A.D. PATEL INSTITUTE OF TECHNOLOGY

(A Constituent College of CVM University)

New V. V. Nagar

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATASCIENCE

Mini Project

Submitted By

Name of Student: Hani Patel(12202120601017)

Name of Student: Jahnvi Mistry(12202120601026)

Programming with Java (202044502)

Semester 5 A.Y. 2024-25

CODE:

```
import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import javax.swing.*;

public class CarRentalSystem extends JFrame {

    // Inner class to represent a Car
    static class Car {
        private String model;
        private double rentPerDay;
        private boolean isAvailable;
        private String type; // Sedan, SUV, Hatchback, etc.
        private String brand; // Toyota, BMW, Ford, etc.
        private String location; // City branches, airport locations

        public Car(String model, double rentPerDay, String type, String brand,
String location) {
            this.model = model;
            this.rentPerDay = rentPerDay;
            this.isAvailable = true;
            this.type = type;
            this.brand = brand;
            this.location = location;
        }

        public boolean isAvailable() {
            return isAvailable;
        }

        public void setAvailable(boolean available) {
            isAvailable = available;
        }

        public String getModel() {
            return model;
        }

        public double getRentPerDay() {
            return rentPerDay;
        }

        public String getType() {
            return type;
        }

        public String getBrand() {
            return brand;
        }

        public String getLocation() {
            return location;
        }
    }
```

```
    // Inner class to represent a Customer
    static class Customer {
        private String name;
        private String phoneNumber;
        private boolean isVIP; // Frequent renter status

        public Customer(String name, String phoneNumber, boolean isVIP) {
            this.name = name;
            this.phoneNumber = phoneNumber;
            this.isVIP = isVIP;
        }

        public String getName() {
            return name;
        }

        public String getPhoneNumber() {
            return phoneNumber;
        }

        public boolean isVIP() {
            return isVIP;
        }
    }

    // Inner class to represent a Reservation
    static class Reservation {
        private Car car;
        private Customer customer;
        private String reservationDate;
        private String returnDate;
        private String status; // Upcoming, Past, Pending, Cancelled
        private double totalAmount;

        public Reservation(Car car, Customer customer, String reservationDate,
String returnDate, String status) {
            this.car = car;
            this.customer = customer;
            this.reservationDate = reservationDate;
            this.returnDate = returnDate;
            this.status = status;
            this.totalAmount = calculateTotalAmount();
        }

        private double calculateTotalAmount() {
            // For simplicity, assume 1 day rental
            return car.getRentPerDay();
        }

        public Car getCar() {
            return car;
        }

        public Customer getCustomer() {
            return customer;
        }
```

```
        public String getStatus() {
            return status;
        }

        public double getTotalAmount() {
            return totalAmount;
        }
    }

    // Billing class to manage payments
    static class Billing {
        private List<Reservation> payments;

        public Billing() {
            payments = new ArrayList<>();
        }

        public void addPayment(Reservation reservation) {
            payments.add(reservation);
        }

        public List<Reservation> getPayments() {
            return payments;
        }

        public double getTotalRevenue() {
            double total = 0;
            for (Reservation payment : payments) {
                total += payment.getTotalAmount();
            }
            return total;
        }

        public List<Reservation> getUnpaidInvoices() {
            // For simplicity, we'll treat all reservations as unpaid for now
            return payments;
        }
    }

    // Rental Service class
    static class RentalService {
        private List<Car> cars;
        private List<Customer> customers;
        private List<Reservation> reservations;
        private Billing billing;

        public RentalService() {
            cars = new ArrayList<>();
            customers = new ArrayList<>();
            reservations = new ArrayList<>();
            billing = new Billing();

            // Sample data
            cars.add(new Car("Toyota Camry", 50.0, "Sedan", "Toyota", "City
Branch"));
```

```
           cars.add(new Car("Honda Accord", 45.0, "Sedan", "Honda", "City
Branch"));
            cars.add(new Car("Ford Mustang", 80.0, "SUV", "Ford", "Airport"));
            cars.add(new Car("Tesla Model 3", 100.0, "Electric", "Tesla", "City
Branch"));

            customers.add(new Customer("John Doe", "1234567890", true));
            customers.add(new Customer("Jane Smith", "0987654321", false));
        }

        public int getTotalCars() {
            return cars.size();
        }

        public List<Car> getAvailableCars() {
            List<Car> availableCars = new ArrayList<>();
            for (Car car : cars) {
                if (car.isAvailable()) {
                    availableCars.add(car);
                }
            }
            return availableCars;
        }

        public List<Car> getRentedCars() {
            List<Car> rentedCars = new ArrayList<>();
            for (Car car : cars) {
                if (!car.isAvailable()) {
                    rentedCars.add(car);
                }
            }
            return rentedCars;
        }

        public List<Reservation> getPendingReservations() {
            List<Reservation> pendingReservations = new ArrayList<>();
            for (Reservation reservation : reservations) {
                if (reservation.getStatus().equals("Upcoming")) {
                    pendingReservations.add(reservation);
                }
            }
            return pendingReservations;
        }

        public void addReservation(Reservation reservation) {
            reservations.add(reservation);
            billing.addPayment(reservation); // Add to billing as well
        }

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

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

```
        public List<Customer> getAllCustomers() {
            return customers;
        }

        public boolean makeReservation(String model, String customerName, String
phoneNumber, String reservationDate, String returnDate) {
            for (Car car : cars) {
                if (car.getModel().equalsIgnoreCase(model) && car.isAvailable())
{
                    car.setAvailable(false);
                    Customer customer = new Customer(customerName, phoneNumber,
false);
                    Reservation reservation = new Reservation(car, customer,
reservationDate, returnDate, "Upcoming");
                    addReservation(reservation);
                    return true;
                }
            }
            return false; // Reservation failed if car not available
        }
    }

    private RentalService rentalService;
    private JPanel mainPanel;

    public CarRentalSystem() {
        rentalService = new RentalService();
        initializeDashboard();
    }

    private void initializeDashboard() {
        setTitle("Car Rental Management System");
        setSize(1000, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        mainPanel = new JPanel(new BorderLayout());
        add(createSideMenu(), BorderLayout.WEST); // Sidebar with menu
        add(mainPanel, BorderLayout.CENTER); // Main area for content

        showDashboard(); // Show dashboard on startup

        setVisible(true);
    }

    private JPanel createSideMenu() {
        JPanel panel = new JPanel(new GridLayout(12, 1));
        panel.setPreferredSize(new Dimension(200, getHeight()));

        JButton dashboardBtn = new JButton("Dashboard");
        JButton fleetManagementBtn = new JButton("Fleet Management");
        JButton customerManagementBtn = new JButton("Customer Management");
        JButton reservationsBtn = new JButton("Reservations");
        JButton billingPaymentsBtn = new JButton("Billing & Payments");
        JButton reportsBtn = new JButton("Reports");
        JButton supportBtn = new JButton("Support");
```

```
        JButton addCustomerBtn = new JButton("Add Customer"); // Button to add a
customer
        JButton makeReservationBtn = new JButton("Make Reservation"); // Button
to make a reservation

        // Add action listeners to buttons
        dashboardBtn.addActionListener(e -> showDashboard());
        fleetManagementBtn.addActionListener(e -> showFleetManagement());
        customerManagementBtn.addActionListener(e -> showCustomerManagement());
        reservationsBtn.addActionListener(e -> showReservations());
        billingPaymentsBtn.addActionListener(e -> showBillingPayments());
        reportsBtn.addActionListener(e -> showReports());
        supportBtn.addActionListener(e -> showSupport());
        addCustomerBtn.addActionListener(e -> openAddCustomerDialog()); // Add
action for adding customer
        makeReservationBtn.addActionListener(e -> openReservationDialog()); //
Add action for making reservation

        // Add buttons to panel
        panel.add(dashboardBtn);
        panel.add(fleetManagementBtn);
        panel.add(customerManagementBtn);
        panel.add(addCustomerBtn); // Add button to panel
        panel.add(makeReservationBtn); // Add reservation button
        panel.add(reservationsBtn);
        panel.add(billingPaymentsBtn);
        panel.add(reportsBtn);
        panel.add(supportBtn);

        return panel;
    }

    private void showDashboard() {
        mainPanel.removeAll();
        String[] columnNames = {"Category", "Value"};
        Object[][] data = {
                {"Total Cars", rentalService.getTotalCars()},
                {"Available Cars", rentalService.getAvailableCars().size()},
                {"Rented Cars", rentalService.getRentedCars().size()},
                {"Total Reservations",
rentalService.getPendingReservations().size()},
        };

        JTable table = new JTable(data, columnNames);
        mainPanel.add(new JScrollPane(table), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void showFleetManagement() {
        mainPanel.removeAll();
        String[] columnNames = {"Model", "Rent Per Day", "Type", "Brand",
"Location"};
        List<Car> cars = rentalService.getAllCars();
        Object[][] data = new Object[cars.size()][5];
```

```
        for (int i = 0; i < cars.size(); i++) {
            Car car = cars.get(i);
            data[i][0] = car.getModel();
            data[i][1] = car.getRentPerDay();
            data[i][2] = car.getType();
            data[i][3] = car.getBrand();
            data[i][4] = car.getLocation();
        }

        JTable table = new JTable(data, columnNames);
        mainPanel.add(new JScrollPane(table), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void showCustomerManagement() {
        mainPanel.removeAll();
        String[] columnNames = {"Name", "Phone Number", "VIP Status"};
        List<Customer> customers = rentalService.getAllCustomers();
        Object[][] data = new Object[customers.size()][3];

        for (int i = 0; i < customers.size(); i++) {
            Customer customer = customers.get(i);
            data[i][0] = customer.getName();
            data[i][1] = customer.getPhoneNumber();
            data[i][2] = customer.isVIP() ? "Yes" : "No";
        }

        JTable table = new JTable(data, columnNames);
        mainPanel.add(new JScrollPane(table), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void openAddCustomerDialog() {
        JDialog dialog = new JDialog(this, "Add New Customer", true);
        dialog.setLayout(new GridLayout(4, 2));
        dialog.setSize(300, 200);

        JTextField nameField = new JTextField();
        JTextField phoneField = new JTextField();
        JCheckBox vipCheckBox = new JCheckBox("VIP Customer");

        dialog.add(new JLabel("Name:"));
        dialog.add(nameField);
        dialog.add(new JLabel("Phone Number:"));
        dialog.add(phoneField);
        dialog.add(new JLabel("VIP Customer:"));
        dialog.add(vipCheckBox);

        JButton addButton = new JButton("Add Customer");
        addButton.addActionListener(e -> {
            String name = nameField.getText();
            String phone = phoneField.getText();
            boolean isVIP = vipCheckBox.isSelected();

            if (!name.isEmpty() && !phone.isEmpty()) {
```

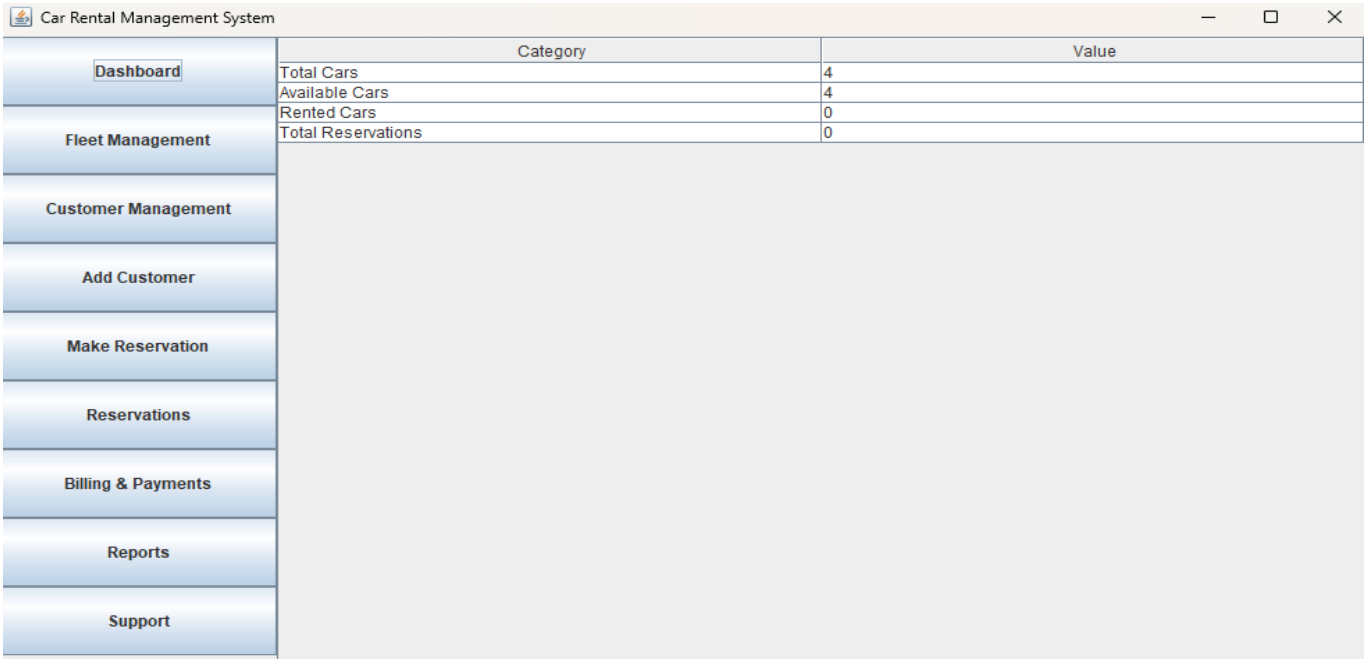
```
                rentalService.addCustomer(new Customer(name, phone, isVIP));
                JOptionPane.showMessageDialog(dialog, "Customer added
successfully!");
                dialog.dispose();
                showCustomerManagement(); // Refresh the customer management
view
            } else {
                JOptionPane.showMessageDialog(dialog, "Please fill all fields.",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        dialog.add(addButton);
        dialog.setVisible(true);
    }

    private void openReservationDialog() {
        JDialog dialog = new JDialog(this, "Make a Reservation", true);
        dialog.setLayout(new GridLayout(5, 2));
        dialog.setSize(400, 300);

        JTextField carModelField = new JTextField();
        JTextField customerNameField = new JTextField();
        JTextField phoneField = new JTextField();
        JTextField reservationDateField = new JTextField(); // Format: YYYY-MM-
DD
        JTextField returnDateField = new JTextField(); // Format: YYYY-MM-DD

        dialog.add(new JLabel("Car Model:"));
        dialog.add(carModelField);
        dialog.add(new JLabel("Customer Name:"));
        dialog.add(customerNameField);
        dialog.add(new JLabel("Phone Number:"));
        dialog.add(phoneField);
        dialog.add(new JLabel("Reservation Date (YYYY-MM-DD):"));
        dialog.add(reservationDateField);
        dialog.add(new JLabel("Return Date (YYYY-MM-DD):"));
        dialog.add(returnDateField);

        JButton reserveButton = new JButton("Reserve Car");
        reserveButton.addActionListener(e -> {
            String model = carModelField.getText();
            String customerName = customerNameField.getText();
            String phoneNumber = phoneField.getText();
            String reservationDate = reservationDateField.getText();
            String returnDate = returnDateField.getText();

            if (rentalService.makeReservation(model, customerName, phoneNumber,
reservationDate, returnDate)) {
                JOptionPane.showMessageDialog(dialog, "Reservation made
successfully!");
                dialog.dispose();
                showReservations(); // Refresh reservations view
            } else {
                JOptionPane.showMessageDialog(dialog, "Car is not available for
reservation.", "Error", JOptionPane.ERROR_MESSAGE);
```

```
 }
        });

        dialog.add(reserveButton);
        dialog.setVisible(true);
    }

    private void showReservations() {
        mainPanel.removeAll();
        String[] columnNames = {"Car Model", "Customer Name", "Phone Number",
"Reservation Date", "Return Date", "Status", "Total Amount"};
        List<Reservation> reservations = rentalService.getPendingReservations();
        Object[][] data = new Object[reservations.size()][7];

        for (int i = 0; i < reservations.size(); i++) {
            Reservation reservation = reservations.get(i);
            data[i][0] = reservation.getCar().getModel();
            data[i][1] = reservation.getCustomer().getName();
            data[i][2] = reservation.getCustomer().getPhoneNumber();
            data[i][3] = reservation.reservationDate;
            data[i][4] = reservation.returnDate;
            data[i][5] = reservation.getStatus();
            data[i][6] = reservation.getTotalAmount(); // Show total amount for
reservation
        }

        JTable table = new JTable(data, columnNames);
        mainPanel.add(new JScrollPane(table), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void showBillingPayments() {
        mainPanel.removeAll();
        String[] columnNames = {"Car Model", "Customer Name", "Total Amount"};
        List<Reservation> payments = rentalService.billing.getPayments();
        Object[][] data = new Object[payments.size()][3];

        for (int i = 0; i < payments.size(); i++) {
            Reservation payment = payments.get(i);
            data[i][0] = payment.getCar().getModel();
            data[i][1] = payment.getCustomer().getName();
            data[i][2] = payment.getTotalAmount(); // Show total amount for
payment
        }

        JTable table = new JTable(data, columnNames);
        mainPanel.add(new JScrollPane(table), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void showReports() {
        mainPanel.removeAll();
        String totalRevenue = "Total Revenue: $" +
rentalService.billing.getTotalRevenue();
```

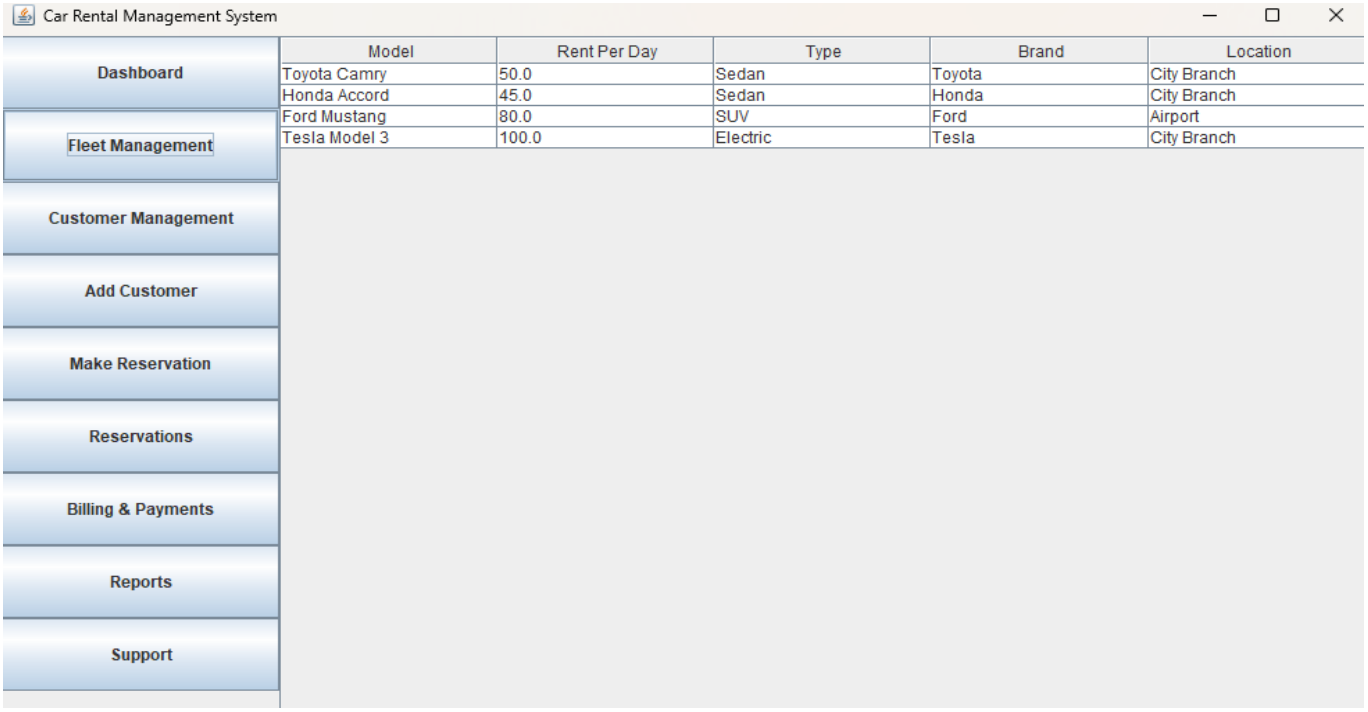
```
String totalPendingReservations = "Total Pending Reservations: " +
rentalService.getPendingReservations().size();

        JTextArea reportArea = new JTextArea(totalRevenue + "\n" +
totalPendingReservations);
        reportArea.setEditable(false);

        mainPanel.add(new JScrollPane(reportArea), BorderLayout.CENTER);
        mainPanel.revalidate();
        mainPanel.repaint();
    }

    private void showSupport() {
        // Placeholder function for support
        JOptionPane.showMessageDialog(this, "Support section is not implemented
yet.");
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(CarRentalSystem::new);
    }
}
```

Output

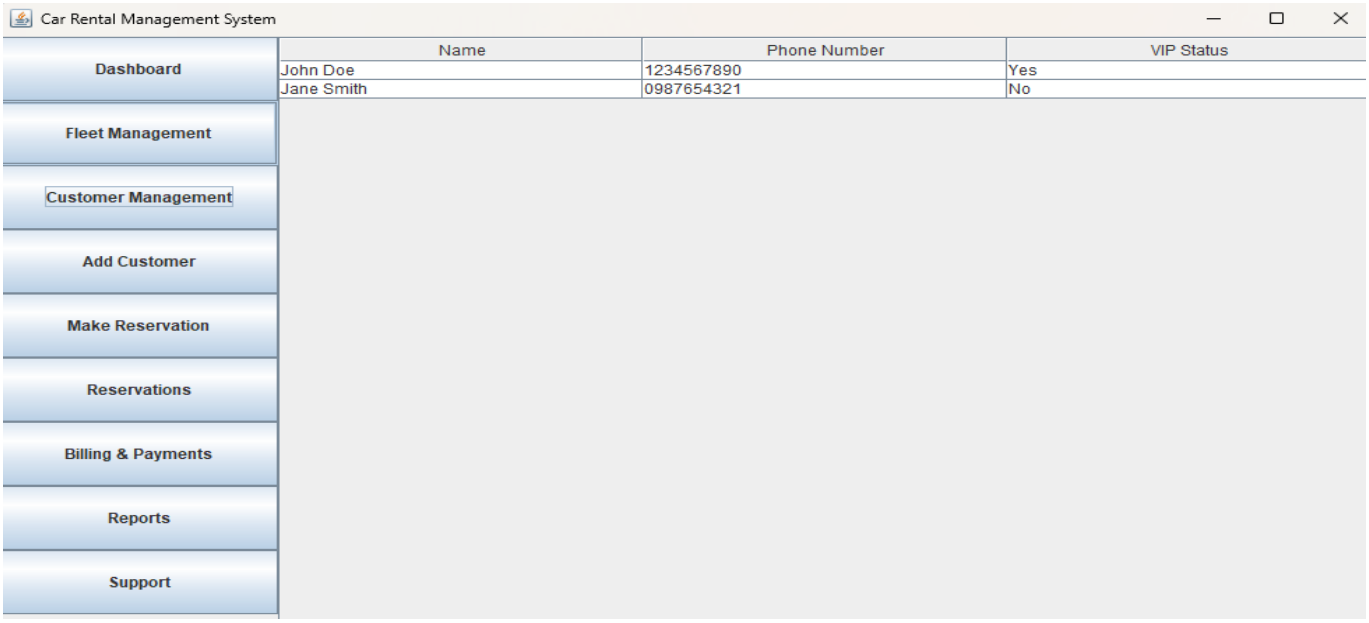
Dashboard



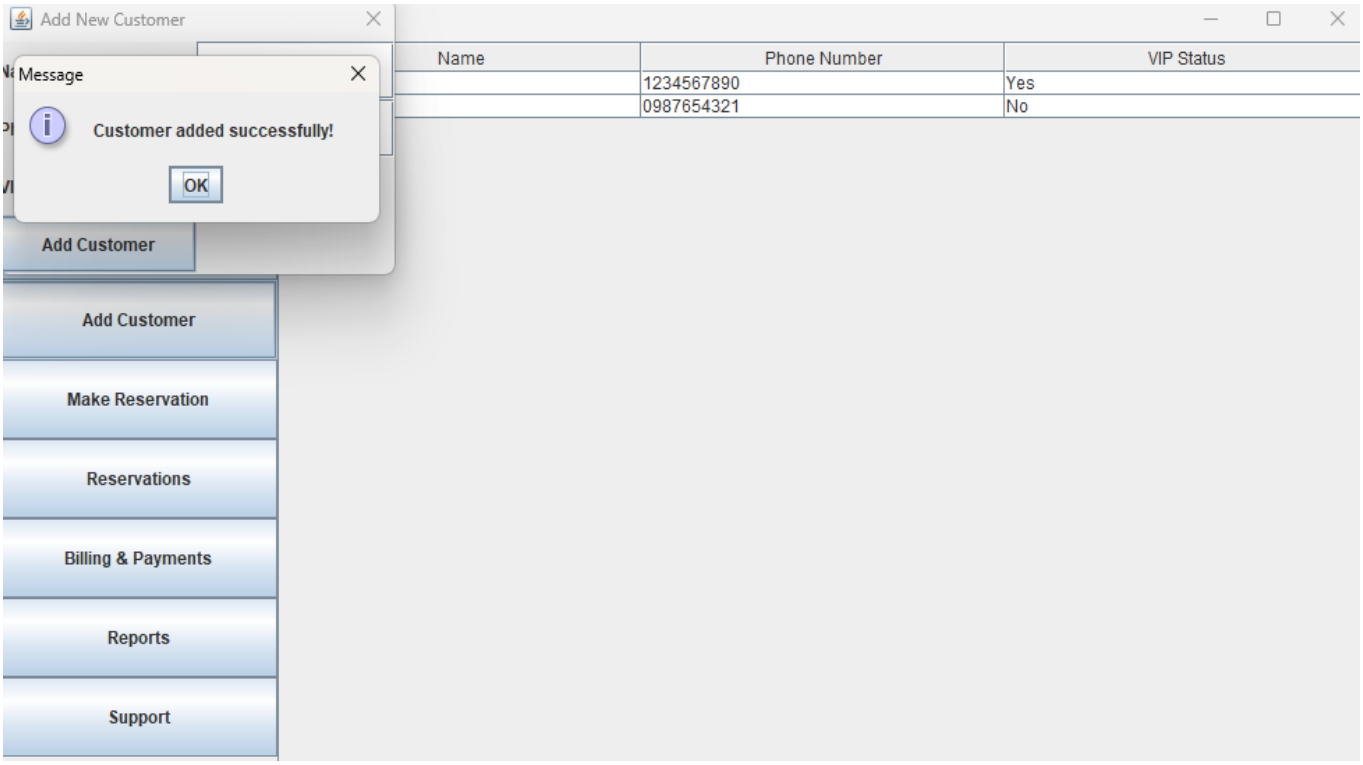
Fleet Management



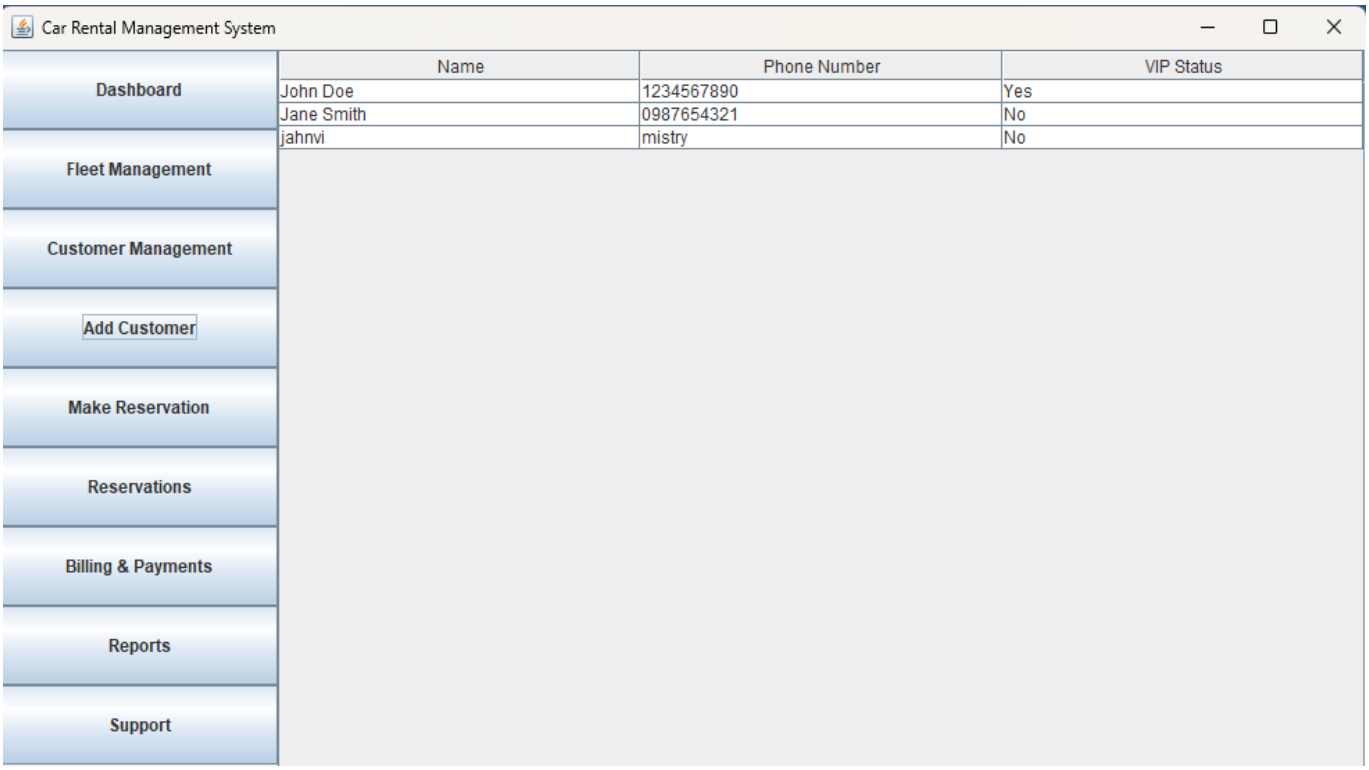
Before adding customer



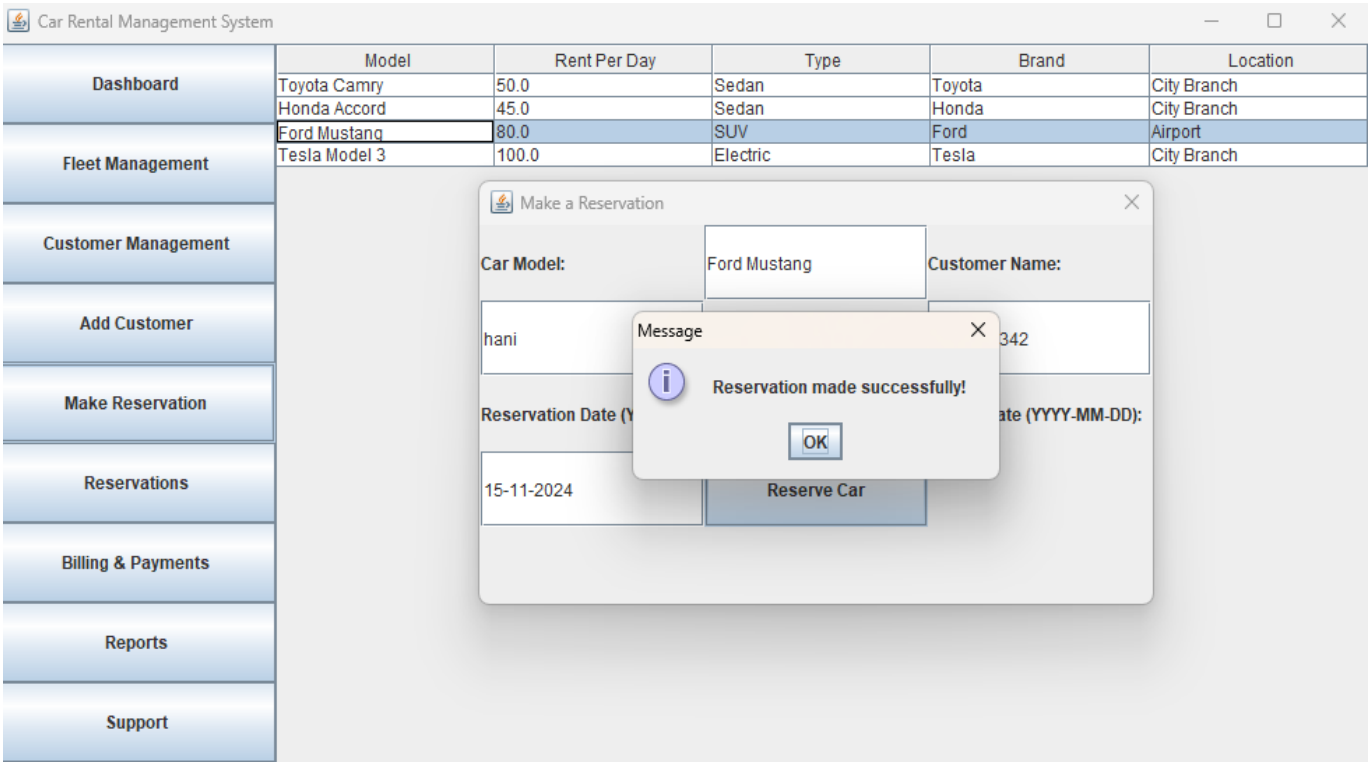
Adding customer



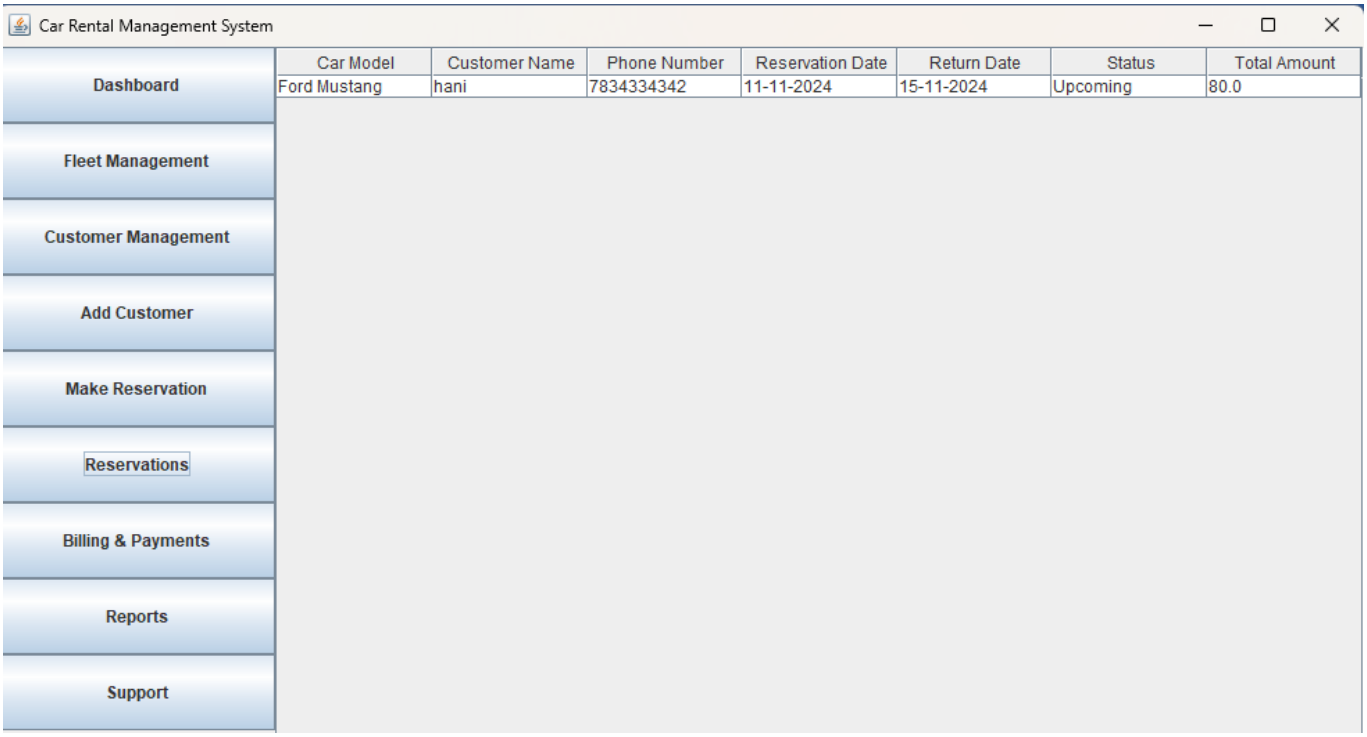
After adding



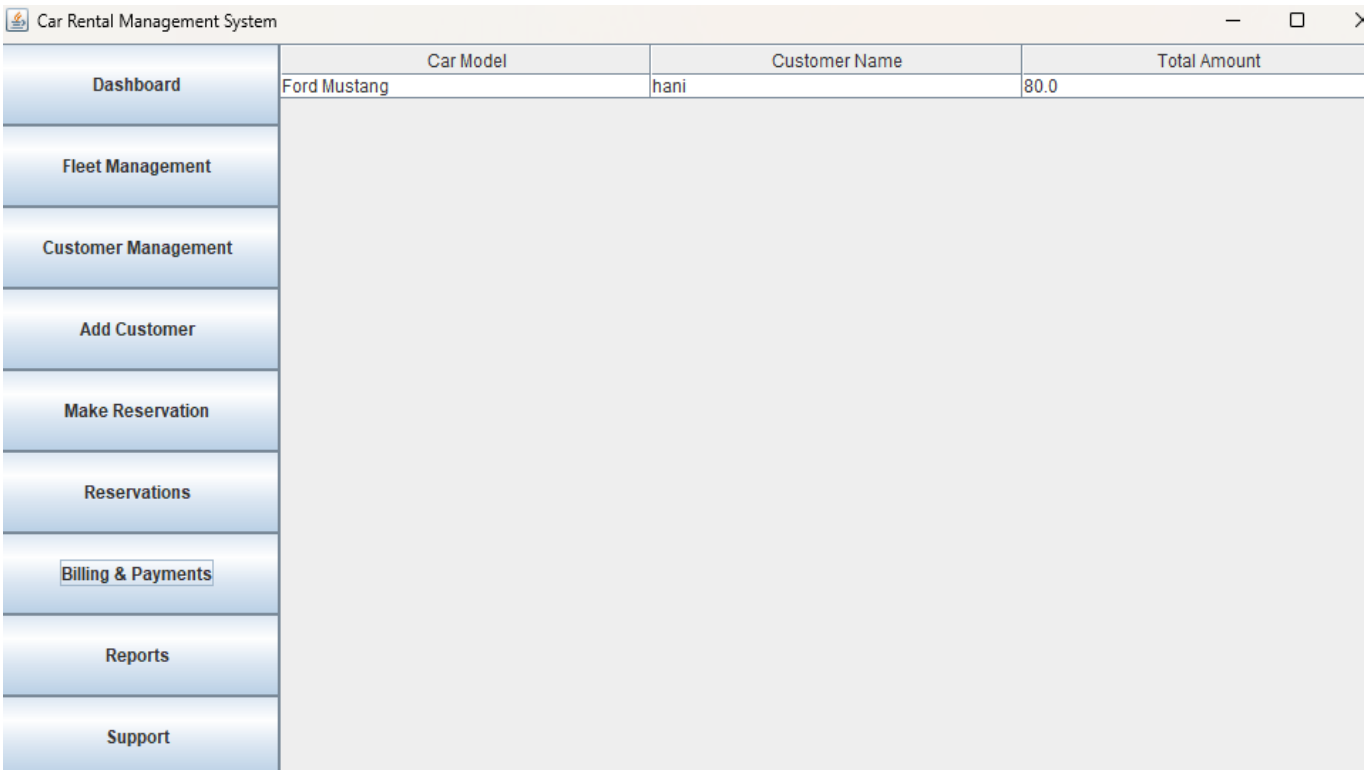
Making Reservation



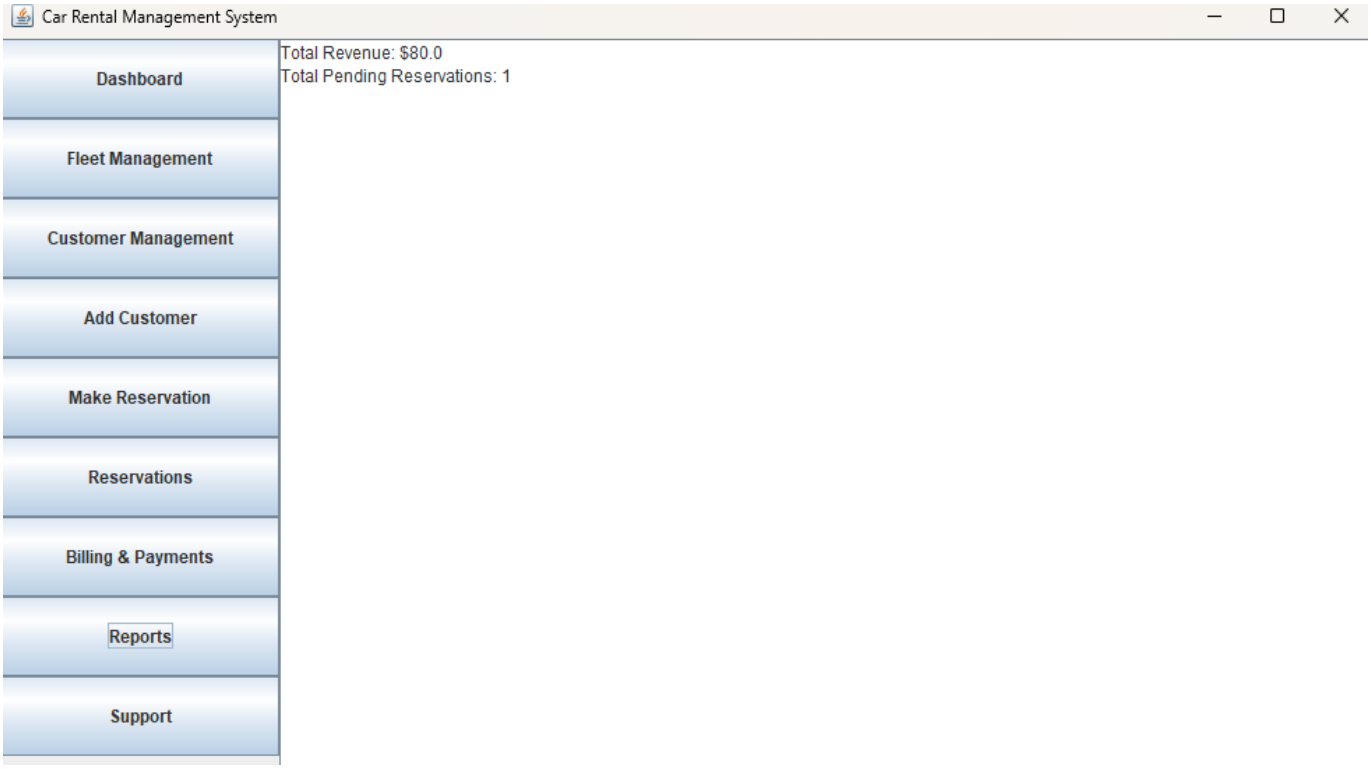
Reservations



Billing and Payment



Reports


Car Rental Management System
Dashboard
Fleet Management
Customer Management
Add Customer
Make Reservation
Reservations
Billing & Payments
Reports
Support
Total Revenue: $80.0
Total Pending Reservations: 1