CAR RENTAL SYSTEM

A PROJECT REPORT

Submitted by

KEERTHANA K(2303811710422076)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"CAR RENTAL SYSTEM"** is the bonafide work of **KEERTHANA K(2303811710422076)**who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 03/12/2024

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Ms.R.ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

[illegible]

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“CAR RENTAL SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

[illegible]

Name of the Student

Place: Samayapuram

Date:03/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The **Car Rental System** is a Java-based application designed to streamline the process of renting cars for customers while enabling car rental companies to manage their fleet and reservations efficiently. Built using **AWT (Abstract Window Toolkit)** for the graphical user interface, the system allows customers to search for available cars, book rentals, and view essential details such as rental prices and the total cost based on the number of days rented. It also includes an option to select the date of rent and payment method, offering various payment options such as **Credit Card**, **Debit Card**, **Cash**, and **UPI**. The rental price is dynamically calculated based on the car selected and the duration of the rental, and the system validates the input, ensuring the date format is correct. On the management side, the system helps rental companies keep track of available cars, handle bookings, and update the fleet status. By combining functionality for both customers and rental companies, the **Car Rental System** enhances the efficiency, transparency, and reliability of the car rental process.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Car Rental System** is a Java-based application utilizing AWT for a user-friendly interface, simplifying car rentals for customers and fleet management for rental companies. Customers can search, book cars, view rental prices, and choose payment methods like Credit Card, Debit Card, Cash, or UPI. The system dynamically calculates rental costs based on car selection and duration, with input validation for date formats. Rental companies can manage bookings, track availability, and update fleet status. It ensures efficiency, transparency, and reliability for both customers and businesses.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1Objective

The Car Rental System that allows users to easily search, view, and book cars through a graphical user interface (GUI). The system enables users to search for cars based on their preferences, display available cars along with their rental prices, and book a car by entering details like customer name, rental days, and payment mode. It also calculates the total cost based on the car’s daily rate and rental duration. The program ensures that user inputs are validated for accuracy and proper formatting, while also handling errors gracefully to prevent crashes. The interactive GUI, built using AWT components, aims to provide an intuitive and user-friendly experience. The core purpose of the system is to automate the car rental process, making it easier for users to rent vehicles while maintaining real-time updates on car availability and pricing, thus simplifying car rental operations for both users and businesses.

1.2Overview

The **Car Rental System** using Java's AWT for creating a graphical user interface (GUI). It allows users to search for available cars, view their details, and book them by providing necessary information such as customer name, rental days, and payment mode. The system dynamically updates the list of available cars after a booking, reflecting real-time availability. It validates user inputs like rental days and date formats, ensuring the data is correct before processing the booking. The total cost is calculated based on the selected car's price per day and the number of rental days. The program handles errors gracefully, providing feedback if the user inputs invalid data. Overall, the system provides a simple, interactive way to manage car rentals, making it easier for users to rent cars while maintaining up-to-date car availability.

1.3 Java Programming Concepts

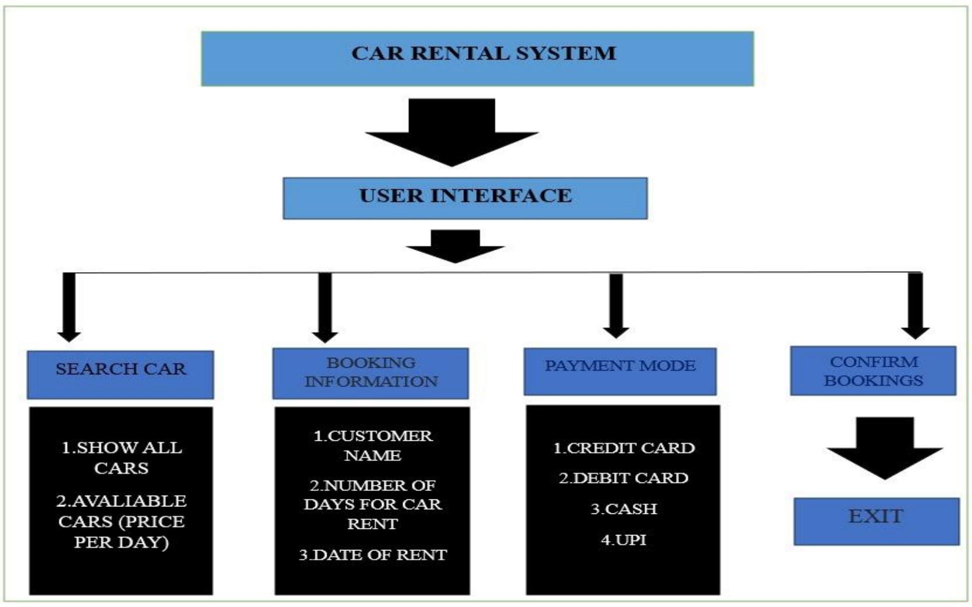
- **Class and Object:** The CarRentalSystem class represents the blueprint of the car rental system, and objects like Label, Button, and TextField are instances of GUI components.
- **Inheritance:** The CarRentalSystem class extends the Frame class, inheriting properties and methods for creating and managing a GUI window.
- **Encapsulation:** The program encapsulates data such as cars and booked Cars within the class, restricting direct access and using methods to manipulate them.
- **Polymorphism:** The action Performed method uses polymorphism to handle various events based on the source of the action (e.getSource()).
- **Event Handling:** The program uses the ActionListener interface to handle button click events like searching, booking, and displaying all cars. The Window Adapter class is used to handle the window-closing event, ensuring proper termination of the application.
- **AWT(Abstract Window Toolkit):**Components such as Label, Text Field, Button, Choice, and List are used to create an interactive interface. The program uses setLayout(null) for manual positioning of components. Custom fonts are applied using the Font class to enhance text appearance.
- **Exception Handling (try-catch):** Though this implementation does not explicitly show exceptions, Java exception handling could be incorporated, particularly for scenarios like invalid input or unsuccessful payment processing. The try-catch blocks would be used to handle exceptions such as InputMismatchException for invalid user input.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work aims to enhance the Car Rental System by adding features such as managing multiple car categories, tracking rental history, and integrating payment processing options. User authentication will be implemented for secure bookings, along with the ability to modify or cancel bookings. The system will also be integrated with a backend database for real-time inventory and customer data management. Additionally, the user interface will be improved for better responsiveness and clearer feedback. A reporting feature will be introduced to generate rental summaries, revenue reports, and customer details, making the system more comprehensive and efficient for both users and rental businesses.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1CarRentalGUI Module

Handles the graphical user interface (GUI) for the system. This module would include the setup and layout of the window, including buttons, labels, text fields, and other components for user interaction.

3.2CarInventory Module

Manages the list of cars available for rent. This module is responsible for storing and retrieving car details like name and price per day. HashMap to store the car data (key-value pairs of car names and prices).

3.3CarBooking Module

Handles car booking functionality. This includes validating user input, calculating rental costs, and updating the availability of cars once a booking is made. Input fields for customer name, rental days, payment mode, and date of rental. It also manages interactions with the HashMap that stores booked cars.

3.4CarSearch Module

Provides the search functionality for users to filter cars based on their search query. Text field for search input, and a mechanism to filter cars displayed in the GUI.

3.5PaymentProcessing Module

Manages payment details for bookings. While the code doesn’t actually process real payments, this module would handle the payment mode selection and validation. Choice for selecting payment options (Credit Card, Debit Card, Cash, UPI).

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

The Car Rental System provides an effective solution for managing car rental operations through an interactive and user-friendly interface. By utilizing Java's AWT for the graphical user interface (GUI) and handling key functionalities such as car search, booking, and payment processing, the system simplifies the rental process for both customers and businesses. The integration of data structures like HashMap ensures efficient management of car inventory and booking records. The system also incorporates essential features such as input validation for rental days and date formats, ensuring data integrity and preventing errors. While the current implementation offers basic functionality, the system can be further expanded with additional features such as payment gateway integration, user authentication, and reporting for better scalability and operational efficiency. Overall, the Car Rental System is a solid foundation for developing a comprehensive car rental solution that can be enhanced as needed to meet evolving business requirements.

4.2 FUTURE SCOPE

The future scope of the Car Rental System includes several enhancements that could improve its functionality, user experience, and scalability. Integrating real-time payment gateways like PayPal or UPI would streamline transactions and provide secure payment options. Adding user authentication would allow customers to create profiles, track bookings, and personalize their experience. Storing data in a database would facilitate better management of car inventory and bookings, enabling the system to scale and handle more users efficiently. Additionally, implementing advanced search filters, real-time car availability tracking, and mobile app integration would further enhance user convenience. Features like booking modification, cancellation, multi-language support, and an admin dashboard would improve both customer satisfaction and administrative control. Finally, expanding the system to include fleet management and analytics would provide car rental businesses with the tools to optimize operations. By incorporating these features, the Car Rental System can evolve into a robust, scalable solution that meets the growing needs of customers and businesses alike.

APPENDIX A

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Car {
    private String carId;
    private String brand;
    private String model;
    private double basePricePerDay;
    private boolean isAvailable;

    public Car(String carId, String brand, String model, double basePricePerDay) {
        this.carId = carId;
        this.brand = brand;
        this.model = model;
        this.basePricePerDay = basePricePerDay;
        this.isAvailable = true;
    }
    public String getCarId() {
        return carId;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double calculatePrice(int rentalDays) {
        return basePricePerDay * rentalDays;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void rent() {
        isAvailable = false;
    }

    public void returnCar() {
        isAvailable = true;
    }
}

class Customer {
    private String customerId;
    private String name;

    public Customer(String customerId, String name) {
        this.customerId = customerId;
        this.name = name;
    }
    public String getCustomerId() {
        return customerId;
    }

    public String getName() {
        return name;
    }
}

class Rental {
    private Car car;
    private Customer customer;
    private int days;

    public Rental(Car car, Customer customer, int days) {
        this.car = car;
        this.customer = customer;
        this.days = days;
    }

    public Car getCar() {
        return car;
    }

    public Customer getCustomer() {
        return customer;
    }

    public int getDays() {
        return days;
    }
}

class CarRentalSystem {
    private List<Car> cars;
    private List<Customer> customers;
    private List<Rental> rentals;

    public CarRentalSystem() {
        cars = new ArrayList<>();
        customers = new ArrayList<>();
        rentals = new ArrayList<>();
    }

    public void addCar(Car car) {
        cars.add(car);
    }

    public void addCustomer(Customer customer) {
        customers.add(customer);
    }

    public void rentCar(Car car, Customer customer, int days) {
        if (car.isAvailable()) {
            car.rent();
            rentals.add(new Rental(car, customer, days));

        } else {
```

```
        System.out.println("Oops! Car is not available for rent.");
    }
}

public void returnCar(Car car) {
    car.returnCar();
    Rental rentalToRemove = null;
    for (Rental rental : rentals) {
        if (rental.getCar() == car) {
            rentalToRemove = rental;
            break;
        }
    }
    if (rentalToRemove != null) {
        rentals.remove(rentalToRemove);

    } else {
        System.out.println("Car was not rented.");
    }
}

public void menu() {
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("===== Car Rental System =====");
        System.out.println("1. Rent a Car");
        System.out.println("2. Return a Car");
        System.out.println("3. Exit");
```

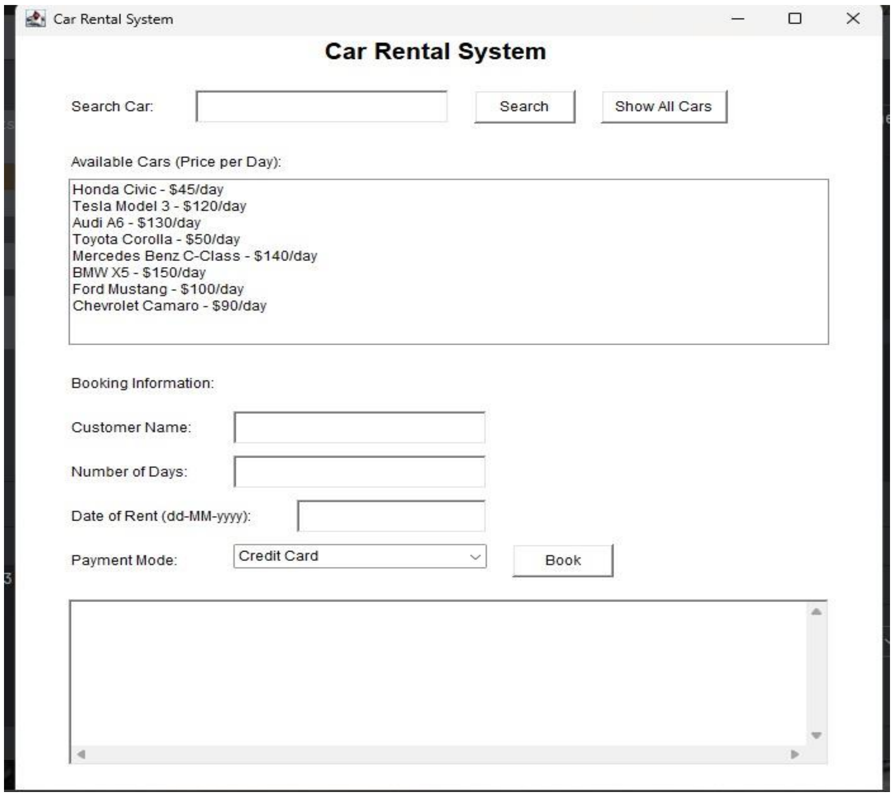
```
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            if (choice == 1) {
                System.out.println("\n== Rent a Car ==\n");
                System.out.print("Enter your name: ");
                String customerName = scanner.nextLine();

                System.out.println("\nAvailable Cars:");
                for (Car car : cars) {
                    if (car.isAvailable()) {
                        System.out.println(car.getCarId() + " - " + car.getBrand() + " " +
car.getModel());
                    }
                }

                System.out.print("\nEnter the car ID you want to rent: ");
                String carId = scanner.nextLine();

                System.out.print("Enter the number of days for rental: ");
                int rentalDays = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                Customer newCustomer = new Customer("CUS" + (customers.size() + 1),
customerName);
                addCustomer(newCustomer);
```

```
        Car selectedCar = null;
        for (Car car : cars) {
            if (car.getCarId().equals(carId) && car.isAvailable()) {
                selectedCar = car;
                break;
            }
        }

        if (selectedCar != null) {
            double totalPrice = selectedCar.calculatePrice(rentalDays);
            System.out.println("\n== Rental Information ==\n");
            System.out.println("Customer ID: " + newCustomer.getCustomerId());
            System.out.println("Customer Name: " + newCustomer.getName());
            System.out.println("Car: " + selectedCar.getBrand() + " " +
selectedCar.getModel());
            System.out.println("Rental Days: " + rentalDays);
            System.out.printf("Total Price: $%.2f%n", totalPrice);

            System.out.print("\nConfirm rental (Y/N): ");
            String confirm = scanner.nextLine();

            if (confirm.equalsIgnoreCase("Y")) {
                rentCar(selectedCar, newCustomer, rentalDays);
                System.out.println("\nCar rented successfully.");
            } else {
                System.out.println("\nRental canceled.");
            }
        } else {
            System.out.println("\nInvalid car selection or car not available for
rent.");
        }
    } else if (choice == 2) {
        System.out.println("\n== Return a Car ==\n");
        System.out.print("Enter the car ID you want to return: ");
        String carId = scanner.nextLine();

        Car carToReturn = null;
        for (Car car : cars) {
            if (car.getCarId().equals(carId) && !car.isAvailable()) {
                carToReturn = car;
                break;
            }
        }

        if (carToReturn != null) {
            Customer customer = null;
            for (Rental rental : rentals) {
                if (rental.getCar() == carToReturn) {
                    customer = rental.getCustomer();
                    break;
                }
            }

            if (customer != null) {
                returnCar(carToReturn);
                System.out.println("Car returned successfully by " +
customer.getName());
            } else {
```

```
                    System.out.println("Car was not rented or rental information is
missing.");
                }
            } else {
                System.out.println("Invalid car ID or car is not rented.");
            }
        } else if (choice == 3) {
            break;
        } else {
            System.out.println("Invalid choice. Please enter a valid option.");
        }
    }

    System.out.println("\nThank you for using the Car Rental System!");
  }

}
public class Main{
  public static void main(String[] args) {
    CarRentalSystem rentalSystem = new CarRentalSystem();

    Car car1 = new Car("C01", "Toyota", "Camry", 60.0); // Different base price per
day for each car
    Car car2 = new Car("C02", "Honda", "Accord", 70.0);
    Car car3 = new Car("C03", "Mahindra", "Thar", 150.0);
    Car car4 = new Car("C04", "Ford", "Mustang", 120.0);
    Car car5 = new Car("C05", "Chevrolet", "Cruze", 80.0);
    Car car6 = new Car("C06", "BMW", "X5", 200.0);
    Car car7 = new Car("C07", "Mercedes-Benz", "E-Class", 180.0);
```

```
        Car car8 = new Car("C08", "Audi", "Q7", 220.0);
        Car car9 = new Car("C09", "Nissan", "Altima", 75.0);
        Car car10 = new Car("C10", "Hyundai", "Tucson", 90.0);

        rentalSystem.addCar(car1);
        rentalSystem.addCar(car2);
        rentalSystem.addCar(car3);
        rentalSystem.addCar(car4);
        rentalSystem.addCar(car5);
        rentalSystem.addCar(car6);
        rentalSystem.addCar(car7);
        rentalSystem.addCar(car8);
        rentalSystem.addCar(car9);
        rentalSystem.addCar(car10);


        rentalSystem.menu();
    }
}
```

APPENDIX B



Car Rental System

Car Rental System

Search Car:

Available Cars (Price per Day):

```
Honda Civic - $45/day
Tesla Model 3 - $120/day
Audi A6 - $130/day
Toyota Corolla - $50/day
Mercedes Benz C-Class - $140/day
BMW X5 - $150/day
Ford Mustang - $100/day
Chevrolet Camaro - $90/day
```

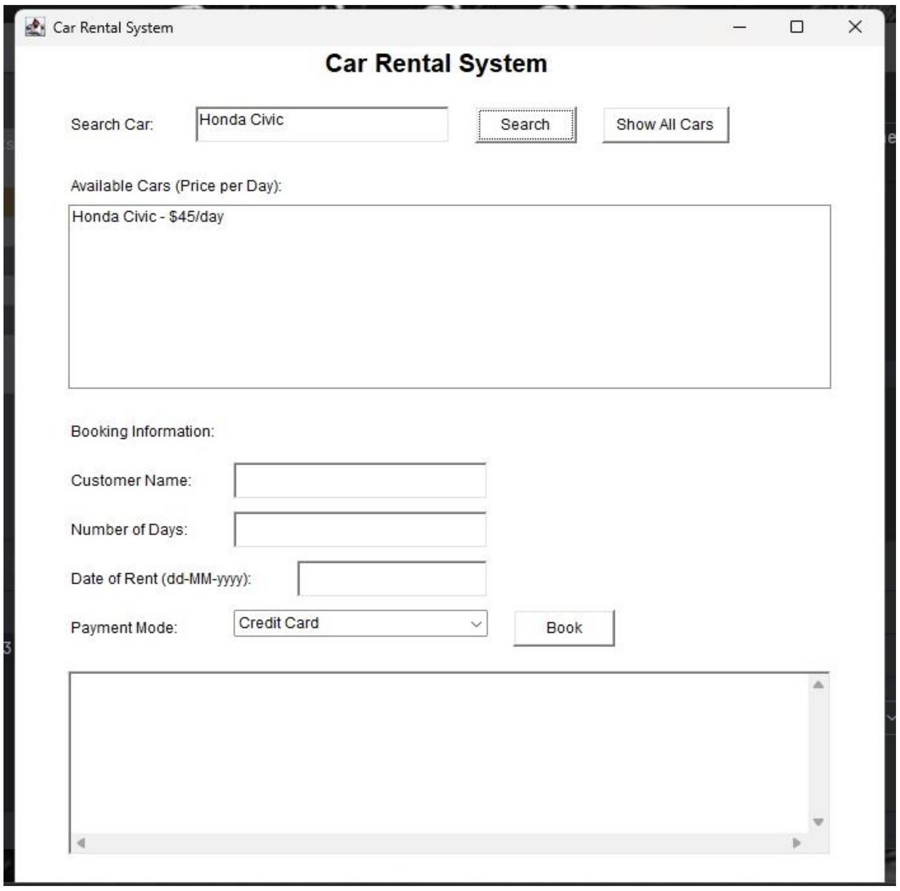
Booking Information:

Customer Name:

Number of Days:

Date of Rent (dd-MM-yyyy):

Payment Mode: Credit Card



Car Rental System

Car Rental System

Search Car: Honda Civic

Available Cars (Price per Day):

```
Honda Civic - $45/day
```

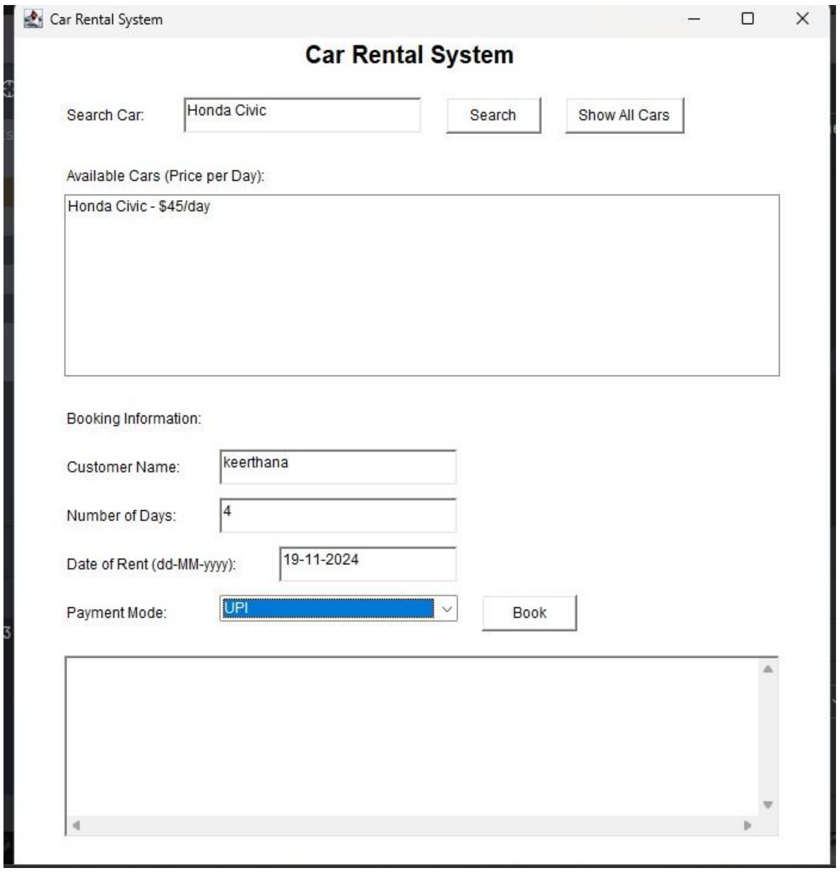
Booking Information:

Customer Name:

Number of Days:

Date of Rent (dd-MM-yyyy):

Payment Mode: Credit Card



Search Car: Honda Civic

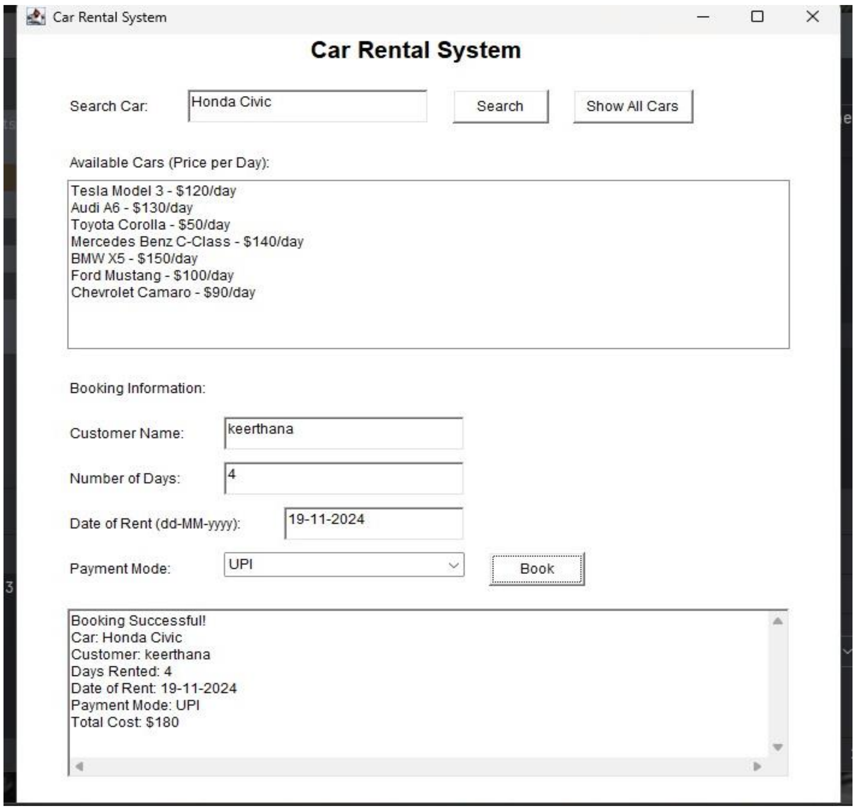
Honda Civic - $45/day

Customer Name: keerthana

Number of Days: 4

Date of Rent (dd-MM-yyyy): 19-11-2024

Payment Mode: UPI



Search Car: Honda Civic

```
Tesla Model 3 - $120/day
Audi A6 - $130/day
Toyota Corolla - $50/day
Mercedes Benz C-Class - $140/day
BMW X5 - $150/day
Ford Mustang - $100/day
Chevrolet Camaro - $90/day
```

Customer Name: keerthana

Number of Days: 4

Date of Rent (dd-MM-yyyy): 19-11-2024

Payment Mode: UPI

```
Booking Successful!
Car: Honda Civic
Customer: keerthana
Days Rented: 4
Date of Rent: 19-11-2024
Payment Mode: UPI
Total Cost: $180
```

REFERENCES

BOOKS

- "Building Java Programs: A Back to Basics Approach" by Stuart Reges and Marty Stepp
- "Car Rental System: A Java-Based Approach" by Muhammad Aslam (1st Edition, 2018)

WEBSITES

- **https://projectsgeek.com/2017/11/car-rental-system-project.html**
- **https://harshananayakkara.medium.com/car-rental-system-using-java-d5c6fd7e7f00**