Car
ID: C002 - Tom Smith - 23456789 - No Car
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - Rented: Audi A5
```

Rental history is updated (with car not returned yet):

```
[Already Returned] Joe,Honda,2020,2025-03-10,2025-03-12,2,Rs3000.0
[Already Returned] John Doe,Honda Jazz,2020,2025-03-10,2025-03-12,2,Rs3000.0
Khushi Bee,Audi A5,2015,2025-03-10,2025-03-14,4,Rs12000.0
```

Returning a car :

```
Available Cars:

ID: V001 - Honda Jazz (2020) - Grey - Rs1500.0/day - Available
ID: V002 - Toyota Yaris (2019) - White - Rs1300.0/day - Available
ID: V003 - Suzuki Swift (2023) - Black - Rs1800.0/day - Available
ID: V004 - BMW M5 (2025) - Red - Rs10000.0/day - Available
ID: V005 - Audi A5 (2015) - Blue - Rs3000.0/day - Rented

--------Car Rental System--------
1. Add Car
2. Remove Car
3. List Cars
4. Add Customer
5. Remove Customer
6. List Customers
7. Rent Car
8. Return Car
9. View Rental History
10. Exit
Enter your choice : 8


Customers:
ID: C001 - John Doe - 12345678 - No Car
ID: C002 - Tom Smith - 23456789 - No Car
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - Rented: Audi A5
Enter your customer ID : C005
Khushi Bee has returned Audi A5.
```

Customers list updated :

```
Customers:
ID: C001 - John Doe - 12345678 - No Car
ID: C002 - Tom Smith - 23456789 - No Car
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - No Car
```

Rental history is updated (with returned car):

```
[Already Returned] Joe,Honda,2020,2025-03-10,2025-03-12,2,Rs3000.0
[Already Returned] John Doe,Honda Jazz,2020,2025-03-10,2025-03-12,2,Rs3000.0
[Already Returned] Khushi Bee,Audi A5,2015,2025-03-10,2025-03-14,4,Rs12000.0
```

Validation :

```
Enter the car ID of the car to rent : 1234
Error: Car ID must be in the format V followed by 3 digits (e.g., V001).
Enter the car ID of the car to rent : V001
Enter your customer ID : 1234
Error: Customer ID must be in the format C followed by 3 digits (e.g., C001).
Enter your customer ID : C001
Enter number of days to rent : abc
Error: Number of days must be an integer.
Enter number of days to rent : 5
```

4.0 Conclusion

4.1 Achievements

Successfully implemented a functional car rental system.

Ensured data is correctly stored and retrieved using text files.

Automated booking, payment, and reporting processes.

4.2 Challenges and problems encountered

Handling and implementing file storage and retrieval efficiently.

Implementing validation for certain modules.

Managing concurrent access without database support.

4.3 Future work

Implement a GUI for enhanced user experience.

Integrate an online payment gateway.

Upgrade to a database management system for scalability.

5.0 References

W3Schools (n.d.). *Python Dates*. [online] www.w3schools.com. Available at: https://www.w3schools.com/python/python_datetime.asp

W3 Schools (2024). *Python Try Except*. [online] www.w3schools.com. Available at: https://www.w3schools.com/python/python_try_except.asp.

W3Schools (2019). *Python File Open*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_file_open.asp.

W3Schools (2019b). *Python File Write*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_file_write.asp.

w3schools (2024). *Python While Loops*. [online] www.w3schools.com. Available at: https://www.w3schools.com/python/python_while_loops.asp.

W3Schools (2019a). *Python Classes*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_classes.asp.

W3schools (2019). *Python Lists*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_lists.asp.

www.w3schools.com. (n.d.). *Python Arrays*. [online] Available at: https://www.w3schools.com/python/python_arrays.asp.

W3Schools (2019d). *Python Inheritance*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_inheritance.asp.

www.w3schools.com. (n.d.). *Python reversed() Function*. [online] Available at: https://www.w3schools.com/python/ref_func_reversed.asp.

6.0 Appendix

6.1 Code listing

```
import datetime

# Global lists to store cars, customers, and rentals
cars = []
customers = []
rentals = []

class Vehicle:
    # Class variable to count the number of vehicles
    vehicle_count = 0

    def __init__(self, model, year, color, daily_rate, is_rented=False):
        self._model = model
        self._year = year
        self._color = color
        self._daily_rate = daily_rate
        self._is_rented = is_rented
        Vehicle.vehicle_count += 1  # Increment vehicle count

    # Getter and Setter for model
    @property
    def model(self):
        return self._model

    @model.setter
    def model(self, model):
        self._model = model

    # Getter and Setter for year
    @property
    def year(self):
        return self._year

    @year.setter
    def year(self, year):
        current_year = datetime.datetime.now().year
        if year > current_year:
            print(f"Error: Year cannot be in the future. Current year is
{current_year}.")
            return  # Simply return without setting the value
        self._year = year

    # Getter and Setter for color
    @property
```

```
    def color(self):
        return self._color

    @color.setter
    def color(self, color):
        self._color = color

    # Getter and Setter for daily_rate
    @property
    def daily_rate(self):
        return self._daily_rate

    @daily_rate.setter
    def daily_rate(self, daily_rate):
        self._daily_rate = daily_rate

    # Getter and Setter for is_rented
    @property
    def is_rented(self):
        return self._is_rented

    @is_rented.setter
    def is_rented(self, is_rented):
        self._is_rented = is_rented

    def __str__(self):
        return f"{self._model} ({self._year}) - {self._color} -
Rs{self._daily_rate}/day - {'Rented' if self._is_rented else 'Available'}"


class Car(Vehicle):
    def __init__(self, car_id, model, year, color, daily_rate,
is_rented=False):
        super().__init__(model, year, color, daily_rate, is_rented)
        self._car_id = car_id

    # Getter and Setter for car_id
    @property
    def car_id(self):
        return self._car_id

    @car_id.setter
    def car_id(self, car_id):
        self._car_id = car_id

    def __str__(self):
        return f"ID: {self._car_id} - {self._model} ({self._year}) -
{self._color} - Rs{self._daily_rate}/day - {'Rented' if self._is_rented else
'Available'}"
```

```
    def add_car():
        print()
        while True:
            car_id = input("Enter car ID: ").strip()
            if len(car_id) == 4 and car_id[0] == "V" and car_id[1:].isdigit():
                if not any(c.car_id == car_id for c in cars):
                    break
                print("Error: Car ID already exists.")
            else:
                print("Error: Car ID must be in the format V followed by 3
digits (e.g., V001).")

        model = input("Enter the car model: ").strip()

        while True:
            try:
                year = int(input("Enter the car year: "))
                current_year = datetime.datetime.now().year
                if year <= current_year:
                    break
                print(f"Error: Year cannot be in the future. Current year is
{current_year}.")
            except ValueError:
                print("Error: Car year must be an integer.")

        while True:
            color = input("Enter the car color: ").strip()
            if color and not any(char.isdigit() for char in color):
                break
            print("Error: Car color cannot be empty or contain digits.")

        while True:
            try:
                daily_rate = float(input("Enter daily rental price: "))
                break
            except ValueError:
                print("Error: Daily rental price must be a valid number.")

        cars.append(Car(car_id, model, year, color, daily_rate))
        print("Car added successfully!")

    def remove_car():
        print()
        if not cars:
            print("The car list is empty.")
            return  # Exit the function early since there's nothing to remove
        Car.list_cars()
```

```
        print()

        # Loop until the car ID is valid
        while True:
            car_id = input("Enter the car ID of the car to remove : ").strip()

            # Validate car ID format
            if len(car_id) == 4 and car_id[0] == "V" and car_id[1:].isdigit():
                break  # Exit loop if input is valid
            else:
                print("Error: Car ID must be in the format V followed by 3
digits (e.g., V001). Please try again.")

        # Search for the car
        car = next((c for c in cars if c.car_id == car_id), None)
        if car:
            cars.remove(car)
            print(f"Car {car.model} removed successfully!")
        else:
            print("Car not found.")

    def list_cars():
        print()
        print("Available Cars:")
        print()
        for car in cars:
            print(car)


class Person:
    def __init__(self, name, number):
        self._name = name
        self._number = number

    # Getter and Setter for name
    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, name):
        self._name = name

    # Getter and Setter for number
    @property
    def number(self):
        return self._number
```

```
    @number.setter
    def number(self, number):
        self._number = number

    def __str__(self):
        return f"{self._name} - {self._number}"


class Customer(Person):
    def __init__(self, name, number, customer_id, rented_car=None):
        super().__init__(name, number)
        self._customer_id = customer_id
        self._rented_car = rented_car

    # Getter and Setter for customer_id
    @property
    def customer_id(self):
        return self._customer_id

    @customer_id.setter
    def customer_id(self, customer_id):
        self._customer_id = customer_id

    # Getter and Setter for rented_car
    @property
    def rented_car(self):
        return self._rented_car

    @rented_car.setter
    def rented_car(self, rented_car):
        self._rented_car = rented_car

    def __str__(self):
        return f"ID: {self._customer_id} - {self._name} - {self._number} -
{'Rented: ' + str(self._rented_car) if self._rented_car else 'No Car'}"

    def add_customer():
        print()
        while True:
            customer_id = input("Enter customer ID: ").strip()
            if len(customer_id) == 4 and customer_id[0] == "C" and
customer_id[1:].isdigit():
                if not any(c.customer_id == customer_id for c in customers):
                    break
                print("Error: Customer ID already exists.")
            else:
                print("Error: Customer ID must be in the format C followed by
3 digits (e.g., C001).")
```

```
        while True:
            name = input("Enter customer name: ").strip()
            if name and not any(char.isdigit() for char in name):
                break
            print("Error: Customer name cannot be empty or contain digits.")

        while True:
            number = input("Enter customer number: ").strip()
            if number.isdigit() and len(number) == 8:
                break
            print("Error: Customer number must be exactly 8 digits.")

        customers.append(Customer(name, number, customer_id))
        print("Customer added successfully!")

    def remove_customer():
        print()

        if not customers:
            print("The customer list is empty.")
            return  # Exit the function early since there's nothing to remove
        Customer.list_customers()
        print()

        # Loop until the customer ID is valid
        while True:
            customer_id = input("Enter the customer ID of the customer to
remove : ").strip()

            # Validate customer ID format
            if len(customer_id) == 4 and customer_id[0] == "C" and
customer_id[1:].isdigit():
                break  # Exit loop if input is valid
            else:
                print("Error: Customer ID must be in the format C followed by
3 digits (e.g., C001). Please try again.")

        # Search for the customer
        customer = next((c for c in customers if c.customer_id ==
customer_id), None)
        if customer:
            customers.remove(customer)
            print(f"Customer {customer.name} removed successfully!")
        else:
            print("Customer not found.")

    def list_customers():
        print()
```

```
        print("Customers:")
        for customer in customers:
            print(customer)



class Rental:
    def __init__(self, customer, car, rental_date, return_date, total_cost):
        self._customer = customer
        self._car = car
        self._rental_date = rental_date
        self._return_date = return_date
        self._total_cost = total_cost

    # Getter and Setter for customer
    @property
    def customer(self):
        return self._customer

    @customer.setter
    def customer(self, customer):
        self._customer = customer

    # Getter and Setter for car
    @property
    def car(self):
        return self._car

    @car.setter
    def car(self, car):
        self._car = car

    # Getter and Setter for rental_date
    @property
    def rental_date(self):
        return self._rental_date

    @rental_date.setter
    def rental_date(self, rental_date):
        self._rental_date = rental_date

    # Getter and Setter for return_date
    @property
    def return_date(self):
        return self._return_date

    @return_date.setter
    def return_date(self, return_date):
```

```
        self._return_date = return_date

    # Getter and Setter for total_cost
    @property
    def total_cost(self):
        return self._total_cost

    @total_cost.setter
    def total_cost(self, total_cost):
        self._total_cost = total_cost

    def __str__(self):
        return f"{self._customer.name} rented {self._car.model} on
{self._rental_date.strftime('%d/%m/%Y')} and returned on
{self._return_date.strftime('%d/%m/%Y')}. Total Cost: Rs{self._total_cost}"


    def rent_car():
        print()
        Car.list_cars()
        print("-----------------------------------------------------------------
-------")
        Customer.list_customers()
        print()

        while True:
            car_id = input("Enter the car ID of the car to rent : ").strip()
            if len(car_id) == 4 and car_id[0] == "V" and car_id[1:].isdigit():
                break
            print("Error: Car ID must be in the format V followed by 3 digits
(e.g., V001).")

        car = next((c for c in cars if c.car_id == car_id), None)
        if car:
            if not car.is_rented:
                while True:
                    customer_id = input("Enter your customer ID : ").strip()
                    if len(customer_id) == 4 and customer_id[0] == "C" and
customer_id[1:].isdigit():
                        break
                    print("Error: Customer ID must be in the format C followed
by 3 digits (e.g., C001).")

                customer = next((c for c in customers if c.customer_id ==
customer_id), None)
                if customer:
                    while True:
                        try:
```

```
                            days = int(input("Enter number of days to rent :
"))
                            if days > 0:
                                 break
                            print("Error: Number of days must be a positive
integer.")
                        except ValueError:
                            print("Error: Number of days must be an integer.")

                    total_cost = days * car.daily_rate
                    rental_date = datetime.date.today()
                    return_date = rental_date +
datetime.timedelta(days=days)  # Calculate final return date

                    car.is_rented = True
                    customer.rented_car = car

                    # Save to rental history file
                    with open("rental_history.txt", "a") as f:
                        f.write(f"{customer.name},{car.model},{car.year},{rent
al_date},{return_date},{days},Rs{total_cost}\n")

                    print()
                    print(f"{customer.name} has rented {car.model} for {days}
days.")
                    print(f"Total cost: Rs{total_cost}")
                    print(f"Rental Date: {rental_date.strftime('%d/%m/%Y')}")
                    print(f"Expected Return Date:
{return_date.strftime('%d/%m/%Y')}")
                else:
                    print("Customer not found. Please add the customer
first.")
            else:
                print("Car is already rented.")
        else:
            print("Car not found.")

    def return_car():
        print()

        found = False
        for c in customers:
            if c.rented_car != None:
                found = True
                break

        if not found:
            print("No cars are rented")
```

```
            return

        Customer.list_customers()

        # Loop until the customer ID is valid
        while True:
            customer_id = input("Enter your customer ID : ").strip()

            # Validate customer ID format
            if len(customer_id) == 4 and customer_id[0] == "C" and
customer_id[1:].isdigit():
                break  # Exit loop if input is valid
            else:
                print("Error: Customer ID must be in the format C followed by
3 digits (e.g., C001). Please try again.")

        # Search for the customer
        customer = next((c for c in customers if c.customer_id ==
customer_id), None)
        if customer and customer.rented_car:
            car_model = customer.rented_car

            car_year = 0
            for car in cars:
                if car.model == car_model:
                    car.is_rented = False
                    car_year = car.year
                    break

            return_date = datetime.date.today()
            customer.rented_car = None
            print(f"{customer.name} has returned {car.model}.")


            # Update the rental history with the return date
            with open("rental_history.txt", "r") as f:
                lines = f.readlines()
            with open("rental_history.txt", "w") as f:
                for line in lines:
                    if f"{customer.name},{car_model},{car_year}," in line:
                        f.write(line.replace(customer.name, "[Already
Returned] " + customer.name))
                    else:
                        f.write(line)
        else:
            print("No car to return or customer not found.")

    def view_rental_history():
        print()
```

```
        try:
            with open("rental_history.txt", "r") as f:
                print("Rental History:")
                for line in f:
                    print(line.strip())
        except FileNotFoundError:
            print("No rental history found.")

class CarRentalSystem:
    def load_data():
        global cars, customers
        try:
            with open("cars.txt", "r") as f:
                for line in f:
                    car_id, model, year, color, daily_rate, rented =
line.strip().split(",")
                    cars.append(Car(car_id, model, int(year), color,
float(daily_rate), rented == "True"))
        except FileNotFoundError:
            pass
            #print("Error: cars.txt file not found.")

        try:
            with open("customers.txt", "r") as f:
                for line in f:
                    name, number, customer_id = line.strip().split(",")
                    customers.append(Customer(name, number, customer_id))
        except FileNotFoundError:
            pass

        try:
            with open("rental_history.txt", "r") as f:
                for line in f:
                    customer_name = line.split(',')[0]
                    car_name = line.split(',')[1]

                    for customer in customers:
                        if customer.name == customer_name:
                            customer.rented_car = car_name
        except FileNotFoundError:
            pass

    def save_data():
        with open("cars.txt", "w") as f:
            for car in cars:
                f.write(f"{car.car_id},{car.model},{car.year},{car.color},{car
.daily_rate},{car.is_rented}\n")
```

```
        with open("customers.txt", "w") as f:
            for customer in customers:
                f.write(f"{customer.name},{customer.number},{customer.customer
_id}\n")

# Main function to run the program
def main():
    CarRentalSystem.load_data()
    while True:
        print("\n--------Car Rental System--------")
        print("1. Add Car")
        print("2. Remove Car")
        print("3. List Cars")
        print("4. Add Customer")
        print("5. Remove Customer")
        print("6. List Customers")
        print("7. Rent Car")
        print("8. Return Car")
        print("9. View Rental History")
        print("10. Exit")

        choice = input("Enter your choice : ")

        if choice == "1":
            Car.add_car()
        elif choice == "2":
            Car.remove_car()
        elif choice == "3":
            Car.list_cars()
        elif choice == "4":
            Customer.add_customer()
        elif choice == "5":
            Customer.remove_customer()
        elif choice == "6":
            Customer.list_customers()
        elif choice == "7":
            Rental.rent_car()
        elif choice == "8":
            Rental.return_car()
        elif choice == "9":
            Rental.view_rental_history()
        elif choice == "10":
            print("Exiting...")
            CarRentalSystem.save_data()
            break
        else:
            print("Invalid choice, please try again.")
```

```
main()
```

6.2 Sample Screenshots

Scenario 3 : Renting an already rented car

```
Available Cars:

ID: V001 - Honda Jazz (2020) - Grey - Rs1500.0/day - Rented
ID: V002 - Toyota Yaris (2019) - White - Rs1300.0/day - Rented
ID: V003 - Suzuki Swift (2023) - Black - Rs1800.0/day - Available
ID: V004 - BMW M5 (2025) - Red - Rs10000.0/day - Available
ID: V005 - Audi A5 (2015) - Blue - Rs3000.0/day - Available
----------------------------------------------------------------

Customers:
ID: C001 - John Doe - 12345678 - Rented: Honda Jazz
ID: C002 - Tom Smith - 23456789 - Rented: Toyota Yaris
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - No Car

Enter the car ID of the car to rent : V001
Car is already rented.
```

Scenario 4 : Removing cars and customers

Removing a car :

```
Available Cars:

ID: V001 - Honda Jazz (2020) - Grey - Rs1500.0/day - Rented
ID: V002 - Toyota Yaris (2019) - White - Rs1300.0/day - Rented
ID: V003 - Suzuki Swift (2023) - Black - Rs1800.0/day - Available
ID: V004 - BMW M5 (2025) - Red - Rs10000.0/day - Available
ID: V005 - Audi A5 (2015) - Blue - Rs3000.0/day - Available

Enter the car ID of the car to remove : V005
Car Audi A5 removed successfully!
```

Display updated car list:

```
Available Cars:

ID: V001 - Honda Jazz (2020) - Grey - Rs1500.0/day - Rented
ID: V002 - Toyota Yaris (2019) - White - Rs1300.0/day - Rented
ID: V003 - Suzuki Swift (2023) - Black - Rs1800.0/day - Available
ID: V004 - BMW M5 (2025) - Red - Rs10000.0/day - Available
```

Removing a customer :

```
Customers:
ID: C001 - John Doe - 12345678 - Rented: Honda Jazz
ID: C002 - Tom Smith - 23456789 - Rented: Toyota Yaris
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
ID: C005 - Khushi Bee - 56789123 - No Car

Enter the customer ID of the customer to remove : C005
Customer Khushi Bee removed successfully!
```

Display updated customer list:

```
Customers:
ID: C001 - John Doe - 12345678 - Rented: Honda Jazz
ID: C002 - Tom Smith - 23456789 - Rented: Toyota Yaris
ID: C003 - Jerry Johnson - 34567891 - No Car
ID: C004 - Isha Narain - 45678912 - No Car
```

6.3 Data Files

Note : Data should be input manually.

1. cars.txt

V001,Honda Jazz,2020,Grey,1500.0,False

V002,Toyota Yaris,2019,White,1300.0,False

V003,Suzuki Swift,2023,Black,1800.0,False

V004,BMW M5,2025,Red,10000.0,False

V005,Audi A5,2025,Red,8000.0,False

2. customers.txt

John Doe,12345678,C001

Tom Smith,23456789,C002

Jerry Johnson,34567891,C003

Isha Narain,45678912,C004

Khushi Bee,67891234,C005

3. rental_history.txt

[Already Returned] Joe,Honda,2020,2025-03-10,2025-03-12,2,Rs3000.0

[Already Returned] [Already Returned] John Doe,Honda Jazz,2020,2025-03-10,2025-03-12,2,Rs3000.0

[Already Returned] Khushi Bee,Audi A5,2015,2025-03-10,2025-03-14,4,Rs12000.0

[Already Returned] John Doe,Honda Jazz,2020,2025-03-11,2025-03-14,3,Rs4500.0

[Already Returned] Tom Smith,Toyota Yaris,2019,2025-03-11,2025-03-16,5,Rs6500.0