

**NATIONAL UNIVERSITY OF MODEREN LANGUAGES**

**PROJECT REPORT ON**

**“CAR PARKING SYSTEM”**

**SUBJECT:**

OBJECT ORIENTED PROGRAMMING

**SUBMITTED TO:**

MR. USMAN SHAHID

**SUBMITTED BY:**

SHEERAZ ALI (FL23724)

MUSA MUMTAZ (FL23756)

SADIA(FL23745)

QASWAR SARFRAZ (FL23734)

HOORIA SAJID (FL23752)

**INDEX:**

**Contents:**

* PROBLEM ANALYZIATION
* PROBLEM REQUIREMENT
* DESIGN(Algorithm)
* IMPLEMENTATION
* FEATURES(Output) (testing and verifying)
* CONCLUSION
* SUGGESTION

DECLARATION:

The Car Parking System GUI is a Java Swing-based application designed to manage vehicle registrations, book parking slots, and handle customer registration and login. The application includes several functionalities encapsulated in various classes and provides a graphical user interface to interact with these features.

**PROBLEM ANALYZATION:**

The Car Parking System GUI addresses the need for an automated solution to manage vehicle parking in a parking facility. The core aspects analyzed for the problem include:

* **Vehicle Management**: There needs to be a system to register and deregister vehicles. This includes storing vehicle details and allowing the user to view all registered vehicles.
* **Customer Management**: The system must manage customer registration and login functionalities, ensuring that only authenticated users can access certain features.
* **Parking Slot Booking**: A mechanism to book parking slots for registered vehicles and to calculate the cost based on the booking duration.
* **Payment Management**: The system should generate a bill based on the hours a vehicle is parked and maintain a record of the total earnings.
* **User Interface**: A user-friendly graphical interface to interact with all the functionalities mentioned above.

**PROBLEM REQUIREMENTS:**

To fulfill the problem analysis, the following requirements have been identified:

**User Registration**:

* Allow new users to register by providing details such as name, email, password, phone number, and gender.
* Store user credentials securely.

**User Login**:

* Authenticate users based on their email and password.
* Provide feedback on unsuccessful login attempts.

**Vehicle Registration**:

* Register a vehicle with details such as model name, year, company, and number plate.
* Limit the number of vehicles a user can register to 5.

**Vehicle De-registration**:

* Allow users to remove a vehicle from the system using the number plate.

**Display Registered Vehicles**:

* Provide a list of all registered vehicles.

**Parking Slot Booking**:

* Allow users to book a parking slot for a registered vehicle.
* Specify the duration of the parking.

**Payment Calculation**:

* Calculate the total parking fee based on a predefined hourly rate.
* Update the total earnings with each transaction.

**Reporting**:

* Generate reports on total earnings, vehicle usage, and registered vehicles.
* Non-Functional Requirements

**Usability**:

* The system should have an intuitive and easy-to-navigate GUI.
* Provide clear messages for successful and unsuccessful operations.

**Performance**:

* The system should handle operations quickly and efficiently, with minimal delay.

**Reliability**:

* Ensure data consistency, particularly for vehicle registration and user authentication.

**Scalability**:

* Design the system in a way that allows for potential future enhancements, such as increasing the limit of registered vehicles or integrating more detailed reporting features.

**Security**:

* Protect user credentials and sensitive information.
* Implement measures to prevent unauthorized access.

**Maintainability**:

* Write clean and modular code to facilitate easy maintenance and future updates.

**Classes and Their Functions**

1. **Vehicle Class**

The Vehicle class models a vehicle with the following attributes:

**Key Attributes:**

* modelName: The model of the vehicle.
* year: The manufacturing year of the vehicle.
* company: The company that manufactured the vehicle.
* numberPlate: The unique identifier (number plate) of the vehicle.

**Key Methods:**

* Constructor to initialize the attributes.
* Getter methods for each attribute.
* toString method to provide a string representation of the vehicle.

1. **VehicleRegistration Class**

The VehicleRegistration class manages the collection of vehicles. It supports adding and removing vehicles and displaying the list of registered vehicles.

**Key Attributes:**

* v: Array to store up to 5 vehicles.
* count: Counter to keep track of the number of registered vehicles.

**Key Methods:**

* addVehicle: Adds a new vehicle if the limit (5) is not exceeded.
* removeVehicle: Removes a vehicle based on its number plate.
* getVehicles: Returns an array of currently registered vehicles.
* showVehicles: Returns a string listing all registered vehicles.

1. **BookSlot Class**

The BookSlot class manages the booking of parking slots.

**Key Attribute:**

* hours: The number of hours for which a slot is booked.

**Key Method:**

* bookSlot: Sets the number of hours for the booking and prints a confirmation message.

1. **Payment Class**

The Payment class calculates the bill for parking based on the number of hours.

**Key Attribute:**

* pricePerHour: Rate per hour for parking.

**Key Method:**

* generateBill: Calculates and prints the total bill based on the number of hours.

1. **Register Class**

The Register class handles customer registration.

**Key Attributes:**

* name, email, password, phoneNumber, gender: Customer details.

**Key Methods:**

* registerCustomer: Registers a customer with the provided details and prints a confirmation message.
* getemail, getpassword: Getter methods for email and password.

1. **Login Class**

The Login class manages customer login.

**Key Attributes:**

* email, password: Credentials for login.

**Key Methods:**

* login: Checks if the provided credentials match the registered user.
* forgetPassword: Sends a password reset link if the email is registered.

1. **CarParkingSystemGUI Class**

The CarParkingSystemGUI class is the main GUI class extending JFrame. It integrates all functionalities and provides the user interface.

**Key Attributes:**

* registeredUser: Stores the registered user details.
* isLogin: Boolean to track if a user is logged in.
* vehicleRegistration: Instance of VehicleRegistration.
* totalEarnings: Accumulates total earnings from parking fees.

**Key Methods:**

* initComponents: Initializes the GUI components and layouts.
* Event listeners for handling user actions such as login, registration, vehicle management, slot booking, and report generation.

**GUI Components and Panels**

The GUI is composed of several panels, each serving different purposes:

* Login Panel: For user login.
* Register Panel: For new user registration.
* Main Panel: Contains buttons to navigate to different functionalities like vehicle registration, reports, etc.
* Vehicle Panel: For adding, removing, and displaying vehicles.
* Report Panel: For generating and displaying various reports.

**Event Handling**

Event listeners are attached to various buttons to handle actions like:

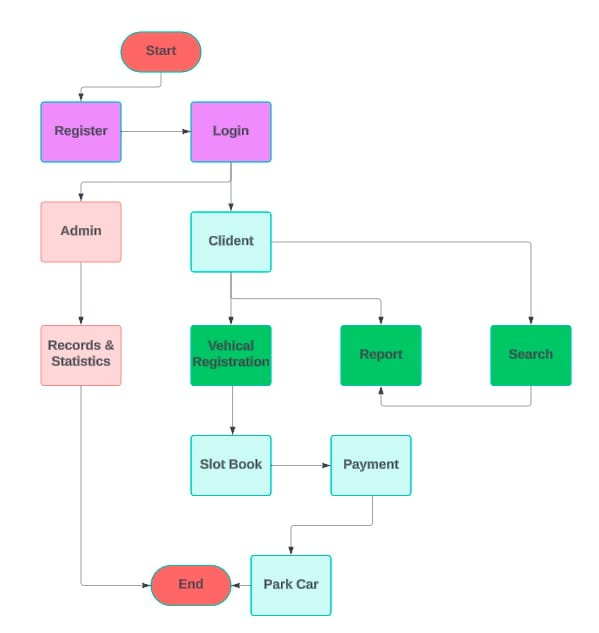
* Login and registration.
* Adding and removing vehicles.
* Booking parking slots.
* Displaying reports.

**Execution Flow**

* Starting the Application: The main method creates an instance of CarParkingSystemGUI.
* User Login: The user logs in using the login panel.
* Navigation: Based on user actions, different panels are shown using CardLayout.
* Vehicle Management: Users can add or remove vehicles and view the list of registered vehicles.
* Slot Booking and Payment: Users can book parking slots for registered vehicles and generate bills.
* Reports: Users can view reports on total earnings and registered vehicles.

**DESIGN**

**(Flowchart):**

****

IMPLEMENTATION **(CODE):**

package gui;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

class Vehicle {

    private String modelName;

    private int year;

    private String company;

    private String numberPlate;

    public Vehicle(String modelName, int year, String company, String numberPlate) {

        this.modelName = modelName;

        this.year = year;

        this.company = company;

        this.numberPlate = numberPlate;

    }

    public String getModelName() {

        return modelName;

    }

    public int getYear() {

        return year;

    }

    public String getCompany() {

        return company;

    }

    public String getNumberPlate() {

        return numberPlate;

    }

    @Override

    public String toString() {

        return "Model Name: " + modelName + ", Year: " + year + ", Company: " + company + ", Number Plate: " + numberPlate;

    }

}

class VehicleRegistration {

    private Vehicle[] v;

    private int count;

    public VehicleRegistration() {

        v = new Vehicle[5];

        count = 0;

    }

    public void addVehicle(String modelName, int year, String company, String numberPlate) {

        if (count < 5) {

            v[count] = new Vehicle(modelName, year, company, numberPlate);

            count++;

            System.out.println("Vehicle added successfully.");

        } else {

            System.out.println("Cannot add more vehicles. The limit is 5.");

        }

    }

    public void removeVehicle(String numberPlate) {

        boolean found = false;

        for (int i = 0; i < count; i++) {

            if (v[i].getNumberPlate().equals(numberPlate)) {

                v[i] = v[count - 1]; // Move the last vehicle to the current position

                v[count - 1] = null; // Nullify the last position

                count--;

                found = true;

                System.out.println("Vehicle removed successfully.");

                break;

            }

        }

        if (!found) {

            System.out.println("Vehicle not found.");

        }

    }

    public Vehicle[] getVehicles() {

        Vehicle[] registeredVehicles = new Vehicle[count];

        System.arraycopy(v, 0, registeredVehicles, 0, count);

        return registeredVehicles;

    }

    public String showVehicles() {

        if (count == 0) {

            return "No vehicles to display.";

        } else {

            StringBuilder sb = new StringBuilder();

            for (int i = 0; i < count; i++) {

                sb.append(v[i].toString()).append("\n");

            }

            return sb.toString();

        }

    }

}

class BookSlot {

    private int hours;

    public BookSlot() {

        hours = 0;

    }

    public void bookSlot(int hours) {

        this.hours = hours;

        System.out.println("Slot booked for " + hours + " hours.");

    }

}

class Payment {

    private double pricePerHour;

    public Payment(double pricePerHour) {

        this.pricePerHour = pricePerHour;

    }

    public double generateBill(int hours) {

        double totalAmount = pricePerHour \* hours;

        System.out.println("Bill generated: " + totalAmount + " for " + hours + " hours.");

        return totalAmount;

    }

}

class Register {

    private String name;

    private String email;

    private String password;

    private String phoneNumber;

    private String gender;

    public Register() {}

    public void registerCustomer(String name, String email, String password, String phoneNumber, String gender) {

[this.name](http://this.name/) = name;

        this.email = email;

        this.password = password;

        this.phoneNumber = phoneNumber;

        this.gender = gender;

        System.out.println("Customer registered with the email: " + email);

    }

    public String getemail() {

        return email;

    }

    public String getpassword() {

        return password;

    }

}

class Login {

    private String email;

    private String password;

    public Login(String email, String password) {

        this.email = email;

        this.password = password;

    }

    public boolean login(Register registeredUser) {

        System.out.println("Attempting to login with email: " + email);

        return registeredUser != null && registeredUser.getemail().equals(email) && registeredUser.getpassword().equals(password);

    }

    public void forgetPassword(Register registeredUser) {

        if (registeredUser != null && registeredUser.getemail().equals(email)) {

            if (registeredUser.getpassword().equals(password)) {

                System.out.println("Password reset link sent to email: " + email);

            } else {

                System.out.println("Incorrect password or email");

            }

        } else {

            System.out.println("Email not found in the system.");

        }

    }

}

public class CarParkingSystemGUI extends JFrame {

    private Register registeredUser;

    private boolean isLogin;

    private VehicleRegistration vehicleRegistration;

    private double totalEarnings;

    public CarParkingSystemGUI() {

        registeredUser = null;

        isLogin = false;

        vehicleRegistration = new VehicleRegistration();

        totalEarnings = 0.0;

        initComponents();

    }

    private void initComponents() {

        setTitle("5 Idiots Car Parking System");

        setSize(600, 400);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new CardLayout());

        JPanel loginPanel = new JPanel(new GridLayout(5, 2));

        JPanel registerPanel = new JPanel(new GridLayout(7, 2));

        JPanel mainPanel = new JPanel(new GridLayout(5, 1));

        JPanel vehiclePanel = new JPanel(new GridLayout(7, 2));

        JPanel reportPanel = new JPanel(new GridLayout(5, 1));

        CardLayout cardLayout = (CardLayout) getContentPane().getLayout();

        // Login Panel

        JTextField loginEmailField = new JTextField();

        JPasswordField loginPasswordField = new JPasswordField();

        JButton loginButton = new JButton("Login");

        JButton registerButton = new JButton("Register");

        loginPanel.add(new JLabel("Email:"));

        loginPanel.add(loginEmailField);

        loginPanel.add(new JLabel("Password:"));

        loginPanel.add(loginPasswordField);

        loginPanel.add(loginButton);

        loginPanel.add(registerButton);

        // Register Panel

        JTextField regNameField = new JTextField();

        JTextField regEmailField = new JTextField();

        JPasswordField regPasswordField = new JPasswordField();

        JTextField regPhoneField = new JTextField();

        JRadioButton maleRadioButton = new JRadioButton("Male");

        JRadioButton femaleRadioButton = new JRadioButton("Female");

        ButtonGroup genderGroup = new ButtonGroup();

        genderGroup.add(maleRadioButton);

        genderGroup.add(femaleRadioButton);

        JPanel genderPanel = new JPanel(new GridLayout(1, 2));

        genderPanel.add(maleRadioButton);

        genderPanel.add(femaleRadioButton);

        JButton submitRegisterButton = new JButton("Submit");

        registerPanel.add(new JLabel("Name:"));

        registerPanel.add(regNameField);

        registerPanel.add(new JLabel("Email:"));

        registerPanel.add(regEmailField);

        registerPanel.add(new JLabel("Password:"));

        registerPanel.add(regPasswordField);

        registerPanel.add(new JLabel("Phone Number:"));

        registerPanel.add(regPhoneField);

        registerPanel.add(new JLabel("Gender:"));

        registerPanel.add(genderPanel);

        registerPanel.add(submitRegisterButton);

        // Main Panel

        JButton vehicleRegButton = new JButton("Vehicle Registration");

        JButton reportButton = new JButton("Report");

        JButton searchButton = new JButton("Search");

        JButton bookSlotButton = new JButton("Book Slot");

        JButton logoutButton = new JButton("Log out");

        mainPanel.add(vehicleRegButton);

        mainPanel.add(reportButton);

        mainPanel.add(searchButton);

        mainPanel.add(bookSlotButton);

        mainPanel.add(logoutButton);

        // Vehicle Panel

        JTextField modelNameField = new JTextField();

        JTextField yearField = new JTextField();

        JTextField companyField = new JTextField();

        JTextField numberPlateField = new JTextField();

        JButton addVehicleButton = new JButton("Add Vehicle");

        JTextField removePlateField = new JTextField();

        JButton removeVehicleButton = new JButton("Remove Vehicle");

        JButton showVehiclesButton = new JButton("Show Vehicles");

        JButton backToMainButton = new JButton("Back to Main Menu");

        vehiclePanel.add(new JLabel("Model Name:"));

        vehiclePanel.add(modelNameField);

        vehiclePanel.add(new JLabel("Year:"));

        vehiclePanel.add(yearField);

        vehiclePanel.add(new JLabel("Company:"));

        vehiclePanel.add(companyField);

        vehiclePanel.add(new JLabel("Number Plate:"));

        vehiclePanel.add(numberPlateField);

        vehiclePanel.add(addVehicleButton);

        vehiclePanel.add(new JLabel("Number Plate to Remove:"));

        vehiclePanel.add(removePlateField);

        vehiclePanel.add(removeVehicleButton);

        vehiclePanel.add(showVehiclesButton);

        vehiclePanel.add(backToMainButton);

        // Report Panel

        JButton dailyEarningsButton = new JButton("Daily Earnings Report");

        JButton vehicleUsageButton = new JButton("Vehicle Usage Report");

        JButton slotUsageButton = new JButton("Slot Usage Report");

        JButton regVehiclesButton = new JButton("Registered Vehicles Report");

        JButton backButton = new JButton("Back");

        reportPanel.add(dailyEarningsButton);

        reportPanel.add(vehicleUsageButton);

        reportPanel.add(slotUsageButton);

        reportPanel.add(regVehiclesButton);

        reportPanel.add(backButton);

        add(loginPanel, "loginPanel");

        add(registerPanel, "registerPanel");

        add(mainPanel, "mainPanel");

        add(vehiclePanel, "vehiclePanel");

        add(reportPanel, "reportPanel");

        // Event Handling

        loginButton.addActionListener(e -> {

            String email = loginEmailField.getText();

            String password = new String(loginPasswordField.getPassword());

            Login login = new Login(email, password);

            if (login.login(registeredUser)) {

                JOptionPane.showMessageDialog(null, "Login successful!");

                isLogin = true;

                cardLayout.show(getContentPane(), "mainPanel");

            } else {

                JOptionPane.showMessageDialog(null, "Login failed. Incorrect email or password.");

                login.forgetPassword(registeredUser);

            }

        });

        registerButton.addActionListener(e -> cardLayout.show(getContentPane(), "registerPanel"));

        submitRegisterButton.addActionListener(e -> {

            String name = regNameField.getText();

            String email = regEmailField.getText();

            String password = new String(regPasswordField.getPassword());

            String phoneNumber = regPhoneField.getText();

            String gender = maleRadioButton.isSelected() ? "Male" : "Female";

            registeredUser = new Register();

            registeredUser.registerCustomer(name, email, password, phoneNumber, gender);

            cardLayout.show(getContentPane(), "loginPanel");

        });

        vehicleRegButton.addActionListener(e -> cardLayout.show(getContentPane(), "vehiclePanel"));

        reportButton.addActionListener(e -> cardLayout.show(getContentPane(), "reportPanel"));

        searchButton.addActionListener(e -> {

            String numberPlate = JOptionPane.showInputDialog("Enter the Number Plate of the vehicle:");

            if (numberPlate != null) {

                Vehicle[] vehicles = vehicleRegistration.getVehicles();

                boolean found = false;

                for (Vehicle vehicle : vehicles) {

                    if (vehicle.getNumberPlate().equals(numberPlate)) {

                        JOptionPane.showMessageDialog(null, vehicle.toString());

                        found = true;

                        break;

                    }

                }

                if (!found) {

                    JOptionPane.showMessageDialog(null, "Vehicle not found.");

                }

            }

        });

        bookSlotButton.addActionListener(e -> {

            Vehicle[] vehicles = vehicleRegistration.getVehicles();

            if (vehicles.length == 0) {

                JOptionPane.showMessageDialog(null, "No registered vehicles found. Please register a vehicle first.");

                return;

            }

            String[] vehicleOptions = new String[vehicles.length];

            for (int i = 0; i < vehicles.length; i++) {

                vehicleOptions[i] = vehicles[i].toString();

            }

            String selectedVehicle = (String) JOptionPane.showInputDialog(

                    null,

                    "Select the vehicle to book a slot for:",

                    "Book Slot",

                    JOptionPane.PLAIN\_MESSAGE,

                    null,

                    vehicleOptions,

                    vehicleOptions[0]

            );

            if (selectedVehicle != null) {

                String hoursStr = JOptionPane.showInputDialog("Enter the number of hours for booking:");

                int hours = Integer.parseInt(hoursStr);

                BookSlot bookSlot = new BookSlot();

                bookSlot.bookSlot(hours);

                Payment payment = new Payment(20.0);

                double bill = payment.generateBill(hours);

                totalEarnings += bill;

                JOptionPane.showMessageDialog(null, "Slot booked for " + hours + " hours.\nTotal amount to be paid: $" + bill);

            }

        });

        logoutButton.addActionListener(e -> {

            isLogin = false;

            cardLayout.show(getContentPane(), "loginPanel");

        });

        // Vehicle Panel event handling

        addVehicleButton.addActionListener(e -> {

            String modelName = modelNameField.getText();

            int year = Integer.parseInt(yearField.getText());

            String company = companyField.getText();

            String numberPlate = numberPlateField.getText();

            vehicleRegistration.addVehicle(modelName, year, company, numberPlate);

            modelNameField.setText("");

            yearField.setText("");

            companyField.setText("");

            numberPlateField.setText("");

        });

        removeVehicleButton.addActionListener(e -> {

            String numberPlate = removePlateField.getText();

            vehicleRegistration.removeVehicle(numberPlate);

            removePlateField.setText("");

        });

        showVehiclesButton.addActionListener(e -> {

            String vehicles = vehicleRegistration.showVehicles();

            JOptionPane.showMessageDialog(null, vehicles);

        });

        backToMainButton.addActionListener(e -> cardLayout.show(getContentPane(), "mainPanel"));

        // Report Panel event handling

        dailyEarningsButton.addActionListener(e -> {

            String report = "Total Earnings: $" + totalEarnings;

            JOptionPane.showMessageDialog(null, report);

        });

        vehicleUsageButton.addActionListener(e -> {

            String report = vehicleRegistration.showVehicles();

            JOptionPane.showMessageDialog(null, report);

        });

        slotUsageButton.addActionListener(e -> {

            JOptionPane.showMessageDialog(null, "Slot Usage Report is not implemented.");

        });

        regVehiclesButton.addActionListener(e -> {

            String report = vehicleRegistration.showVehicles();

            JOptionPane.showMessageDialog(null, report);

        });

        backButton.addActionListener(e -> cardLayout.show(getContentPane(), "mainPanel"));

        setVisible(true);

    }

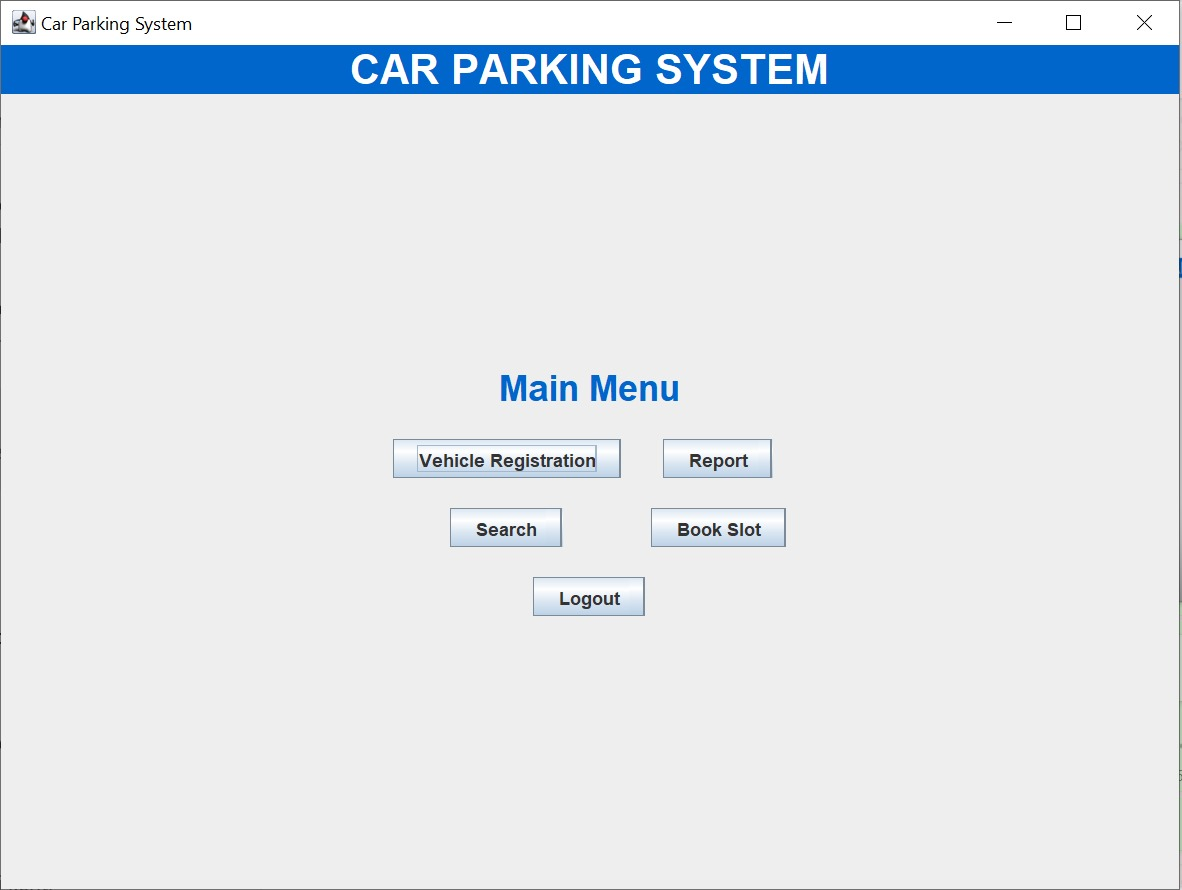
    public static void main(String[] args) {

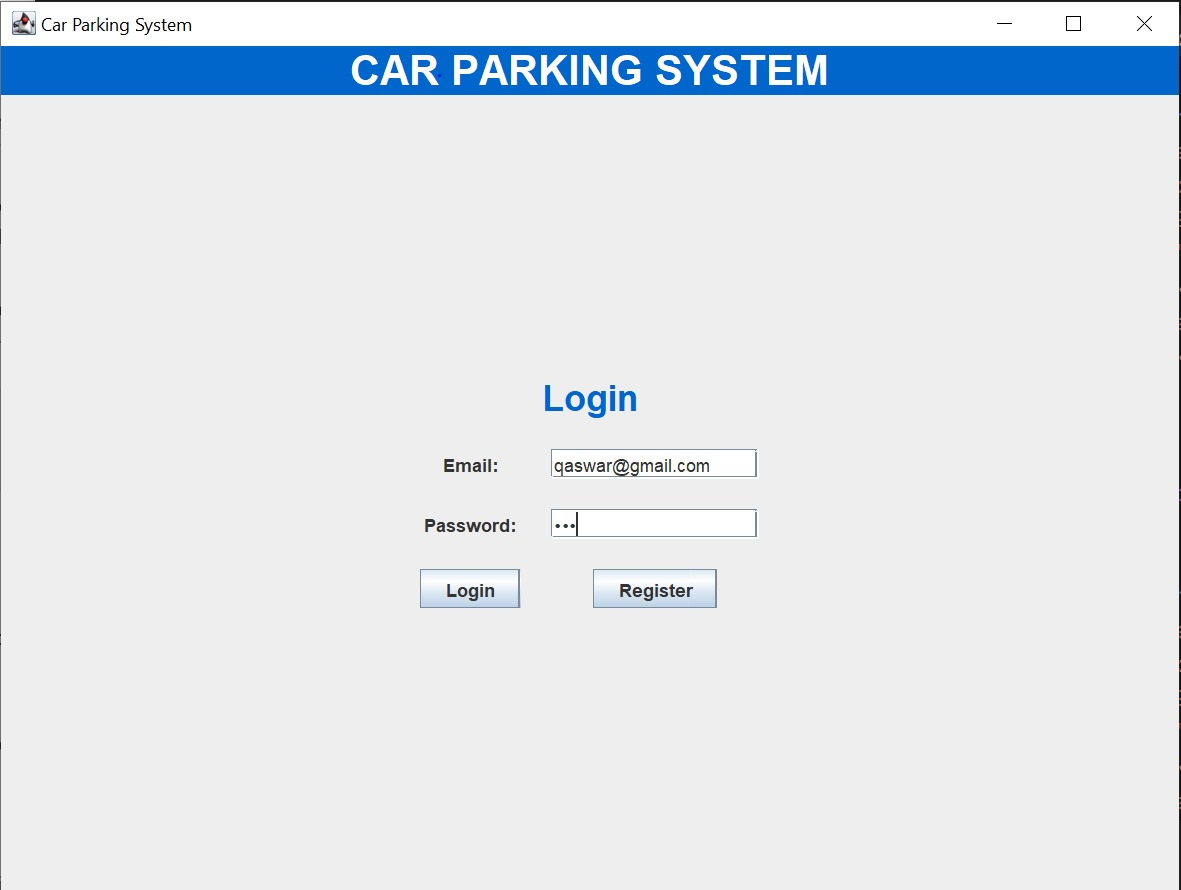
        new CarParkingSystemGUI();

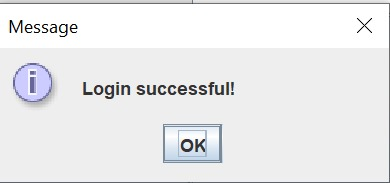
    }

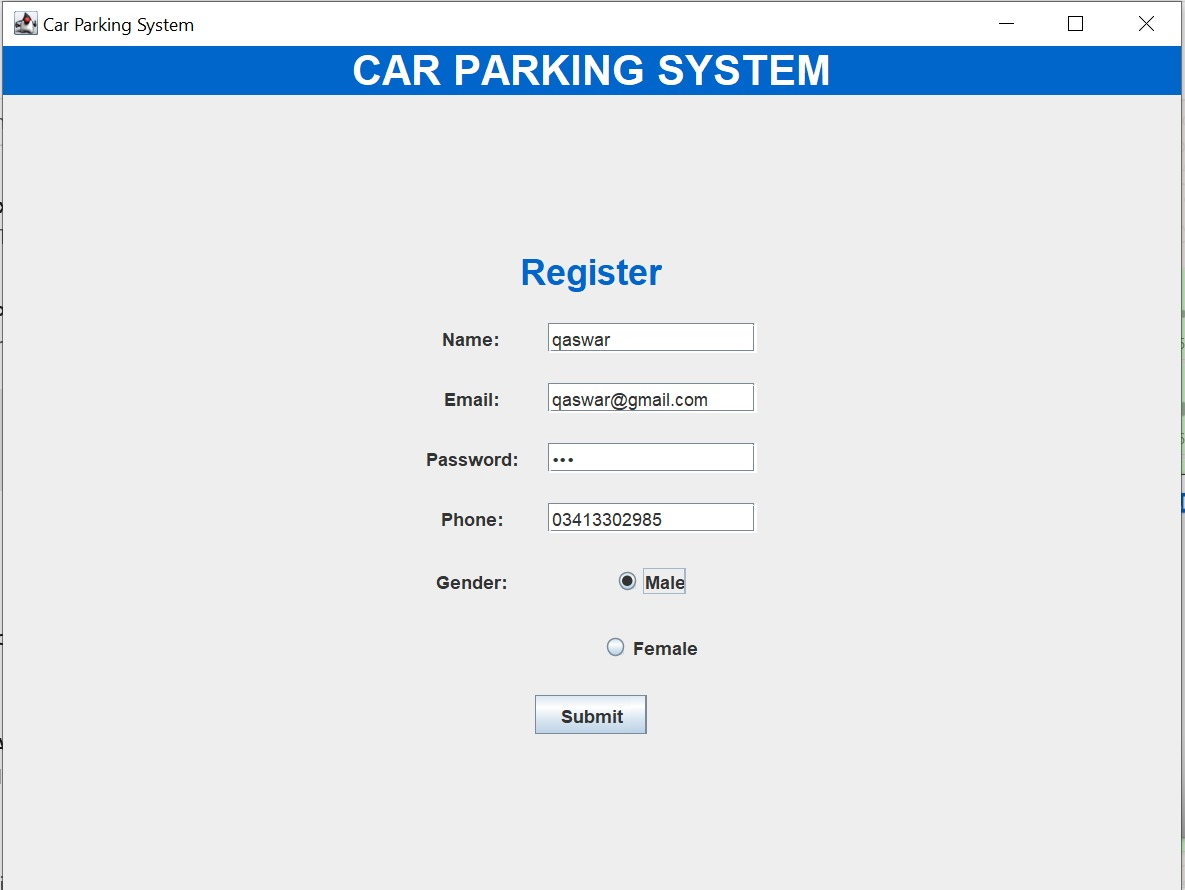
}

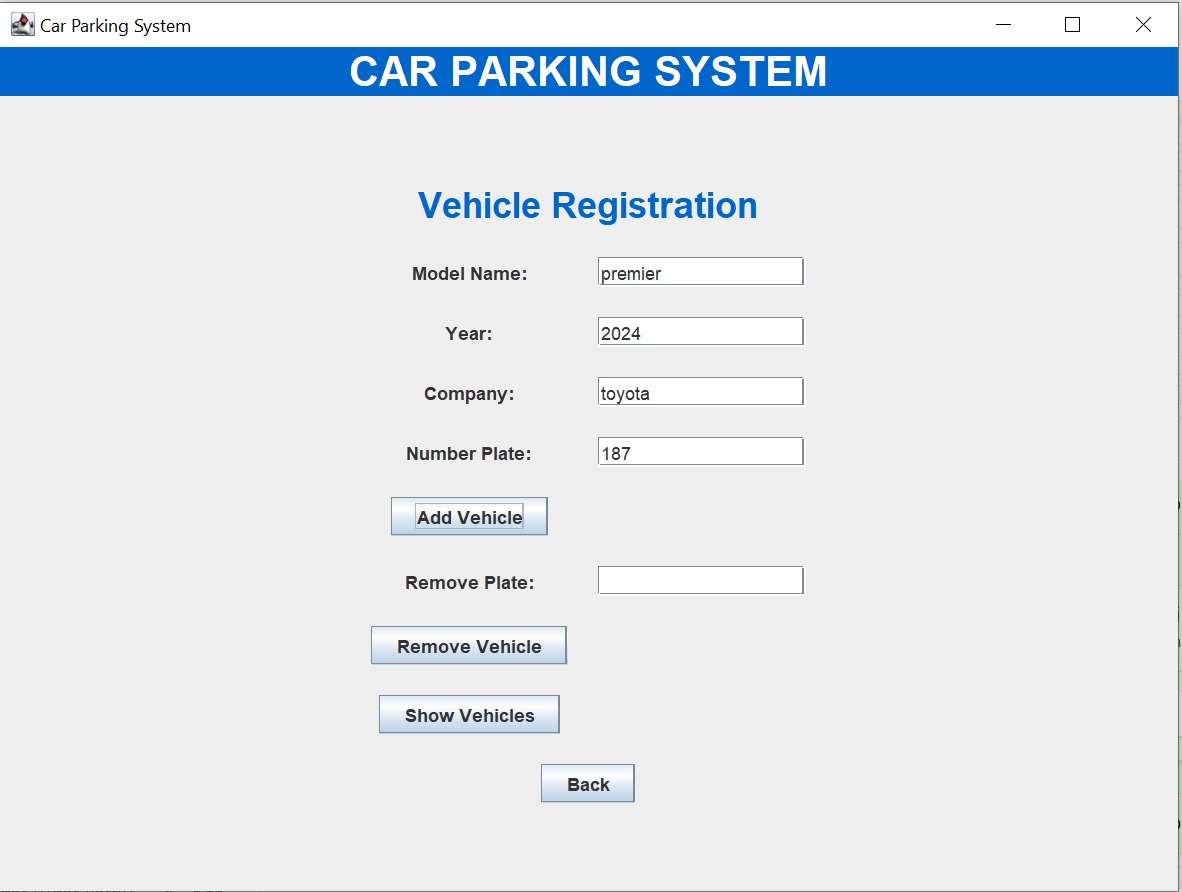
FEATURES(Output) (testing and verifying)

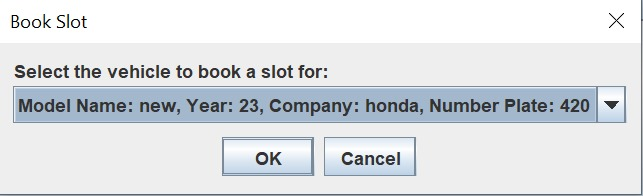


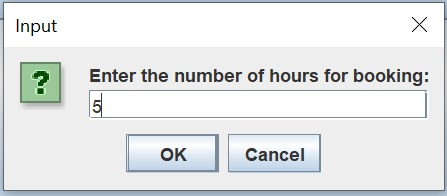


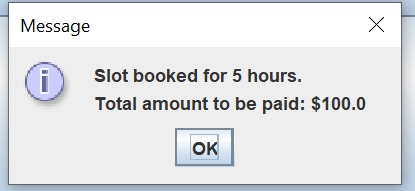












A screenshot of a computer screen

Description automatically generated

CONCLUSION:

The Car Parking System GUI provides a comprehensive solution for managing vehicle registrations, booking parking slots, and handling customer accounts with an intuitive graphical interface. The application leverages Java Swing for the GUI and encapsulates different functionalities into well-defined classes, ensuring modularity and maintainability.

SUGGESTIONS:

….………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………