

# **ASANSOL ENGINEERING COLLEGE**



## **CAR RENTAL MANAGEMENT SYSTEM**

Report accomplished by --

**KARAN KUMAR NONIA**

Roll no.: 10800222101

Sub name: Object Oriented Programming

Sub code: PCC CS503

Year: 2024

**DEPARTMENT OF INFORMATION  
TECHNOLOGY**

## **Table of Content**

|                                  |    |
|----------------------------------|----|
| 1) Abstract .....                | 2  |
| 2) Introduction .....            | 3  |
| 3) Problem Definition .....      | 4  |
| 4) Methodology .....             | 5  |
| 5) Source Code .....             | 8  |
| 6) Outcomes .....                | 13 |
| 7) Conclusion & Future work..... | 16 |

## **Abstract**

Customer satisfaction is crucial for the success of any service-based industry. In the car rental business, traditional methods of managing car bookings and rental records often lead to inefficiencies and customer dissatisfaction. The Car Rental Management System reduces human errors by automating the process of managing cars, customers, and rental transactions. This system allows managers to monitor fleet availability, manage rental periods, calculate total charges, and track customer information seamlessly. In many rental services, cars are sometimes double-booked, leading to conflicts and dissatisfied customers. While this is a common issue, there is currently no system that significantly enhances communication between the rental service and customers. Hence, the goal of this system is to streamline the car rental process, reduce errors, and improve overall operational efficiency.

## Introduction

### **The Problem:**

In the car rental business, managing a fleet and handling customer bookings manually often leads to numerous issues. Rental agencies using traditional methods like paper-based logs or spreadsheets frequently face problems such as double-booking cars, mismanagement of customer records, incorrect rental durations, and calculation errors in billing. Such mistakes can result in poor customer experiences, reduced efficiency, and financial losses. A more automated system is needed to handle these tasks reliably and efficiently.

### **Project Objectives:**

The objective of the Car Rental Management System is to reduce human error and streamline the rental process, ensuring a smooth experience for both customers and staff. By automating core operations, the system minimizes the chances of double-booking, incorrect billing, and mismanagement of cars or customer information. With this system, errors are reduced to less than 0.1%, enhancing both operational efficiency and customer satisfaction.

### **Features:**

This system offers the following features:

- **Car availability:** Displays all available cars across economy, luxury, and SUV categories, ensuring that cars are not double-booked during the same time slots.
- **Customer details:** Keeps track of customer information such as name, phone number, and driver's license number, ensuring accurate and organized records.
- **Rental management:** Facilitates car rental by allowing customers to book vehicles for a specified duration (from 1 hour to 7 days), and tracks all active rentals along with return dates.
- **Rental charges:** Automatically calculates the total rental amount based on the car type (economy, luxury, SUV) and rental duration.
- **Most frequently rented car:** Displays the car model that has been rented the most frequently, helping the rental agency optimize their fleet.
- **Billing:** Generates detailed bills for each rental, including all necessary details such as car model, rental duration, and total charges.
- **Total rentals of the day:** Provides a summary of how many customers rented cars and how many rentals were processed on a given day.

## Problem Definition

Using the concept of Object-Oriented Programming (OOP), design a "Car Rental Management System." Suppose the car rental service has a fleet of 30 cars, categorized into economy, luxury, and SUV models. Each car can be rented for a minimum of 1 hour and a maximum of 7 days. You need to maintain the rental details, including the following: Car information (CarID, Model, Type, Availability), Customer details (CustomerID, Name, Phone, LicenseNumber), and Rental records (RentalID, CarID, CustomerID, RentalStart, RentalEnd, TotalAmount). Your system should be able to provide the following functionalities:

- **Show available cars for rent:** Display the list of cars available for rental in each category (economy, luxury, SUV).
- **Allow customers to rent cars for a specified duration:** Facilitate customer car rentals for any time between 1 hour and 7 days, based on the availability of the vehicle.
- **Calculate the total amount for each rental:** Automatically compute the total rental cost depending on the car type and rental duration.
- **Show a list of all current rentals and their return dates:** Display an overview of all cars that are currently rented, including details of when they are due for return.
- **Identify the most frequently rented car:** Provide insights into which car model is the most frequently rented by customers.
- **Generate rental bills:** Print detailed bills for each rental transaction, showing all necessary details including the customer information, car details, rental period, and the total amount charged.

## **Methodology**

The Car Rental Management System is implemented using the Java programming language. The object-oriented features of Java play a crucial role in building this system. This project is developed in two stages:

### **1. Designing :**

In this stage, I designed the system and identified the challenges that might arise during development, along with solutions to overcome them. The system design revolves around Object-Oriented Programming (OOP) principles such as encapsulation, inheritance, and polymorphism, which help organize the code and structure the functionality.

### **2. Coding: -**

The coding section is divided into three main parts: the CarRentalSystem class, the Main class, and the Car and Customer classes.

#### **CarRentalSystem class :**

This class manages the main features such as car availability, rental processing, billing, and tracking the most rented car. Static variables are used for car inventory, rental tracking, and calculating total rentals as they are shared across all customers and rentals.

#### ***CarRentalSystem :***

##### **Data Members:**

- static List<Car> cars: Stores the list of all cars in the system, marking them as available or rented.
- static List<Rental> rentals: Stores rental records, tracking car rentals, and return dates.
- static double totalRevenue: Keeps track of the total revenue collected from all rentals.
- static String mostRentedCar: Keeps track of the most frequently rented car model.
- int customerId: Stores the customer ID of each rental.
- String customerName: Stores the name of the customer.
- String customerPhone: Stores the phone number of the customer.
- double rentalAmount: Stores the total rental cost.
- int carId: Stores the car ID assigned to the customer.
- LocalDateTime rentalStart: Stores the start time of the rental period.
- LocalDateTime rentalEnd: Stores the end time of the rental period.

##### **Member Methods: -**

- `public boolean rentCar(int carId, String customerName, String customerPhone, LocalDateTime start, LocalDateTime end)`: This method checks if the car is available for the specified period and rents it if available, returning true; otherwise, returns false.
- `public void showAvailableCars()`: Displays the list of available cars for rent.
- `public void printRentalDetails()`: Prints the details of all active rentals, including car information, rental period, and customer details.
- `public double calculateTotalRevenue()`: Returns the total revenue generated from rentals.
- `public String getMostRentedCar()`: Returns the car model that has been rented the most times.
- `public void printBill(int rentalId)`: Prints the bill for the specified rental, showing details of the car, rental period, and the total rental amount.

## Car Class :

This class represents the car details such as CarID, model, type (economy, luxury, SUV), and availability status.

### **Data member: -**

- `int carId`: Unique ID for each car.
- `String model`: Car model name.
- `String type`: The category of the car (economy, luxury, SUV).
- `boolean isAvailable`: Indicates whether the car is available for rent.

### **Member Methods :**

- **`static void menuCard()`**: Print the menu card (dishes name along with their prices)
- **`Menu()`**: take the order and store the name of the dish in the `itemArray` and quantity in the `qtyArray`, and calculate the bill;
- **`double getBill()`**: Return the bill of the customer;
- **`int[] getItemArray()`**: Return the `itemArray` (orders dishes names)
- **`int[] getQtyArray()`**: Return the `qtyArray` (orders dishes quantity)

## Customer class:

This class stores customer details such as CustomerID, Name, Phone, and LicenseNumber.

### **Data Members:**

- `int customerId`: Unique ID for each customer.
- `String customerName`: Name of the customer.
- `String customerPhone`: Contact phone number.
- `String licenseNumber`: Driver's license number of the customer.

**Member Methods:**

- `public String getCustomerDetails():` Returns a formatted string with the customer's details.

**Rental class:**

This class handles the rental process by linking a customer with a car and maintaining the rental period.

**Data Members:**

- `int rentalId:` Unique ID for each rental transaction.
- `int carId:` ID of the rented car.
- `int customerId:` ID of the customer who rented the car.
- `LocalDateTime rentalStart:` Start date and time of the rental.
- `LocalDateTime rentalEnd:` End date and time of the rental.
- `double totalAmount:` Total amount charged for the rental.

**Member Methods:**

- `public double calculateAmount():` Calculates the total rental amount based on car type and rental duration.

**Main class:**

This class is responsible for executing the system and interacting with the user.

**Member Methods:**

- `public static void main(String[] args):` The main method that initiates the Car Rental Management System and handles user input, allowing users to rent cars, check availability, and view rental details.



## Source code

### Car Class :

```
public class Car {
    private int carId;
    private String model;
    private String type;
    private boolean isAvailable;

    public Car(int carId, String model, String type) {
        this.carId = carId;
        this.model = model;
        this.type = type;
        this.isAvailable = true;
    }

    public int getCarId() {
        return carId;
    }

    public String getModel() {
        return model;
    }

    public String getType() {
        return type;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void markAsRented() {
        this.isAvailable = false;
    }

    public void markAsAvailable() {
        this.isAvailable = true;
    }
}
```

### Rental Class :

```
import java.time.LocalDateTime;

public class Rental {
```

```

private static int rentalIdCounter = 1;
private int rentalId;
private int carId;
private int customerId;
private LocalDateTime rentalStart;
private LocalDateTime rentalEnd;
private double totalAmount;

public Rental(int carId, int customerId, LocalDateTime rentalStart, LocalDateTime
rentalEnd, double totalAmount) {
    this.rentalId = rentalIdCounter++;
    this.carId = carId;
    this.customerId = customerId;
    this.rentalStart = rentalStart;
    this.rentalEnd = rentalEnd;
    this.totalAmount = totalAmount;
}

public int getRentalId() {
    return rentalId;
}

public int getCarId() {
    return carId;
}

public int getCustomerId() {
    return customerId;
}

public LocalDateTime getRentalStart() {
    return rentalStart;
}

public LocalDateTime getRentalEnd() {
    return rentalEnd;
}

public double getTotalAmount() {
    return totalAmount;
}
}

```

## **CarRentalSystem Class :**

```

import java.time.Duration;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

```

```

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CarRentalSystem {
    private List<Car> cars = new ArrayList<>();
    private List<Rental> rentals = new ArrayList<>();
    private double totalRevenue = 0;
    private Map<String, Integer> carRentalCount = new HashMap<>();
    private int customerIdCounter = 100;

    private static final DateTimeFormatter DATE_FORMATTER =
    DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");

    public boolean rentCar(int carId, String customerName, String customerPhone,
    LocalDateTime start, LocalDateTime end) {
        for (Car car : cars) {
            if (car.getCarId() == carId && car.isAvailable()) {
                car.markAsRented();
                int customerId = customerIdCounter++;
                double rentalAmount = calculateRentalAmount(car, start, end);

                Rental rental = new Rental(carId, customerId, start, end, rentalAmount);
                rentals.add(rental);
                totalRevenue += rentalAmount;

                carRentalCount.put(car.getModel(),
                carRentalCount.getOrDefault(car.getModel(), 0) + 1);

                System.out.println("Car rented successfully!");
                System.out.printf("Customer ID: %d, Car ID: %d, Rental Amount: %.2f\n",
                customerId, carId, rentalAmount);
                return true;
            }
        }
        System.out.println("Car not available!");
        return false;
    }

    private double calculateRentalAmount(Car car, LocalDateTime start, LocalDateTime
    end) {
        long hours = Duration.between(start, end).toHours();
        double rate = switch (car.getType()) {
            case "Luxury" -> 100;
            case "SUV" -> 75;
            default -> 50;
        };
        return hours * rate;
    }

    public void showAvailableCars() {

```

```

        System.out.println("Available Cars:");
        for (Car car : cars) {
            if (car.isAvailable()) {
                System.out.printf("Car ID: %d, Model: %s, Type: %s\n", car.getCarId(),
car.getModel(), car.getType());
            }
        }
    }

    public void printRentalDetails() {
        System.out.println("Rental Details:");
        for (Rental rental : rentals) {
            System.out.printf("Rental ID: %d, Car ID: %d, Customer ID: %d, Total Amount:
%.2f\n",
                rental.getRentalId(), rental.getCarId(), rental.getCustomerId(),
rental.getTotalAmount());
        }
    }

    public double calculateTotalRevenue() {
        return totalRevenue;
    }

    public String getMostRentedCar() {
        return carRentalCount.entrySet().stream()
            .max(Map.Entry.comparingByValue())
            .map(Map.Entry::getKey)
            .orElse("No rentals yet");
    }

    public void printBill(int rentalId) {
        for (Rental rental : rentals) {
            if (rental.getRentalId() == rentalId) {
                System.out.printf("Customer ID: %d, Car ID: %d, Rental Period: %s to %s,
Total Amount: %.2f\n",
                    rental.getCustomerId(), rental.getCarId(),
                    rental.getRentalStart().format(DATE_FORMATTER),
                    rental.getRentalEnd().format(DATE_FORMATTER),
                    rental.getTotalAmount());
            }
        }
    }

    public List<Car> getCars() {
        return cars;
    }
}

```

## **KKN\_CRMS Class :**

This class contains the main function and its name is KKN\_CRMS

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class KKN_CRMS {
    private static final DateTimeFormatter DATE_FORMATTER =
        DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");

    public static void main(String[] args) {
        CarRentalSystem system = new CarRentalSystem();
        Scanner scanner = new Scanner(System.in);

        system.getCars().add(new Car(1, "Honda Civic", "Economy"));
        system.getCars().add(new Car(2, "BMW X5", "Luxury"));
        system.getCars().add(new Car(3, "Ford Explorer", "SUV"));

        boolean running = true;
        while (running) {
            System.out.println("1. Show available cars");
            System.out.println("2. Rent a car");
            System.out.println("3. Show rental details");
            System.out.println("4. Exit");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    system.showAvailableCars();
                    break;
                case 2:
                    System.out.print("Enter car ID: ");
                    int carId = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter customer name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter customer phone: ");
                    String phone = scanner.nextLine();
                    System.out.print("Enter rental start time (dd-MM-yyyy HH:mm): ");
                    LocalDateTime start = LocalDateTime.parse(scanner.nextLine(),
DATE_FORMATTER);
                    System.out.print("Enter rental end time (dd-MM-yyyy HH:mm): ");
                    LocalDateTime end = LocalDateTime.parse(scanner.nextLine(),
DATE_FORMATTER);
                    system.rentCar(carId, name, phone, start, end);
                    break;
                case 3:
                    system.printRentalDetails();
                    break;
                case 4:
```

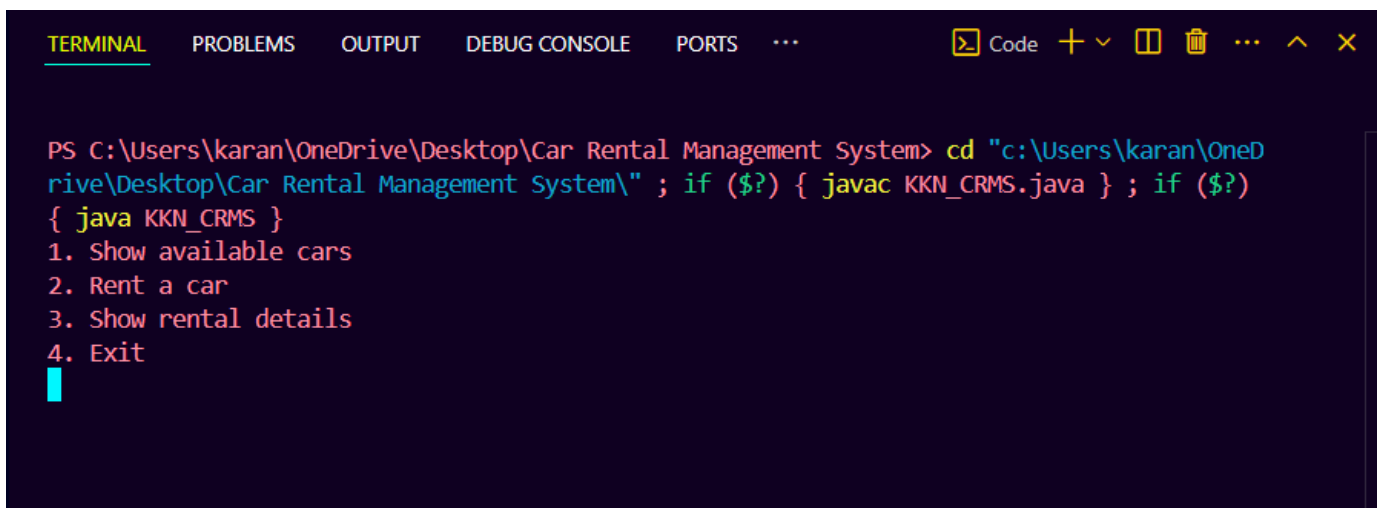
```

        running = false;
        break;
    default:
        System.out.println("Invalid option. Try again.");
    }
}
scanner.close();
}
}

```

## Outcomes

### 1. First view of the Terminal Window when the program is run



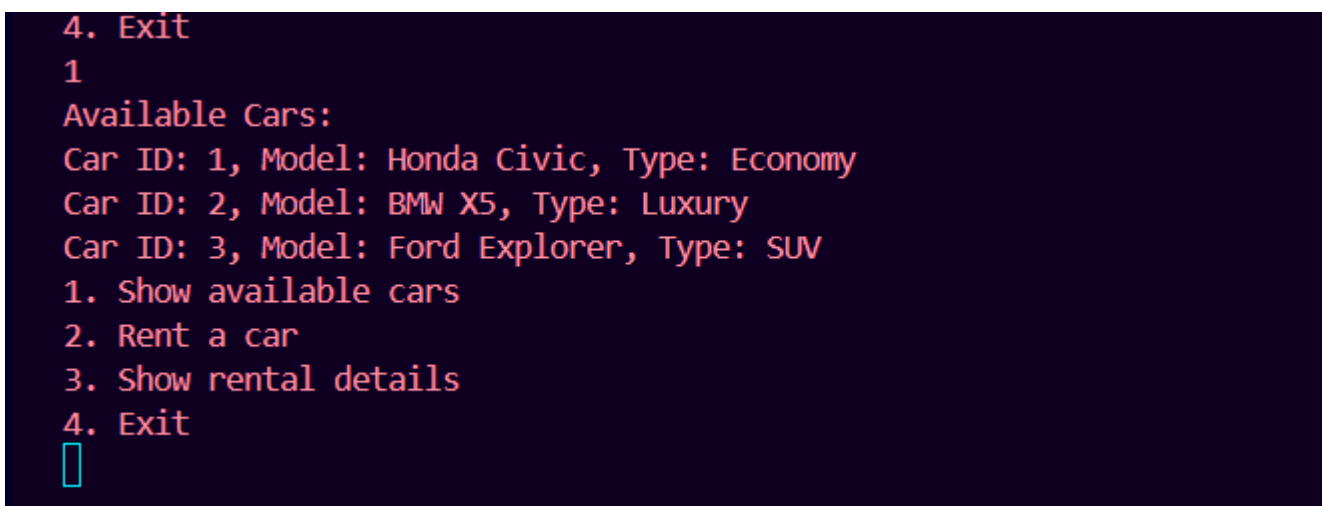
```

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  ...  Code  +  -  [ ]  [ ]  ...  ^  x

PS C:\Users\karan\OneDrive\Desktop\Car Rental Management System> cd "c:\Users\karan\OneDrive\Desktop\Car Rental Management System\" ; if ($?) { javac KKN_CRMS.java } ; if ($?) { java KKN_CRMS }
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit

```

### 2. when 1 is press then “all available cars are shown”



```

4. Exit
1
Available Cars:
Car ID: 1, Model: Honda Civic, Type: Economy
Car ID: 2, Model: BMW X5, Type: Luxury
Car ID: 3, Model: Ford Explorer, Type: SUV
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit

```

### 3. Now pressed 2 for Renting a car

Now we have to select the car ID from the available cars and then have to give the details like customer name, phone no., renting start and end date.

```
1
Available Cars:
Car ID: 1, Model: Honda Civic, Type: Economy
Car ID: 2, Model: BMW X5, Type: Luxury
Car ID: 3, Model: Ford Explorer, Type: SUV
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit
2
Enter car ID: 2
Enter customer name: Karan
Enter customer phone: 9304397220
Enter rental start time (dd-MM-yyyy HH:mm): 15-09-2024 15:15
Enter rental end time (dd-MM-yyyy HH:mm): 19-09-2024 12:12
Car rented successfully!
Customer ID: 100, Car ID: 2, Rental Amount: 9200.00
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit
```

4. Since the car with ID 1 is booked then it will be automatically be removed from the available cars list  
we can check this by choosing the option 1 to show all available cars .

```
1
Available Cars:
Car ID: 1, Model: Honda Civic, Type: Economy
Car ID: 3, Model: Ford Explorer, Type: SUV
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit
```

## 5. Now choose option 3 to show all Rental Details

```
3
Rental Details:
Rental ID: 1, Car ID: 2, Customer ID: 100, Total Amount: 9200.00
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit
█
```

## 6 To exit the program, press 4.

```
Rental ID: 1, Car ID: 2, Customer ID: 100, Total Amount: 9200.00
1. Show available cars
2. Rent a car
3. Show rental details
4. Exit
4
PS C:\Users\karan\OneDrive\Desktop\Car Rental Management System> █
```



## Conclusion & Future work

### **Conclusion:**

We successfully developed a **Car Rental Management System** that streamlines the rental process, from booking to billing. This system allows for efficient management of car rentals by storing customer details, tracking car availability, and managing rental transactions. It simplifies the car rental process by automating the management of rental records, calculating rental fees, and generating customer bills. The interconnected relationship between customers, cars, and rental transactions ensures accuracy and ease of use. This system significantly improves the efficiency and reliability of car rental operations, contributing to smoother business processes.

### **Future Work:**

1. **Improved User Interface:** We plan to design a more intuitive and visually appealing user interface to enhance user experience.
2. **Online Booking Feature:** A customer-facing version will be developed, allowing users to book cars online with ease.
3. **Database Integration:** Future iterations will include a robust database to store and manage all details more effectively.
4. **Mobile Application:** A mobile app version of the system will be developed to allow customers to rent cars on the go.
5. **Advanced Analytics:** Integrating advanced analytics to monitor usage patterns and optimize fleet management.

These planned improvements will make the system more scalable, user-friendly, and adaptable, ensuring the Car Rental Management System continues to meet the evolving needs of users and businesses.