Induvidual ASSIGNMENT

COMP40003

SOFTWARE DEVELOPMENT AND APPLICATION MODELLING

COMP40003 SDAM 2 Induvidual Assignment
CFK23A1COM

HAND OUT DATE: 18-03-2024

HAND IN DATE: 10-06-2024

WEIGHTAGE: 50%

INSTRUCTION TO CANDIDATES:

1. **Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).**
2. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
3. **Cases of plagiarism will be penalized**
4. **Group assignment report should be submitted in the form of softcopy to the link provided on LMS – one submission per group.**
5. **Individual application should be submitted as a zipped folder to the link provided on LMS – one submission for each student.**

Acknowledgement

I would like to give my heartiest appreciation to my lecturer, Mr. Pradeep Pallegama, for providing me with this Invaluable opportunity.

His guidance and expertise have given me to explore and gain my Skills about various aeras of this topic.

My gratitude also Giving to my Senior friends, whose who gave support and engagement were helpful in many aeras of my assignment.

Their devotion and commitment were essential in achieving the goals of this project.

In conclusion, I would like to thank the librarian and university library staff for their Full support in providing academic resources. And also, Their support has been a Reason to the success of my work.

Table of Contents

01. Introduction

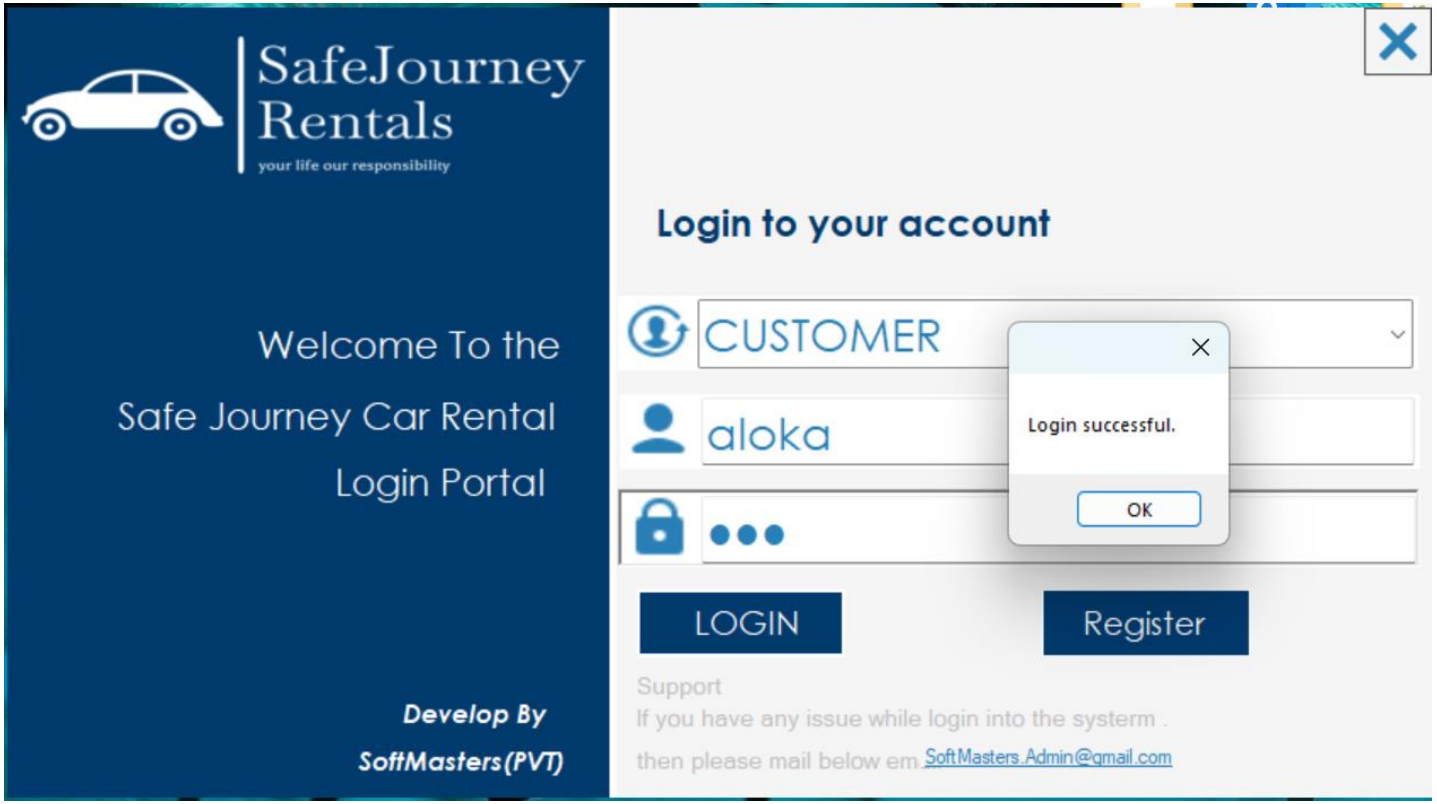
Safe Journey rentals is platform that customer can book a car. Special thing is Our Client can select our available Driver and our vehicle based on their choice. The Safe Journey platform maintains transparency with the client. When the customer adds booking and makes payment and confirm it. System admin can verify customer Booking Details and Payment receipt. After the Verification Admin can send a mail to the customer as a confirmation of the booking. For Customer can keep it as evident.

For Admin He can monitor all the operation through from hos dashboard and manage cars and drivers From his Dashboard. There are separate navigation buttons for that also.

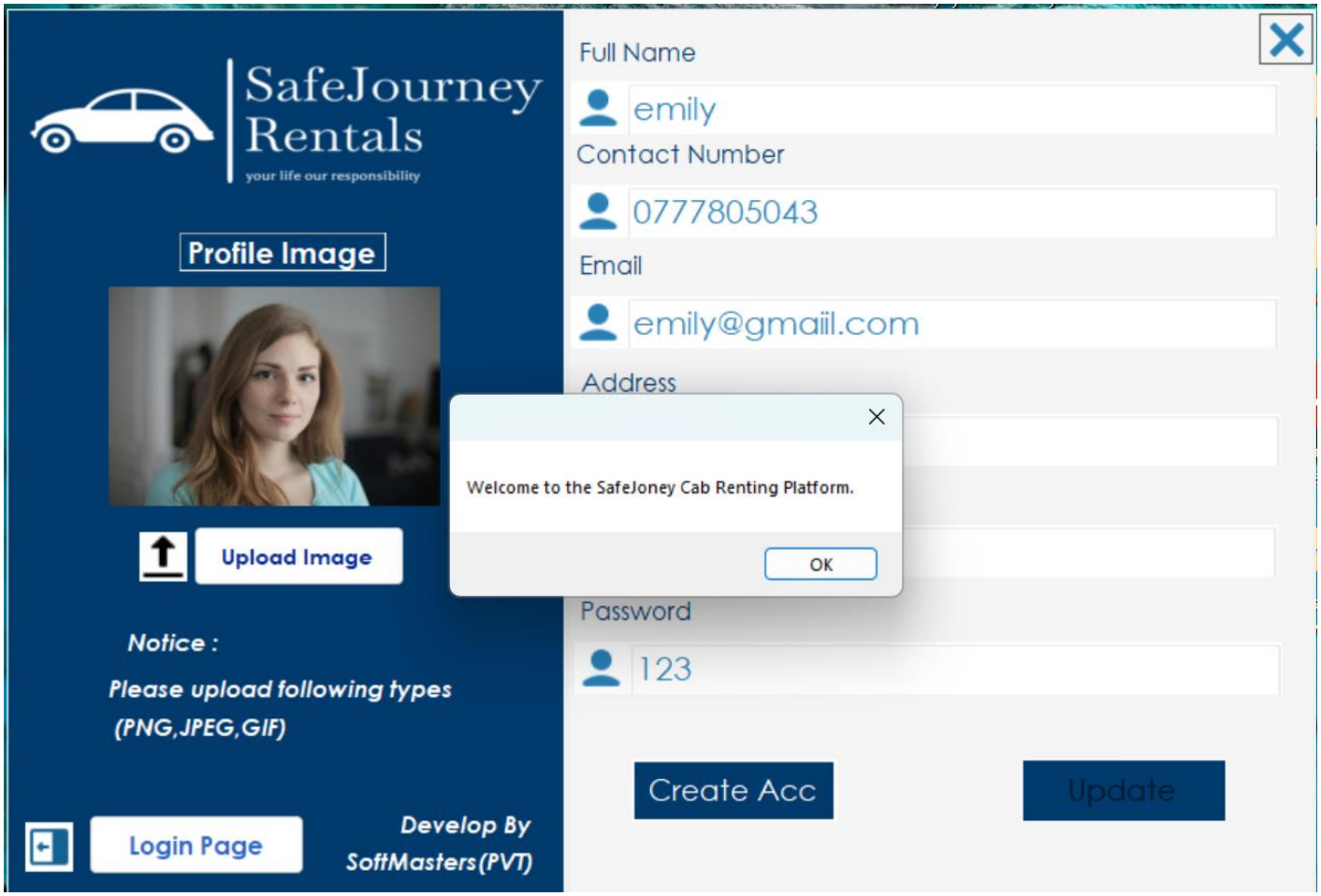
For Driver Can View the latest booking from his dashboard and after he done the job he can confirmed It as job done. See the booking that he has.

02. Screen Shots of Working app.

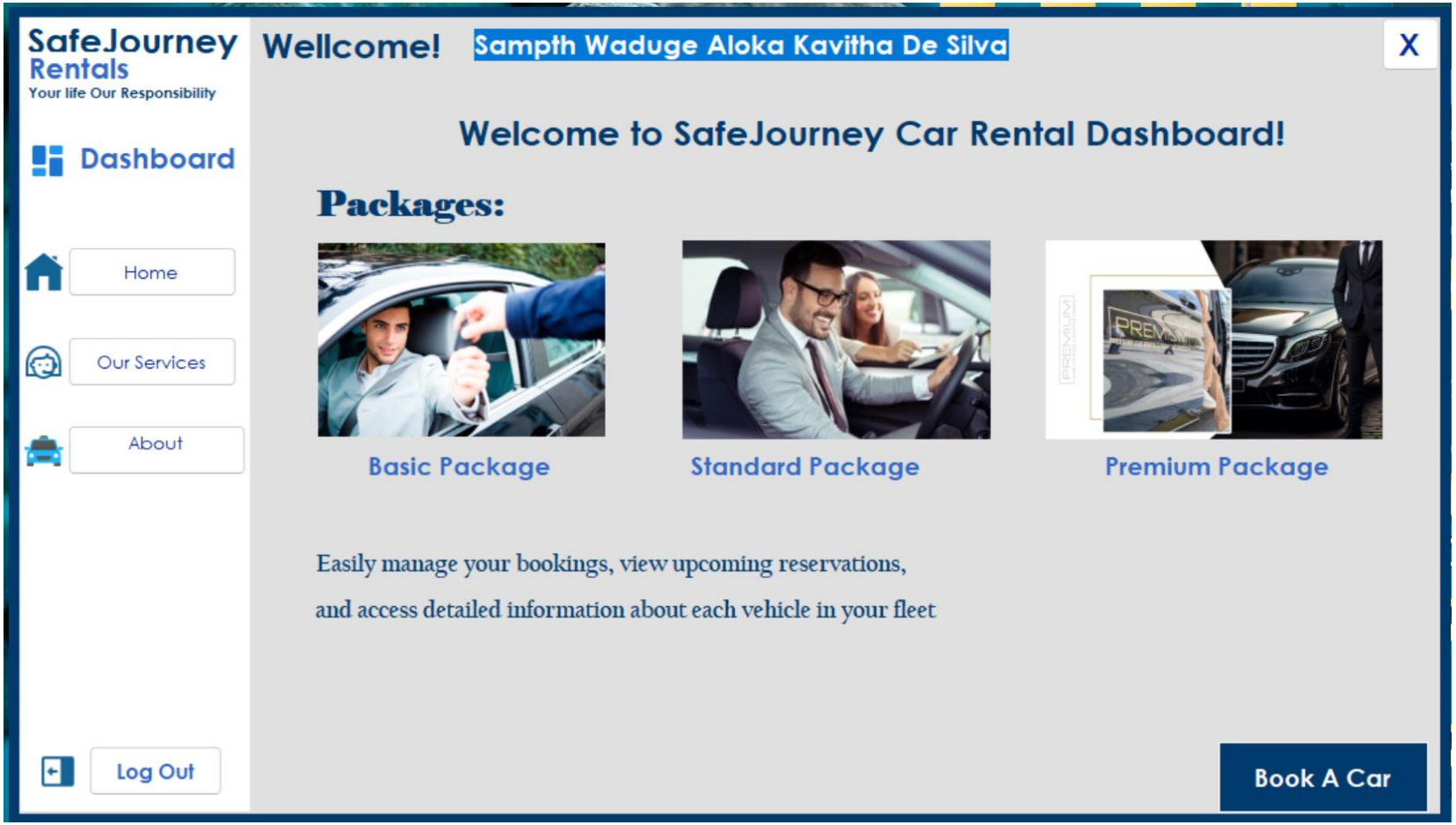
Loging form:



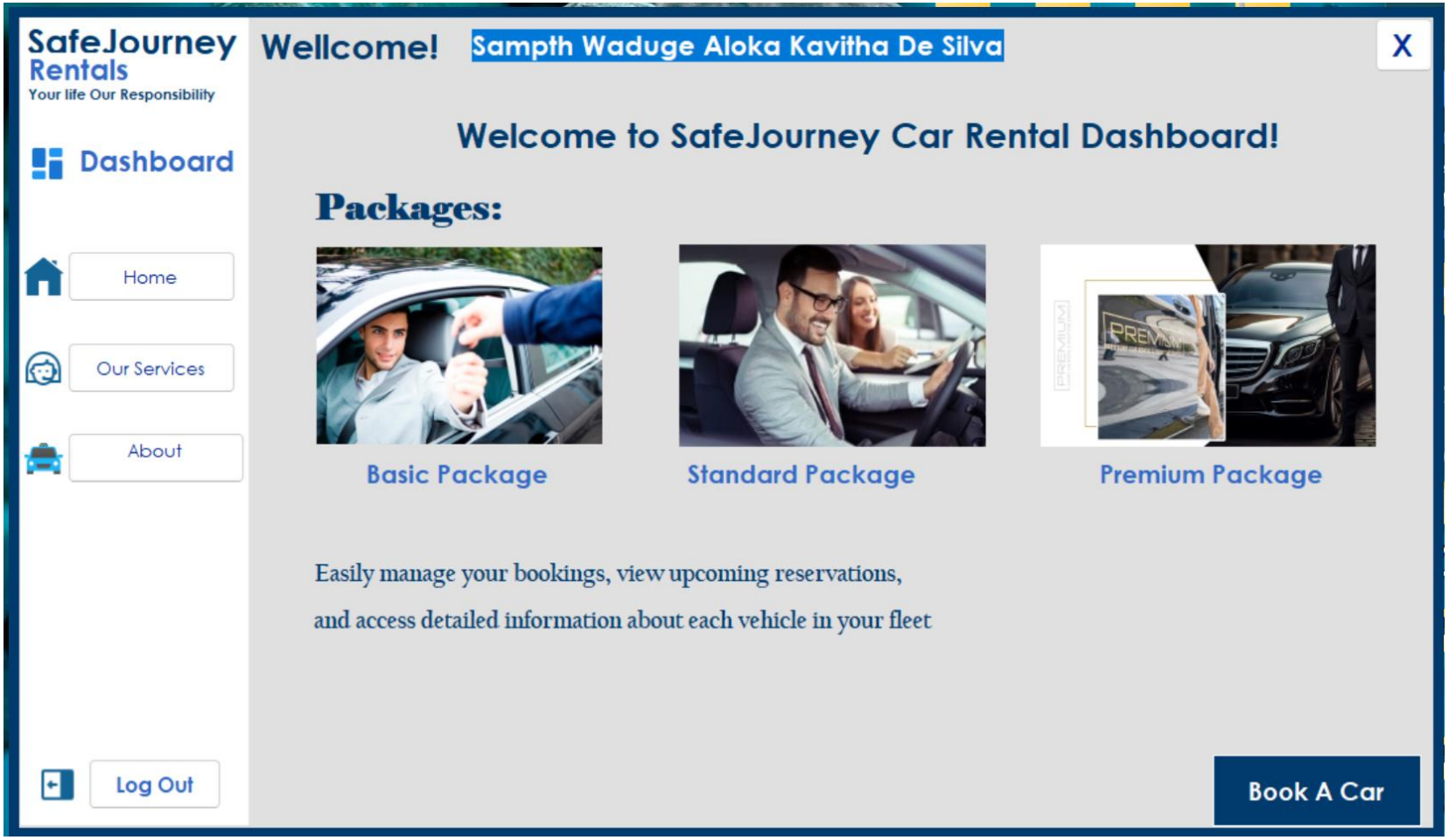
Registration form:



Customer Dashboard form:

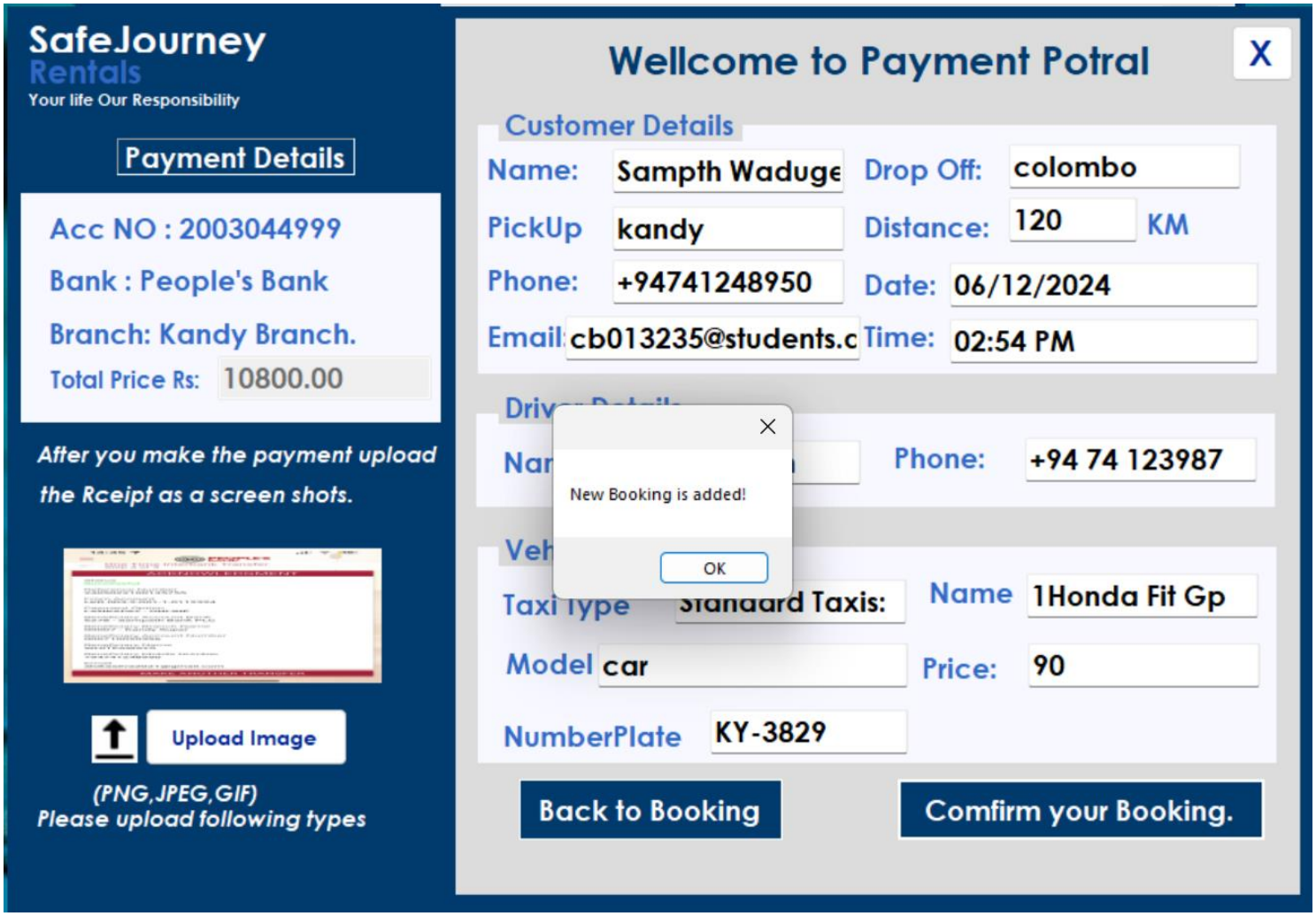


Customer Dashboard form:

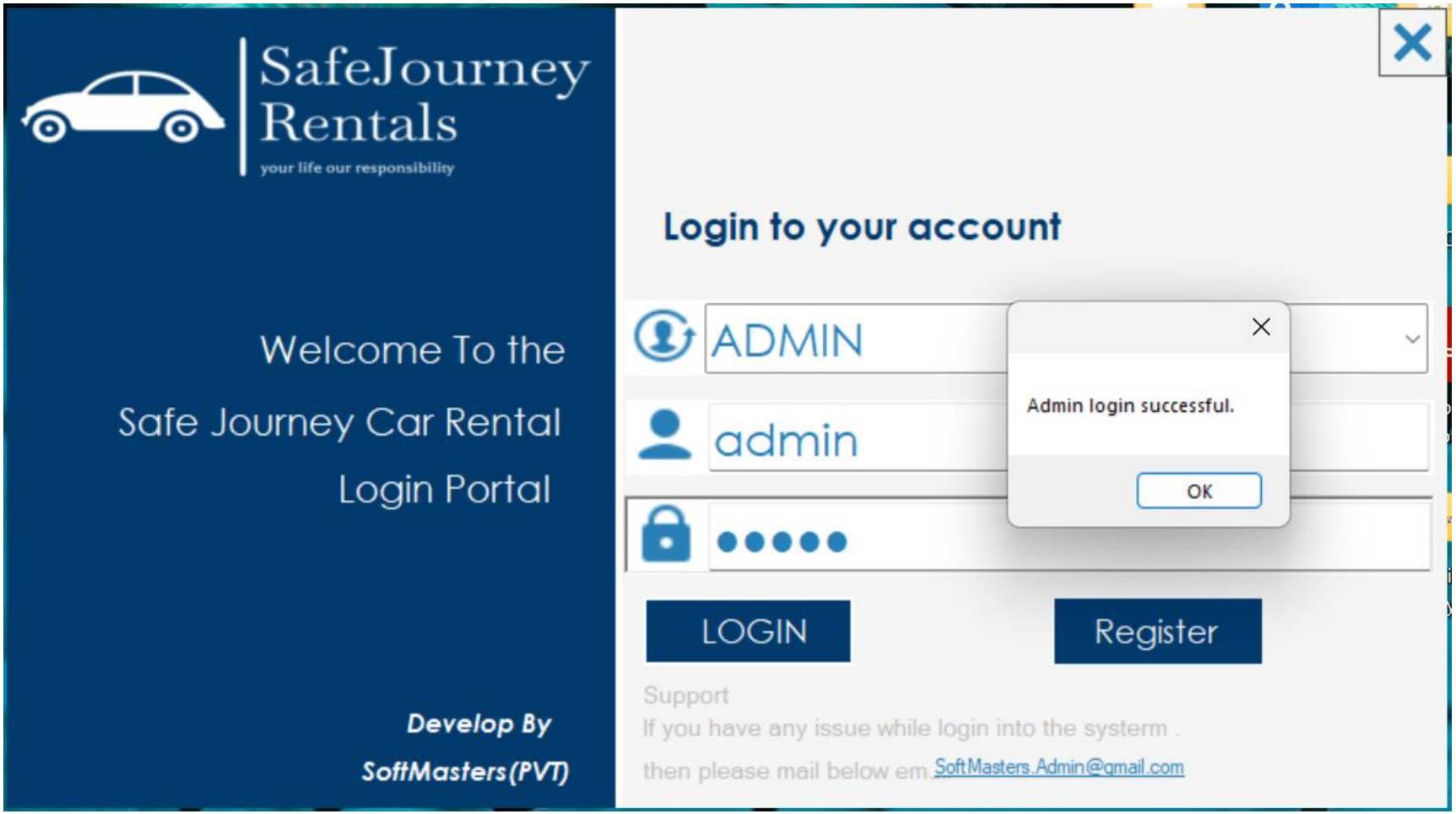


Add Booking form:

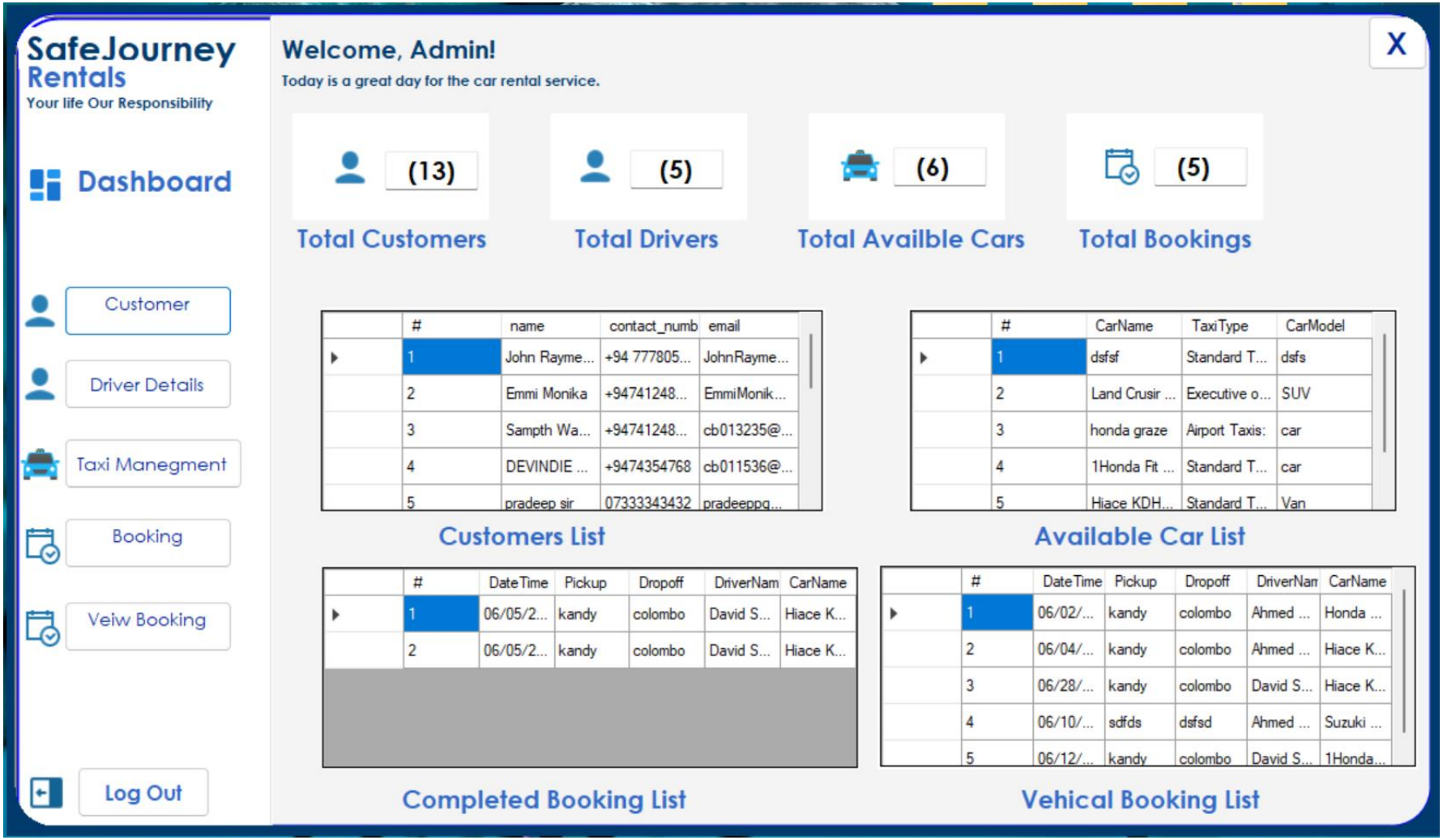
Payment form:



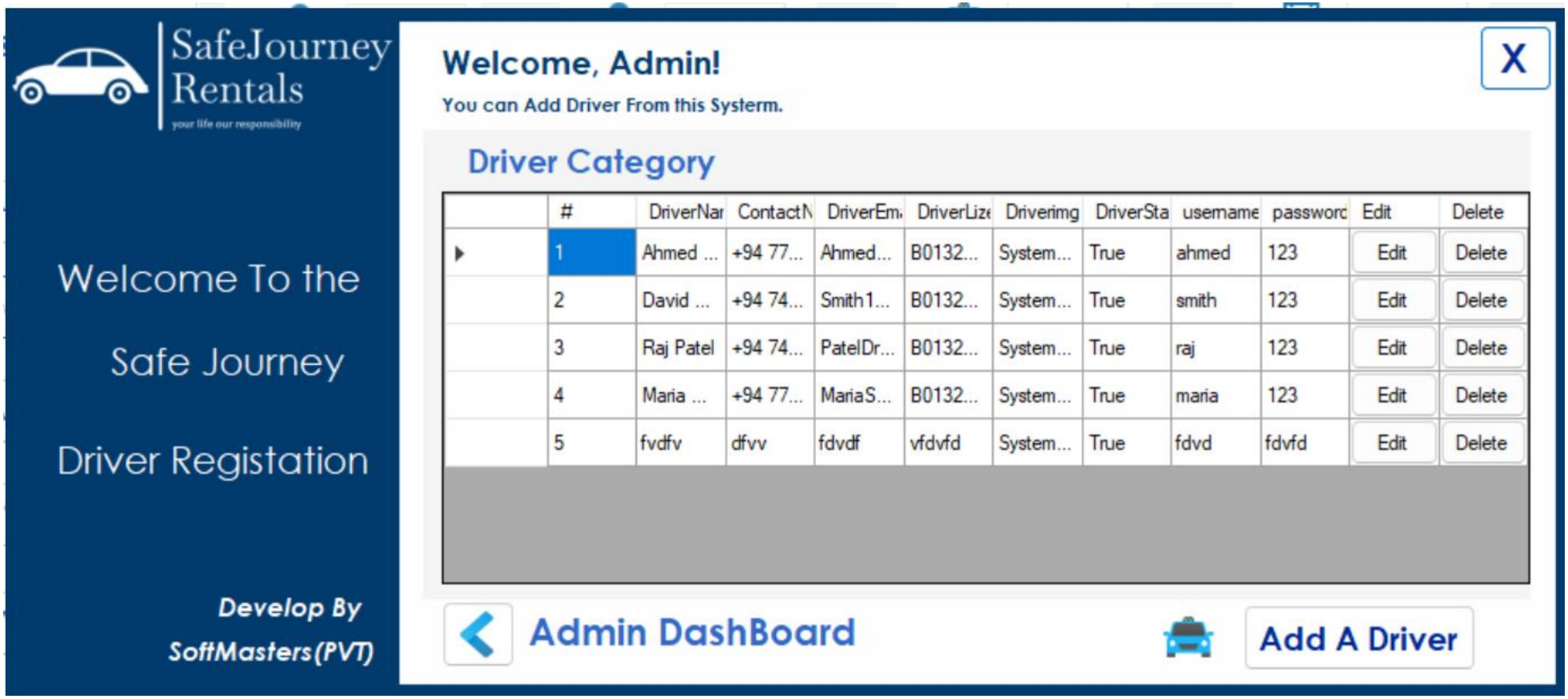
Admin Login form:



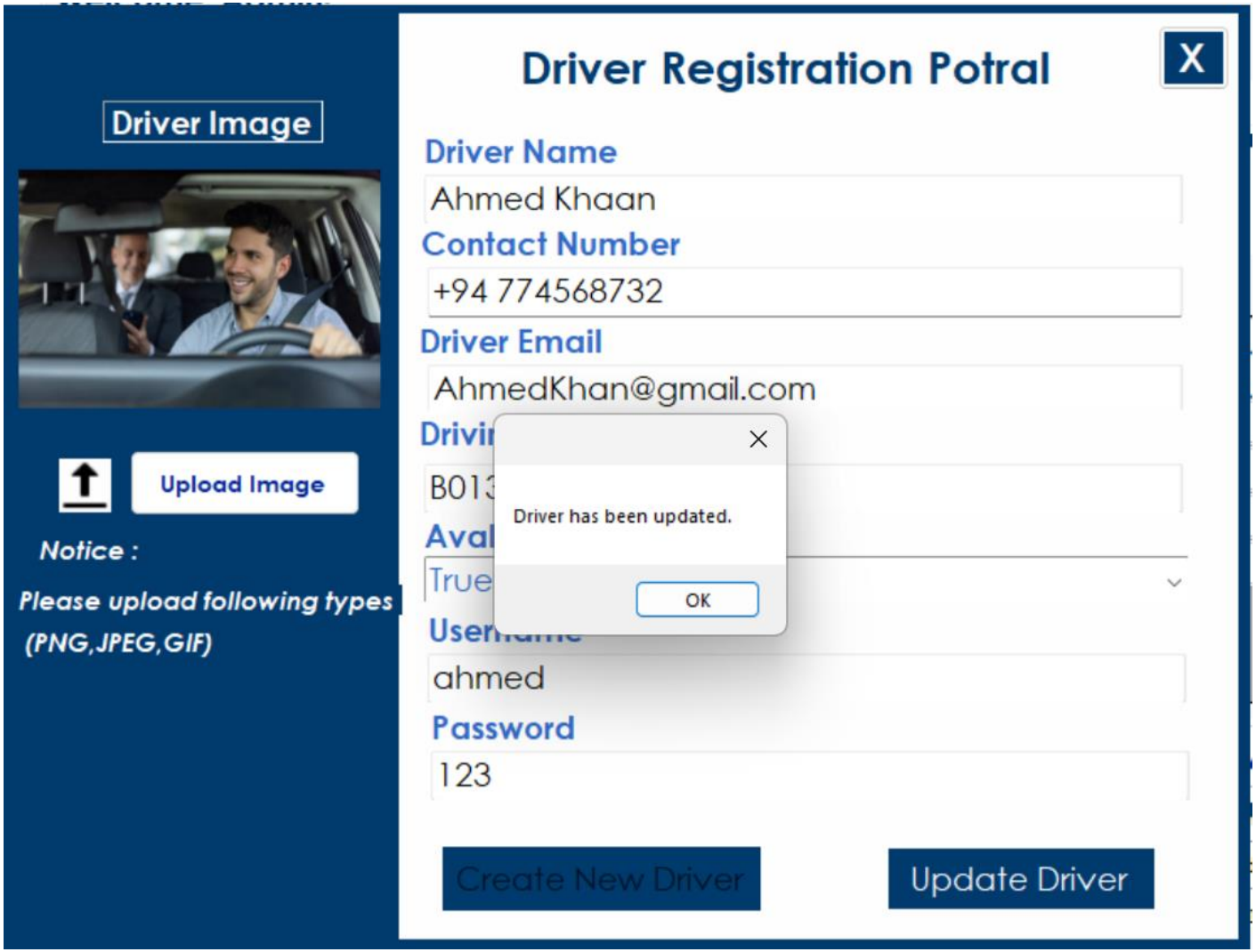
Admin Dashboard form:



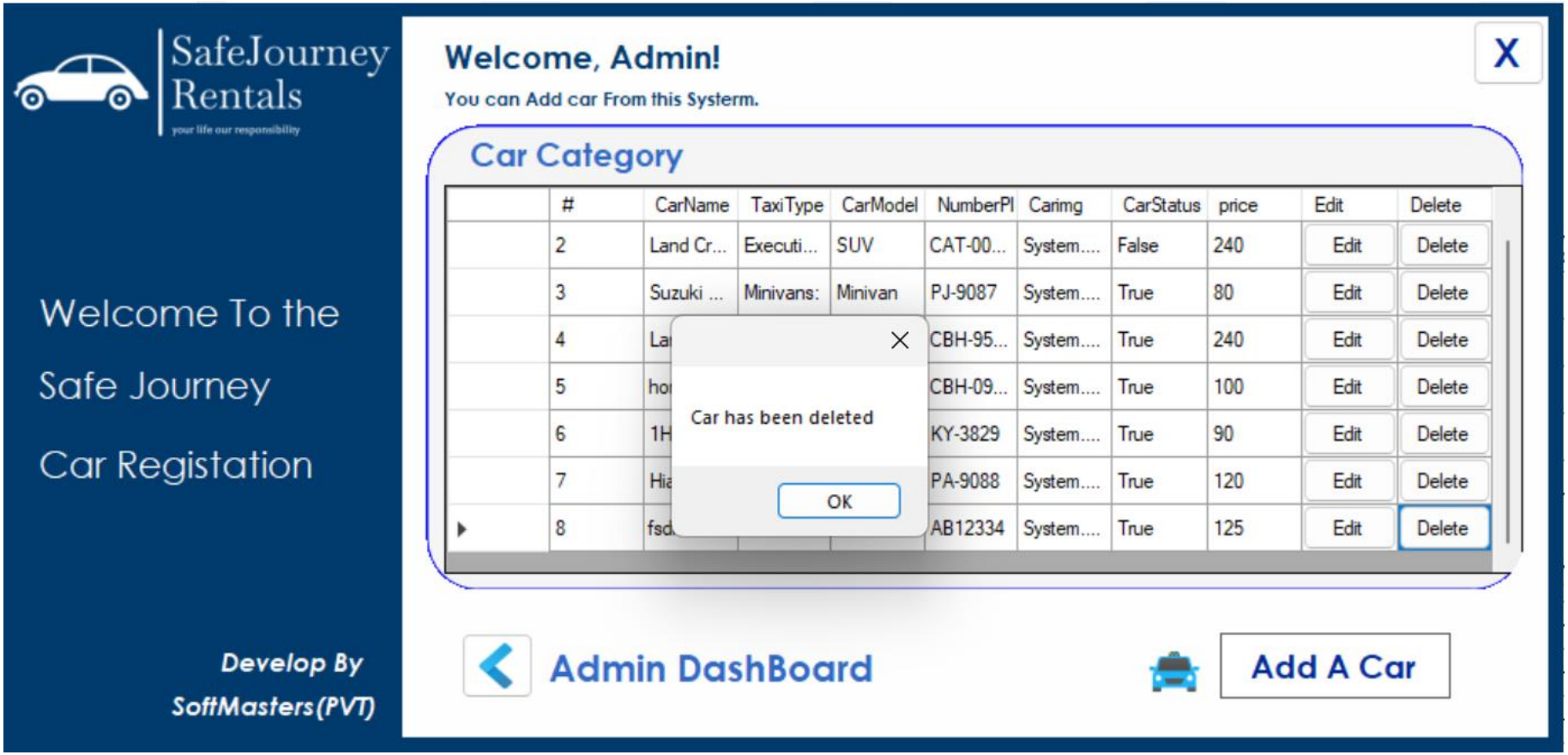
Driver Management form:



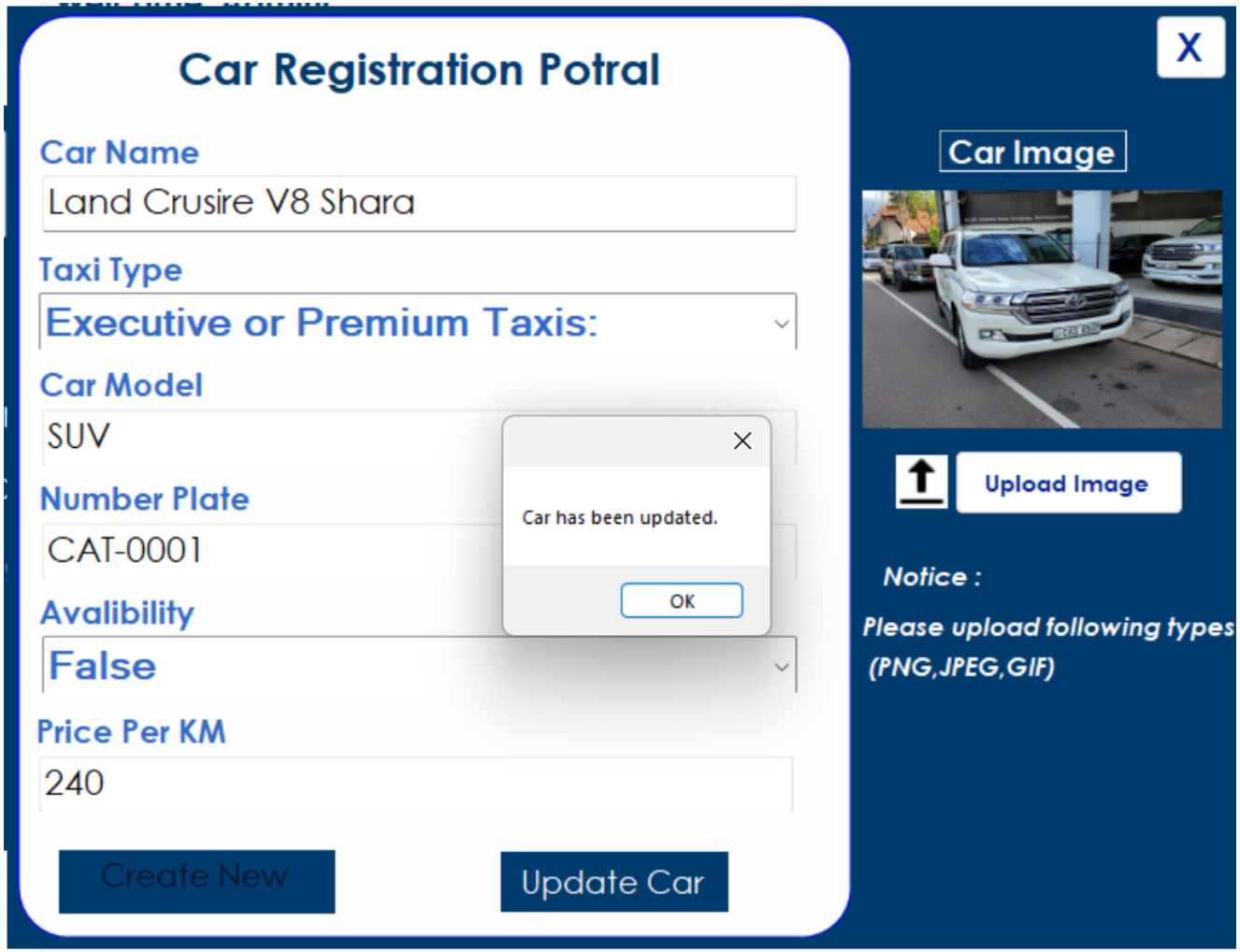
Update Driver form:



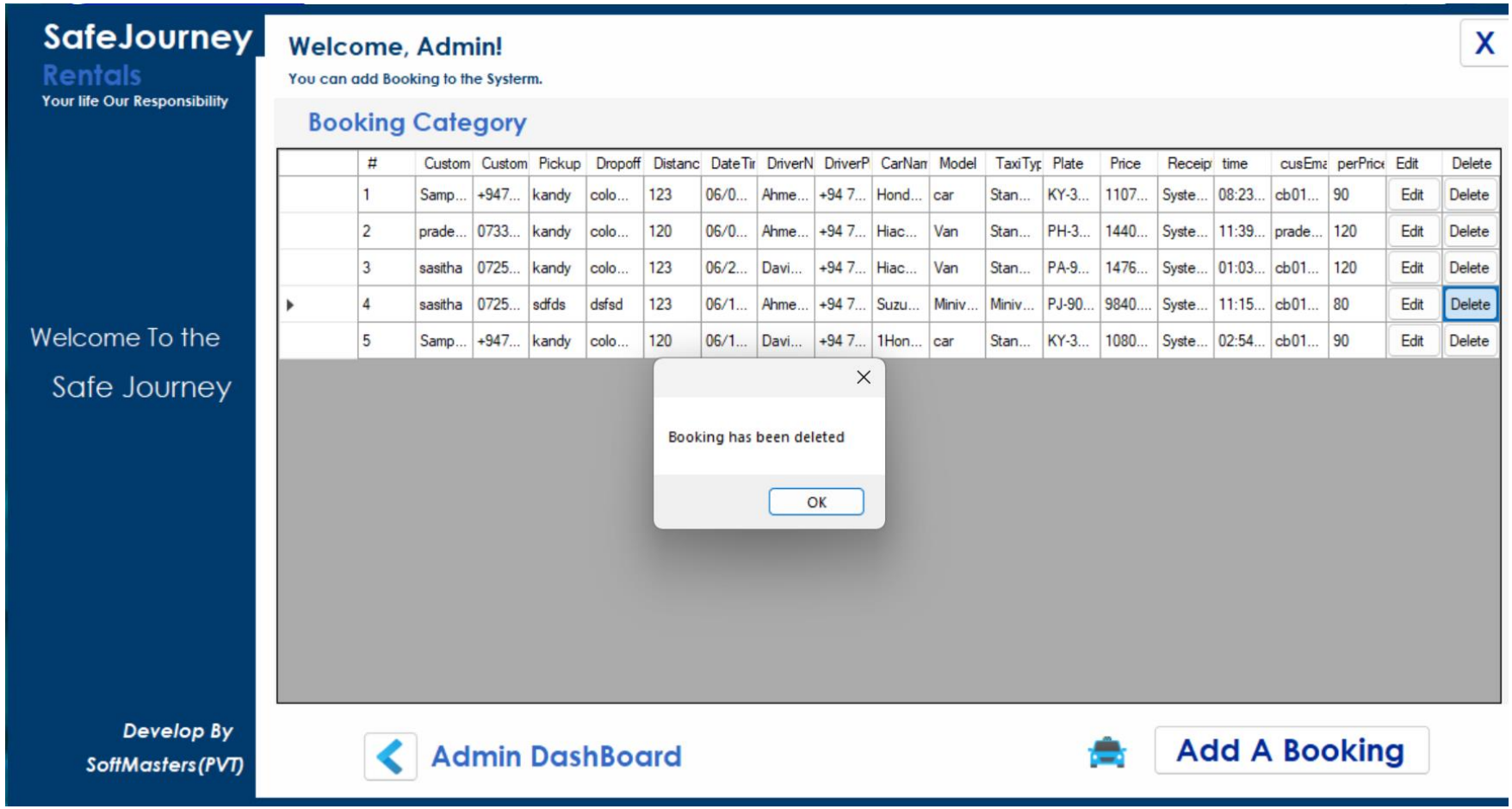
Car Management form:



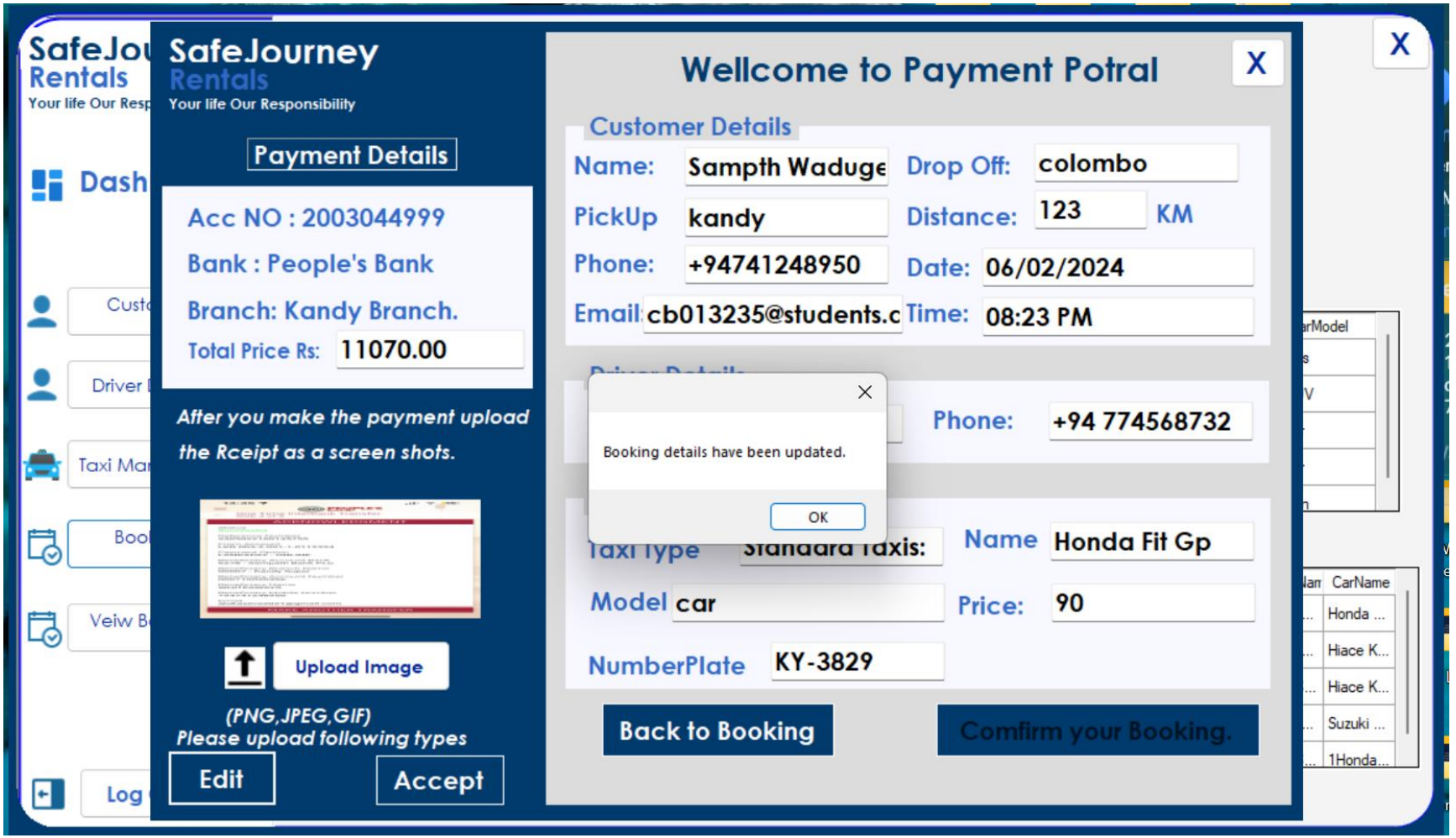
Car Update form:



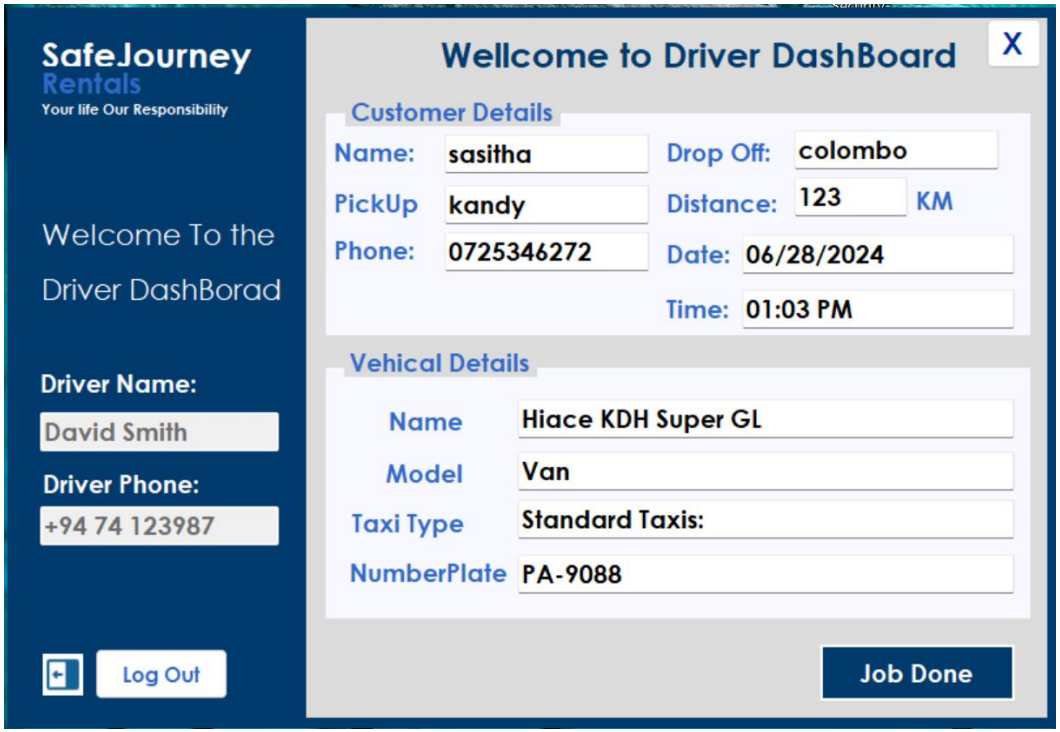
Booking Management form:

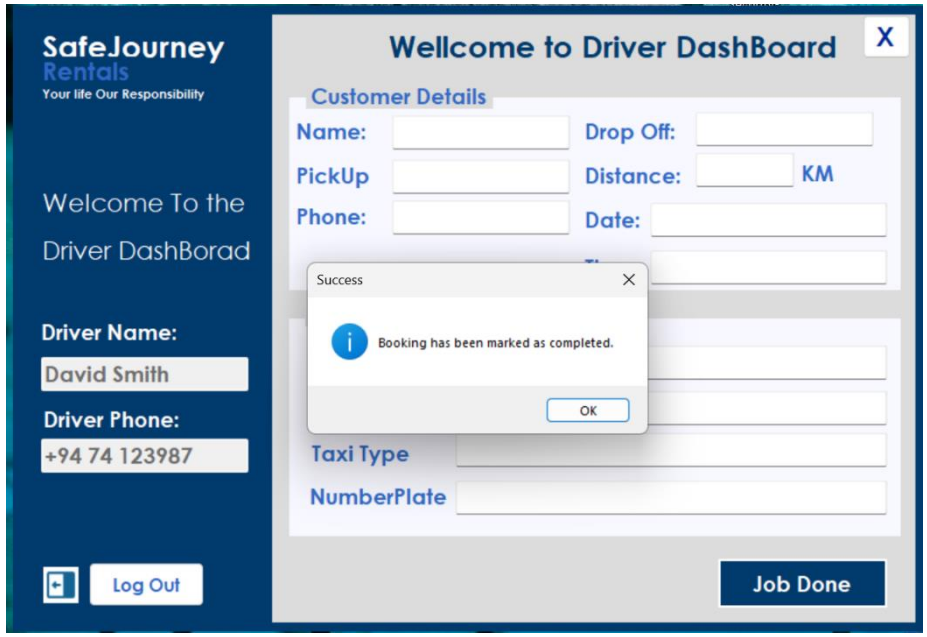


Booking Update form:



Driver Dashbord form:





02. Project Cord.

02.1 Classes:

Person class:

```
        using Cab_Manegment_Systerm2.classes;
using Cab_Manegment_Systerm2.forms;
using Cab_Manegment_Systerm2;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Cab_Manegment_Systerm2.classes
{
    public abstract class Person
    {
        // Propertiees

        public string Email { get; set;}
        public string Address { get; set;}

        public string Username { get; set; }

        public string Password { get; set; }

        // Virtual method to get details
        public virtual string GetDetails()
        {
            return $"Email: {Email}, Address: {Address}, Username: {Username}";
        }
        // Abstract method to get details
       // public abstract string GetDetails();

    }
}
```

Car class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Cab_Manegment_Systerm2.classes
{
    public  class Cab
    {
        // Adding attributes related to the cab

        // Property for Car ID
        public string Car_ID { get; set; }

        // Property for Car Name
        public string Car_Name { get; set; }

        // Property for Taxi Type
        public string Taxi_Type { get; set; }

        // Property for Car Model
        public string Car_Model { get; set; }

        // Private field for Number Plate
        private string _numberPlate;

        // Property for Car Image stored as a byte array
        public byte[] _img { get; set; }

        // Property for Car Availability
        public string Car_Availability { get; set; }

        // Private field for Price
        private string _price;



        // Property for Number Plate with validation
        public string Number_Plate
        {
            get => _numberPlate;
            set
```

```
            {
                // Validate the number plate using a numberplate pattern
                if (!Regex.IsMatch(value, @"^[A-Z0-9-]+$"))
                {
                    throw new ArgumentException("Invalid Number Plate.");
                }
                _numberPlate = value; // Set the value if it is valid
            }
        }

        // Property for Price with validation
        public string Price
        {
            get => _price;
            set
            {
                // Validate the price to ensure it is a valid decimal
                if (!decimal.TryParse(value, out _))
                {
                    throw new ArgumentException("Invalid price.");
                }
                _price = value; // Set the value if it is valid
            }
        }

        //    // Abstract method for adding a car
        //    public abstract bool AddCar(string carName, string taxiType, string carModel, string numberPlate, byte[] carImage, string carStatus, string
price);

        //}
    }
}
```

- For other class like, driver, admin and order also have.

Error Handling:

For Add Booking Class.

```
//load available drivers
2 references
public void LoadDrivers(DateTime bookingDate)
{
    try
    {
        //load drivers to the combo box
        comboDriver.Items.Clear();
        List<string> driverNames = _db.GetAvailableDrivers(bookingDate);
        comboDriver.Items.AddRange(driverNames.ToArray());
    }
    catch (Exception ex)
    {
        //error hadelling massage box.
        MessageBox.Show($"An error occurred while loading drivers: {ex.Message}"
    }
}
```

```
//load available cars
2 references
public void LoadCar(string carType, DateTime bookingDate)
{
    try
    {
        //ensure that taxi type is not null
        if (string.IsNullOrEmpty(carType))
        {
            MessageBox.Show("Car type cannot be null or empty.", "Error", MessageBoxB
            return;
        }
        //add into a Combo box
        comboVehical.Items.Clear();
        List<string> carModels = _db1.GetCarModels(carType, bookingDate);
        comboVehical.Items.AddRange(carModels.ToArray());
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred while loading car models: {ex.Message}");
    }
}
```

```
private void comboType_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        //to get the taxi type from combo box
        selectedType = comboType.SelectedItem?.ToString();
        //ensure that taxi type is not null
        if (string.IsNullOrEmpty(selectedType))
        {
            MessageBox.Show("Please select a valid taxi type.", "Error", MessageBox
            return;
        }
        LoadCar(selectedType, dateTimePicker1.Value);
    }
    //error hadelling
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred while handling the selected index chang
    }
}
```

```
//forword parameters to Payments form.
1 reference
private void ForwardBookingDetails()
{
    try
    {
        // Manually trigger Distance_TextChanged to ensure price is calculated
        Distance_TextChanged(this, EventArgs.Empty);

        //get all booking details
        string calPrice = CalPrice.Text;
        string selectedType = this.selectedType;
        string carName = comboVehical.Text;
        string selectedDriverName = comboDriver.SelectedItem?.ToString();
        string distance = Distance.Text;
        string pickup = txtPickUp.Text;
        string dropoff = txtdropoff.Text;
        //get the date
        DateTime selectedDate = dateTimePicker1.Value.Date;
        DateTime selectedTime = dateTimePicker2.Value;

        // Check if any required field is null or empty
        if (string.IsNullOrEmpty(calPrice))
        {
            MessageBox.Show("Calculated Price is empty.", "Error", MessageBoxButt
            return;
        }
        if (string.IsNullOrEmpty(selectedType))
        {
```

For Car Db Helper Class.

```
using System.IO;
using System.Configuration;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Imaging;
using System.Collections.Generic;

namespace Cab_Manegment_Systerm2.classes
{
    24 references
    public class DbCars : IDisposable
    {
        // Private fields for database connection components
        private string connectionString; // Connection string to the database
        private SqlConnection _cn; // SQL connection
        private SqlCommand _cm; // SQL command
        private SqlDataAdapter _da; // SQL data adapter

        private SqlDataReader _dr; // SQL data reader

        private DataTable _dt; // DataTable to hold data

        // Public static fields for global use
        public static int _id = 0;
        public static byte[] _img;

        // Property to expose the connection string
        1 reference
        public string ConnectionString
        {
            get { return connectionString; }
        }

        // Constructor to initialize the database connection
        8 references
        public DbCars()
        {
            // Initialize the connection string from configuration
            connectionString = ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString;
            _cn = new SqlConnection(connectionString);
            _cm = new SqlCommand { Connection = _cn };
            _da = new SqlDataAdapter(); // Initialize the SQL data adapter
            _dt = new DataTable(); // Initialize the DataTable
        }
```

Get the connection

```
    //get today available cars this is for admin dashborad.
    1 reference
    public DataTable GetAvailableCars()
    {
        // Create a new DataTable to store the results
        DataTable carTable = new DataTable();

        // Get today's date and time
        DateTime bookingDate = DateTime.Now;

        try
        {
            // Create a new SQL connection using the connection string
            using (SqlConnection cn = new SqlConnection(connectionString))
            {
                // this for get available car on today to get that have join with bookings to verify that
                string query = @"
                SELECT d.CarName, d.TaxiType, d.CarModel
                FROM Cars d
                LEFT JOIN Bookings b ON d.CarName = b.CarName AND CONVERT(VARCHAR, b.DateTime, 101) = @BookingDate
                WHERE d.CarStatus = 1 AND b.CarName IS NULL";

                // Create a new SQL command using the query and connection
                using (SqlCommand cmd = new SqlCommand(query, cn))
                {
                    // Add the booking date parameter to the SQL command
                    cmd.Parameters.AddWithValue("@BookingDate", bookingDate.ToString("MM/dd/yyyy")); // Ensure

                    // Create a data adapter to fill the DataTable with the query results
                    SqlDataAdapter da = new SqlDataAdapter(cmd);
                    da.Fill(carTable);
                }
            }
        }
        catch (Exception ex)
        {
            // Show an error message if an exception occurs
            MessageBox.Show("An error occurred: " + ex.Message);
        }

        // Return the DataTable containing the available cars
        return carTable;
    }
```

get available cars

```
//method to add a car to data base
1 reference
public bool AddCar(Cab cab)
{
    bool isSuccess = false;
    SqlConnection conn = new SqlConnection(connectionString);

    try
    {
        string sql = "INSERT INTO Cars(CarName,TaxiType,CarModel,NumberPlate,Ca
        SqlCommand cmd = new SqlCommand(sql, conn);

            // Create parameters to add
            cmd.Parameters.AddWithValue("@CarName", cab.Car_Name);
            cmd.Parameters.AddWithValue("@TaxiType", cab.Taxi_Type);
            cmd.Parameters.AddWithValue("@CarModel", cab.Car_Model);
            cmd.Parameters.AddWithValue("@NumberPlate", cab.Number_Plate);
            cmd.Parameters.AddWithValue("@Carimg", cab._img);
            cmd.Parameters.AddWithValue("@CarStatus", cab.Car_Availability);
            cmd.Parameters.AddWithValue("@price", cab.Price);

        conn.Open();
            int rows = cmd.ExecuteNonQuery();

            if (rows > 0)
            {
                isSuccess = true;
            }
            else
            {
                isSuccess = false;
            }

    }
    catch (Exception ex)
    {
        // If error occurs, display it in a message box
```

add cars to the data base.

For Manage cars Form. (Crud Operations)

```
//load cars to the data gride view and give crud operation
1 reference
private void DgvCars_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex < 0) return;

    string colName = DgvCars.Columns[e.ColumnIndex].Name;
    //adding edit button the dgv
    if (colName == "ColEdit")
    {
        DbCars._id = (int)DgvCars.CurrentRow.Cells[0].Value;
        FrmAddCar f = new FrmAddCar(this);
        //accesing the text boxes
        f.txtName.Text = DgvCars.CurrentRow.Cells[2].Value.ToString();
        f.comboType.Text = DgvCars.CurrentRow.Cells[3].Value.ToString();
        f.txtModel.Text = DgvCars.CurrentRow.Cells[4].Value.ToString();
        f.txt_NumberPlate.Text = DgvCars.CurrentRow.Cells[5].Value.ToString();
        f.txtPrice.Text = DgvCars.CurrentRow.Cells[8].Value.ToString();
        //load the image
        DbCars.ShowImageInPictureBox(f.pic, DgvCars, 6);
        if (DgvCars.CurrentRow.Cells[7].Value != null)
        {
            f.comboAvailability.Text = DgvCars.CurrentRow.Cells[7].Value.ToString();
        }
        else
        {
            f.comboAvailability.Text = string.Empty; // or some default value
        }
        //make update button available true
        f.Update_Btn.Enabled = true;
        //disable create button
        f.Create_Btn.Enabled = false;

        f.ShowDialog();
    }
```

Edit cars

```
    //giving the delete method to dgv
    else if (colName == "ColDelete")
    {
        try
        {
            using (DbCars dbCars = new DbCars())
            {
                dbCars.OpenConnection();
                //access car by is id in data base
                using (SqlCommand cm = new SqlCommand("DELETE FROM Cars WHERE id = @id", dbCars.GetConnection()))
                {
                    cm.Parameters.AddWithValue("@id", (int)DgvCars.CurrentRow.Cells[0].Value);
                    cm.ExecuteNonQuery();
                }
            }
            MessageBox.Show("Car has been deleted");
            LoadCars();
            //update the admin dashbord
            AdminDashboard dashboard = new AdminDashboard();
            //after that load cars to dgv
            dashboard.LoadCars();
        }
        catch (Exception ex)
        {
            MessageBox.Show("An error occurred: " + ex.Message);
        }
    }
}
```

Delete cars

For User Login Verification (Form Loging)

```
    }
    //login for customers
    else if (selectedUserType == "CUSTOMER")
    {
        try
        {
            using (Db db = new Db())
            {
                db.OpenConnection();

                string query = "SELECT * FROM CustomersProfiles WHERE username = @user
                using (SqlCommand command = new SqlCommand(query, db._cn))
                {
                    command.Parameters.AddWithValue("@username", enteredUsername);
                    command.Parameters.AddWithValue("@password", enteredPassword);

                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        if (reader.HasRows)
                        {
                            reader.Read();
                            LoggedInCustomer = new Customer
                            {
                                CustomerName = reader["name"].ToString(),
                                CustomerContactNumber = reader["contact_number"].ToStr
                            };

                            MessageBox.Show("Login successful.");
                            this.DialogResult = DialogResult.OK;

                            this.Hide();

                            CustomerDashboard f = new CustomerDashboard(LoggedInCustom
                            f.Show();
                            FrmAddBooking b = new FrmAddBooking(LoggedInCustomer);  //
```

```
    //driver loging verfication
    else if (selectedUserType == "DRIVER")
    {
        try
        {
            //get the connection of drivers data base
            using (DbDrivers db = new DbDrivers())
            {
                db.OpenConnection();

                //verfy driver username & password from database
                string query = "SELECT * FROM Drivers WHERE username = @username AND password = @password";
                using (SqlCommand command = new SqlCommand(query, db._cn))
                {
                    command.Parameters.AddWithValue("@username", enteredUsername);
                    command.Parameters.AddWithValue("@password", enteredPassword);

                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        if (reader.HasRows)
                        {
                            reader.Read();
                            LoggedInDriver = new Driver
                            {
                                DriverName = reader["driverName"].ToString(),
                                DriverContactNumber = reader["contactNumber"].ToString(),
                            };

                            MessageBox.Show("Login successful.");
                            this.DialogResult = DialogResult.OK;
                            this.Hide();
                            //load the driver dashborad
                            DriverDashBoard f = new DriverDashBoard(LoggedInDriver);
                            f.Show();
                            //FrmAddBooking b = new FrmAddBooking(LoggedInCustomer);  // Pass the Customer ob

                            //close the connection
                            db.CloseConnection();

                        }
                        else
                        {
                            MessageBox.Show("Invalid username or password for the Driver.");
                        }
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("An error occurred: " + ex.Message);
        }
```

```
    //loging for admin
    if (selectedUserType == "ADMIN")
    {
        try
        {
            using (AdminDbHelper db = new AdminDbHelper())
            {
                db.OpenConnection();

                string query = "SELECT * FROM AdminDetails WHERE Username = @User
                using (SqlCommand command = new SqlCommand(query, db._cn))
                {
                    command.Parameters.AddWithValue("@Username", enteredUsername)
                    command.Parameters.AddWithValue("@Password", enteredPassword)

                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        if (reader.HasRows)
                        {
                            reader.Read();
                            //forword admin name to dashbord
                            //Admin loggedInAdmin = new Admin
                            //{
                            //    //AdminName = reader["name"].ToString(),
                            //   // AdminContactNumber = reader["contact_number"]
                            //};

                            MessageBox.Show("Admin login successful.");
                            this.Hide();

                            AdminDashboard f = new AdminDashboard();
                            f.Show();
                            //close the connection
```

Testing for FrmLoginpage GUI (Customer)

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCOther001 Person Class	Check Username & Password.	Verify user input Username & password before proceeding to the user verification.	The Login form is open. The database contains some user details within tables.	Username Password	First keep filed empty can click login button. Second Fill the username & password and Click the Login Button.	For Empty Username message box appears with the text: "You have to enter a username.". For Empty Password message box appears with the text: "You have to enter a password." Else: validate login	Expected results.	credentials are valid, the user can successfully logged in	Pass
TCCustomer001	Upload a Profile Image	Verify that the Open Image method correctly opens the file explorer dialog and loads the selected image into the picture box.	The Picture Box control is initialized and visible on the form.	Valid image file formats (PNG, BMP, JPG)	Click the upload Button. Opens File Explorer & Select image with valid format. Confirm the selection by clicking “OPEN”.	file explorer dialog opens. user can select an image file with a valid format After confirm display in picture box	Same as Expected results. Files explore open with grep the valid formats.	File explorers need to show only Valid image file formats (PNG, BMP, JPG)	Pass

TCAdmin001	Admin Login Functionality	Verify that the login process for an admin from the database and allow to access admin dashboard upon successful login.	The login form is open. Exceptional handling initialized.	Admin username & password Form the database.	Select the admin for user credentials. Input invalid username & password first. Secondly, input valid username & password.	At the first try it will display an error message “Invalid login for admin”. Second try, its success fully redirects to admin dashboard.	Same as expected Results.	The database connection is established. Need to input valid username & password.	Pass
TCDriver001	Driver Login Functionality	Verify that the login process for a Driver from the database and allow to access Driver dashboard upon successful login.	The login form is open. Exceptional handling initialized.	Driver username & password Form the database.	Select the Driver for user credentials. Input invalid username & password first. Secondly, input valid username & password.	At the first try it will display an error message “Invalid login for Driver”. Second try, its success fully redirects to Driver dashboard.	Same as expected Results.	The database connection is established. Need to input valid username & password.	Pass
TCCustomer002	Customer Login Functionality	Verify that the login process for a customer from the database and allow to access Customer dashboard upon successful login.	The login form is open. Exceptional handling initialized.	Customer username & password Form the database.	Select the Customer for user credentials. Input invalid username & password first. Secondly, input valid username & password.	At the first try it will display an error message “Invalid login for Customer”. Second try, its success fully redirects to Customer dashboard.	Same as expected Results.	The database connection is established. Need to input valid username & password.	Pass

Testing for FrmCustomerRegister GUI (Customer)

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCustomer003	Check Email Uniqueness	Verify user input email address is not already registered in the database.	The customer registration form is open. The database contains some customer records with existing emails.	Existing Email New Email	Enter Existing Email and create. Enter a New Email and create.	For Existing ones: display massage box “this email already exists”. For new Emails: Display massage box “welcome to safe journey”. And registration process done.	Expected results.	Input email that not registered in the data base.	Pass
TCCustomer004	Upload a Profile Image	Verify that the Open Image method correctly opens the file explorer dialog and loads the selected image into the picture box.	The Picture Box control is initialized and visible on the form.	Valid image file formats (PNG, BMP, JPG)	Click the upload Button. Opens File Explorer & Select image with valid format. Confirm the selection by clicking “OPEN”.	file explorer dialog opens. user can select an image file with a valid format After confirm display in picture box	Same as Expected results. Files explore open with grep the valid formats.	File explorers need to show only Valid image file formats (PNG, BMP, JPG)	Pass
TCCustomer005	Ensure all text boxes Are filled.	In customer dashboard there are 6 user input and one selected image. So in to verify that all are supplied before	The customer registration form is open. Exceptional handling initialized.	Customer Input data.	First do not input all the details and click the create button. Second: input all the user input and	For Unfilled form display an Exception error message “required parameters not supplied”. For Filled Form:	Same as expected Results.	The database connection is established.	Pass

		registering a customer.			click the create button.	Display “welcome to safe journey platform” And clear all user input and redirect to login page.		Need to input valid details to proceed.	

Testing for FrmAddBooking GUI (Customer)

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCustomer006	Validate User to input Taxi type first.	All the cars based on the taxi type, displays an error message when the user attempts to change the date without selecting a taxi type first	The application is running. The selected Type variable is null or empty.	Null or Empty value or Taxi type.	Change the value of the dateTimePicker1 control by selecting a new date.	An Argument Exception is thrown for user invalid number plate format.	displays a message box with the error message "Please select Taxi Type first."	Taxi type is necessary to proceed load cars, based on the selected date.	Pass
TCCar007	Load Cars Based on Taxi Type and Booking Date	Verify that the Load Car method correctly loads available car models based on the specified taxi type and booking date.	The booking Date parameter represents a valid booking date. The car Type parameter is set to a valid taxi type. The database contains Cars records.	Load car date base Data that is available regarding booking date.	Input Valid taxi type & Booking date. Click Combo box arrow function.	The Load Car method successfully retrieves and load to combo Vehicle with available car	Same as Expected. Avoid display vehicles that have booking on that day.	Customer can not select cars that already have a booking on that day.	Pass
TCDriver002	Load Drivers Based on Booking Date	Verify that the Load Driver method correctly loads available Drivers based on the booking date.	The booking Date parameter represents a valid booking date. The database contains Cars records.	Load the driver form the database.	Input Booking date. Click Combo box arrow function.	The Load Driver method successfully retrieves and load to combo Driver with available drivers	Same as Expected. Avoid display drivers that already have booking on that day.	Customer cannot select drivers that already have a booking on that day.	Pass

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCustomer007	View Driver Details	Verify that the btnViewDriver Button correctly displays the driver details in a new form when a driver is selected from the combo box.	The combo Driver combo box is populated with driver names. The database contains driver records. The FrmAddDriver form is properly set up to display driver details.	The driver Details regarding the name.	Select a driver from the combo Driver combo box. Click the "View Driver" button.	If a driver is selected, the driver details should be displayed in the FrmAddDriver form. Any exceptions should be caught, and an error message should be displayed.	Same as Expected results. When no driver is selected, an error message shown.	To display diver details. Need to select a driver first. The connection of the database must be true	Pass
TCCustomer008	View Car Details	Verify that the btnViewCar Button correctly displays the Car details in a new form when a Car is selected from the combo box.	The database contains driver records. The FrmAddCar form is properly set up to display Car details	The Car Details regarding the name.	Select a Car from the combo Car combo box. Click the "View Car " button.	If a Car is selected, the Car details should be displayed in the FrmAddCar form. Any exceptions should be caught, and an error message should be displayed.	Same as Expected results. When no Car is selected, an error message shown.	To display Car details. Need to select a Car first. The connection of the database must be true	Pass

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCustomer009	Calculate Price Based on Distance and Car Name	Verify that the Distance method correctly calculates the total price regarding to the distance entered and the selected car's price per kilometer.	The Distance Textbox is available for input. The database contains Car Price records.	The Car Pricing details regarding the car name.	Select a car from the comboVehical combo box. Enter a valid numeric distance in the Distance TextBox.	A valid car is selected, and a numeric distance is entered, the total price is correctly calculated and displayed in the CalPrice TextBox. For non-numeric Values display “Invalid distance”	Same as Expected results.	The CalPrice TextBox displays the calculated price or an error message Regarding on the input.	Pass
TCOrder001	Forward Booking Details	Verify that the ForwardBooking Details method correctly validates inputs and forwards booking details to the FrmPayments form	_loggedIn Customer object is initialized and available. All the Textboxes are visible.	All the User Input Data. Validate if There is And empty Value.	Input all the text boxes that are required for add booking. Click the "Submit " button.	If all values are filled, then redirect to payment form with values. Else, there is, and empty textbox indicate Error massage with text box name which is empty.	Same as Expected results. All the input forward to payment form.	All the text boxes need to be filled with valid inputs to redirect to the payment form.	Pass

Testing for FrmPayments GUI

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/ Fail)
TCOrder002	Retrieve Booking details and display on text boxes.	After the user adds the booking details in booking form then it forward to this form. There is a constructor to receive those data.	All the booking details are forwarded to this form.	Forwarded booking data.	Add a booking in the booking form. Then Click the submit button.	All the user input data Need to be displayed in textboxes. If there is a empty field in booking form display a error massage.	Same as Expected results.	All the Booking text boxes need to be filled with valid data to proceed.	Pass
TCOrder003	Set Booking Details with Car Information	Verify that the SetBookingDetails method correctly assigns the provided booking details with addition details regarding to customer details stored in Database.	_loggedInCustomer object is initialized and available with a valid CustomerName. The DbCars class has a method GetCarDetailsByName.	Customer Details form The database.	Add a booking in the booking form. Then Click the submit button.		Same as Expected results. All the input forward to payment form.	All the text boxes need to be filled with valid inputs to redirect to the payment form.	Pass
TCOrder004	Upload a Payment Receipt.	Verify that the Open Image method correctly opens the file explorer dialog and loads the selected image into the picture box.	The Picture Box control is initialized and visible on the form.	Valid image file formats (PNG, BMP, JPG)	Click the upload Button. Opens File Explorer & Select image with valid format. Confirm the selection by clicking “OPEN”.	file explorer dialog opens. user can select an image file with a valid format After confirm display in picture box	Same as Expected results. Files explore open with grep the valid formats.	File explorers need to show only Valid image file formats (PNG, BMP, JPG)	Pass

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/ Fail)
TCOther001 (Payment class)	View Image Popup	Verify that clicking the pic Picture Box displays the image in a popup	The form containing the pic Picture Box is open. The pic Picture Box has an image assigned.	Valid Payment Receipt image	Initialize Picture Box. Click Picture Box. Verify Image Popup.	The pic Picture Box contains a valid image. The pic_Click event handler is triggered. The ImagePopupForm is displayed.	Same as Expected results.	The ImagePopupForm is displayed with the image.	Pass
TCOther002 (Payment class)	Verify New booking confirmation.	Verify that the ComfirmBTN_Click method correctly creates a new booking using the provided customer, cab, driver, and order details and adds it to the database.	The FrmPayments form is open and all necessary text fields and controls are initialized. The _db object is properly initialized and connected to the database.	All data that need to add New Booking.	Add a booking in the booking form. Then Click the submit button. Then redirected to payment check details and Click confirmed button	The ComfirmBTN Click method is executed without errors. And display "New Booking is added!". If any value that null will display Error massage.	Same as Expected results. New Booking is added in database.	The booking is successfully added to the database. All text fields on the form are cleared.	Pass

Testing for Adding Cars GUI

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCar001	Validate Number plate format	Verify that the Number Plate property validation and detects invalid formats.	The `cab` class instance created.	Invalid number plate format: “ÄBC-123”	Set the number plate property to each invalid format. Verify that an Argument Exception is thrown.	An Argument Exception is thrown for user invalid number plate format.	An error massage box showing invalid number plate.	The Number Plate property should not hold any invalid value. Either null value.	Pass
TCCar002	Validate Price Format	Verify that the Price property validation correctly detects invalid formats.	The Cab class instance created.	Invalid price formats like "ABC", "-12.44", "$150"	Attempt to set the Price property and test each invalid format.	An Argument Exception is thrown for each invalid price format.	An error massage box showing invalid Price per Km.	The Price per KM property should not hold any invalid value. Either null value.	Pass
TCCar003	fill all user input and check	Filling all the required fields in the Adding vehicles Gui and click Add Car button	The system is initialized and ready to process input. The Cab class instance created.	All user inputs are getting.	Filling the required input fields Clicking “Add Car” button	A successful massage.	A Success massage box shows [“Car is added”] Added in to data base	The user input all are filled then it successfully stored in data base.	Pass
TCCar004	Fill user inputs with left one or two and click add car button.	Filling the required fields (not all) in the Adding vehicles Gui and click Add Car button	The system is initialized and ready to process input. The Cab class instance created.	For Null user inputs like “_”	Filling the required input fields with left a one. Clicking “Add Car” button	A Error massage.	Indicate an Exceptional error massage on a massage box.	The user input need to be all filled when add a car.	Pass

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCCar005	Open Image File Dialog	Verify that the Open Image method correctly opens the file explorer dialog and loads the selected image into the picture box.	The Picture Box control is initialized and visible on the form.	Valid image file formats (PNG, BMP, JPG)	Click the upload Button. Opens File Explorer & Select image with valid format. Confirm the selection by clicking “OPEN”.	file explorer dialog opens. user can select an image file with a valid format After confirm display in picture box	Same as Expected results. Files explore open with grep the valid formats.	File explorers need to show only Valid image file formats (PNG, BMP, JPG)	Pass

Testing for Adding Drivers GUI

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCDriver003	fill all user input and check	Filling all the required fields in the Adding Driver Gui and click Add Driver button	The system is initialized and ready to process input. The Driver class instance created.	All user inputs are getting.	Filling the required input fields Clicking “Add Driver” button	A successful massage.	A Success massage box shows [“Driver is added”] Added in to data base	IF user input all are filled then it successfully stored in data base.	Pass
TCDriver004	Fill user inputs with left one or two and click add Driver button.	Filling the required fields (not all) in the Adding Driver Gui and click Add Driver button	The system is initialized and ready to process input. The Driver class instance created.	For Null user inputs like “_”	Filling the required input fields with left a one. Clicking “Add Car” button	An Error massage.	Indicate an Exceptional error massage on a massage box.	The user input needs to be all filled when add a Driver.	Pass
TCDriver005	Open Image File Dialog	Verify that the Open Image method correctly opens the file explorer dialog and loads the selected image into the picture box.	The Picture Box control is initialized and visible on the form.	Valid image file formats (PNG, BMP, JPG)	Click the upload Button. Opens File Explorer & Select image with valid format. Confirm the selection by clicking “OPEN”.	file explorer dialog opens. user can select an image file with a valid format After confirm display in picture box	Same as Expected results. Files explore open with grep the valid formats.	File explorers need to show only Valid image file formats (PNG, BMP, JPG)	Pass

Testing for CarManegment GUI

Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCAdmin002	Load Cars on Form Load	Verify that the list of cars is loaded correctly when the CarManegment form is loaded.	The database is populated with car data.	Car data base data.	Open the CarManegment form. Observe the DgvCars DataGridView.	The DgvCars DataGridView should display all the cars from the database.	Same as Expected.	Need to store cars in database first.	Pass
TCAdmin003	Add a New Car	Verify that a new car can be added using the Add car button.	The car management form is open.	Car data base data.	Click the add car button. Then add a new car in FrmAddcar from. Click the Create_btn. Then check GgvCars in car management form.	A new car Should Appear in the dgvCars. The car details Should match with inputs.	Same as Expected.	All user inputs need to be filled to add a new car.	Pass
TCAdmin004	Edit & Delete Car Details	Verify that car details can be edited and delete using the ColEdit button and ColDelete button in the DgvCars DataGridView.	The car management form is open. At least one car needs to be in list.	Car data base data.	First click the Col edit, then change data in FrmAddCar and Click Update button. Check dgv cars its live update. Secondly, click Col Delete and check its deleting form the dgv cars.	The selected car needs to update successfully and appear on dgv cars. The selected car needs to delete and appear on dgv cars.	Same as Expected.	All user inputs need to be filled to Update a car.	Pass

Testing for DriverManegment GUI

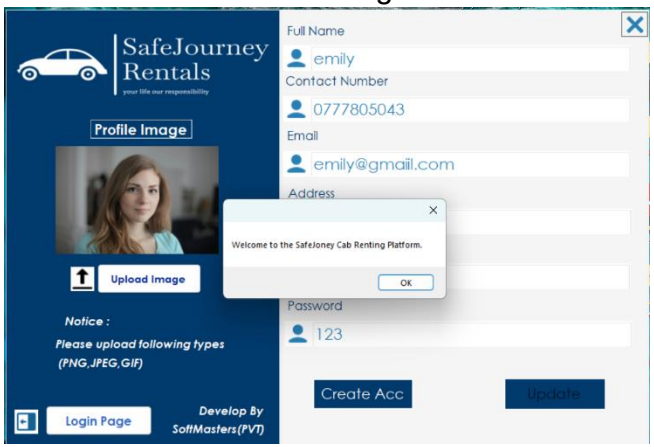
Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCAdmin005	Load Drivers on Form Load	Verify that the list of Drivers is loaded correctly when the DriverManegment form is loaded.	The database is populated with Driver data.	Driver data base data.	Open the DriverManegment form. Observe the DgvDriver DataGridView.	The DgvDriver DataGridView should display all the Driver from the database.	Same as Expected.	Need to store Driver in database first.	Pass
TCAdmin006	Add a New Driver	Verify that a new Driver can be added using the Add Driver button.	The Driver management form is open.	Driver data base data.	Click the add Driver button. Then add a new Driver in FrmAddDriver from. Click the Create_btn. Then check DgvDriver in Driver management form.	A New Driver Should Appear in the dgvDriver. The details Should match with inputs.	Same as Expected.	All user inputs need to be filled to add a new Driver .	Pass
TCAdmin007	Edit & Delete Driver Details	Verify that Driver details can be edited and delete using the ColEdit button and ColDelete button in the dgvDriver DataGridView.	The Driver management form is open. At least one Driver needs to be in list.	Driver data base data.	First click the Col edit, then change data in FrmAddDriver and Click Update button. Check dgvDriver its live update. Secondly, click Col Delete and check its deleting form the dgvDriver.	The selected Driver needs to update successfully and appear on dgvDriver . The selected car needs to delete and appear on dgvDriver.	Same as Expected.	All user inputs need to be filled to Update a Driver .	Pass

Testing for BookingManegment GUI

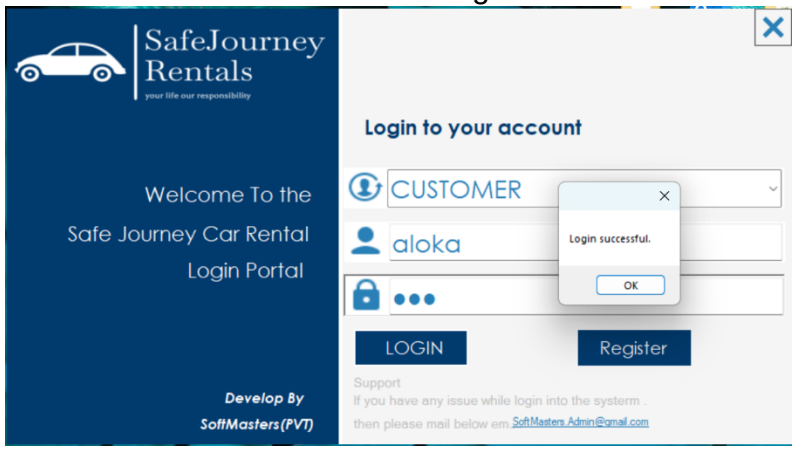
Test Case ID	Test Case	Test Description	Preconditions	Test Data	Test Steps	Expected Result	Actual Result	Post-conditions	Status (Pass/Fail)
TCAdmin008	Load Booking on Form Load	Verify that the list of Booking is loaded correctly when the Booking Manegment form is loaded.	The database is populated with Booking data.	Bookings data base data.	Open the Booking Manegment form. Observe the DgvBooking DataGrid View.	The DgvBooking DataGridView should display all the Bookings from the database.	Same as Expected.	Need to store Bookings in database first.	Pass

05.Screen Shots of Main Test cases

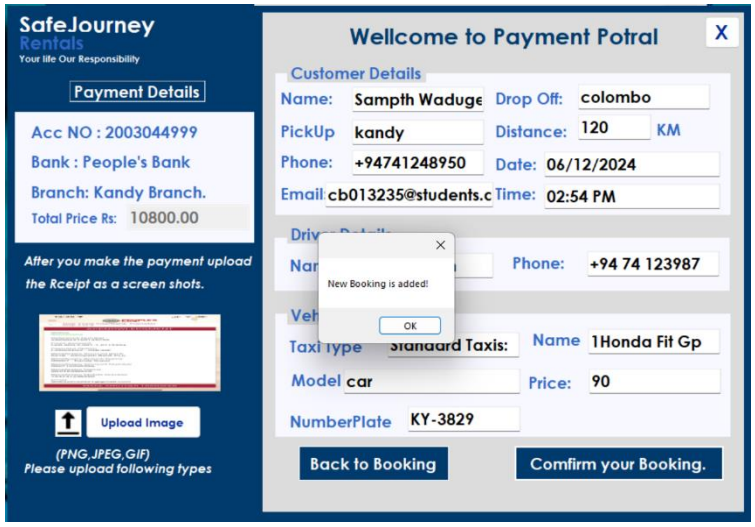
Customer Register



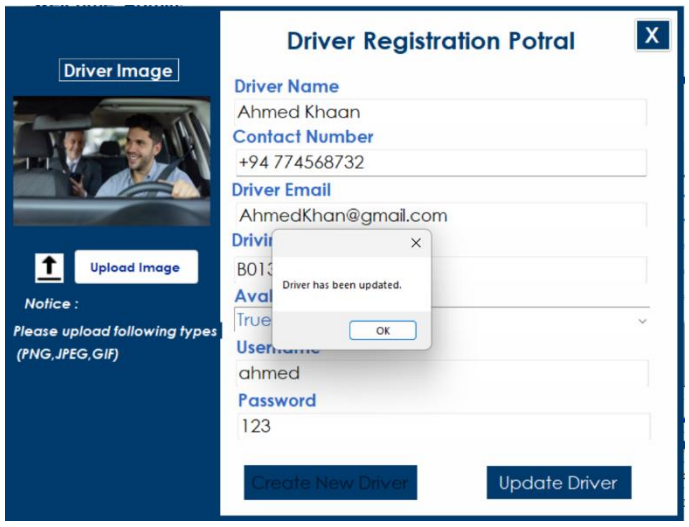
Customer Login



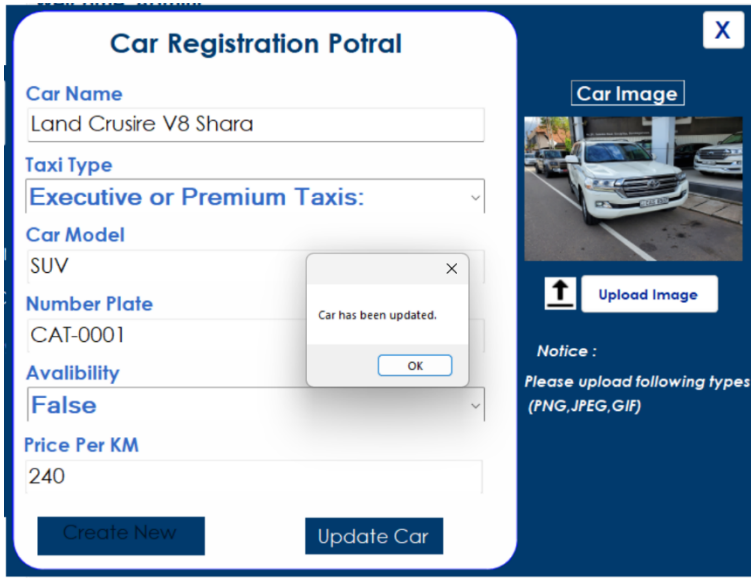
Add Booking (Customer)



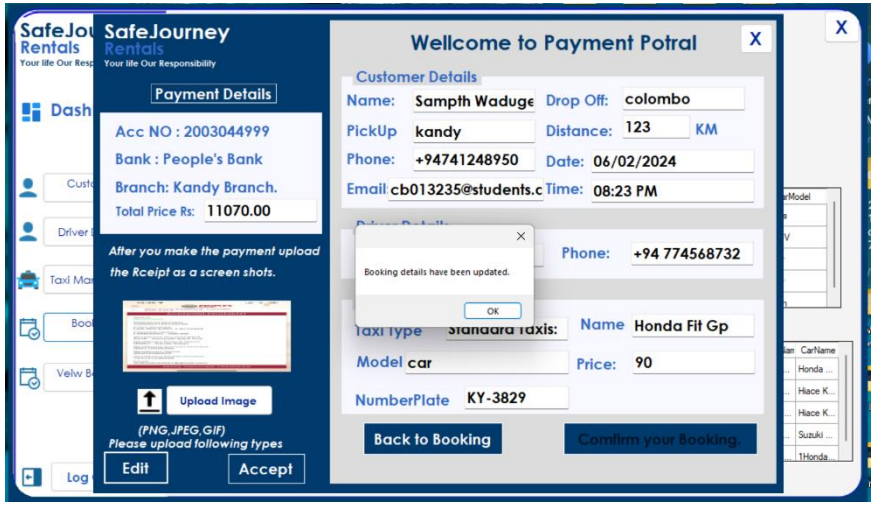
Edit Driver Details



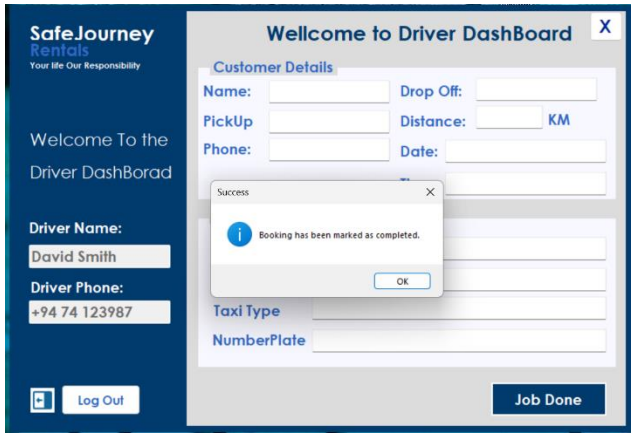
Update Car (Car).



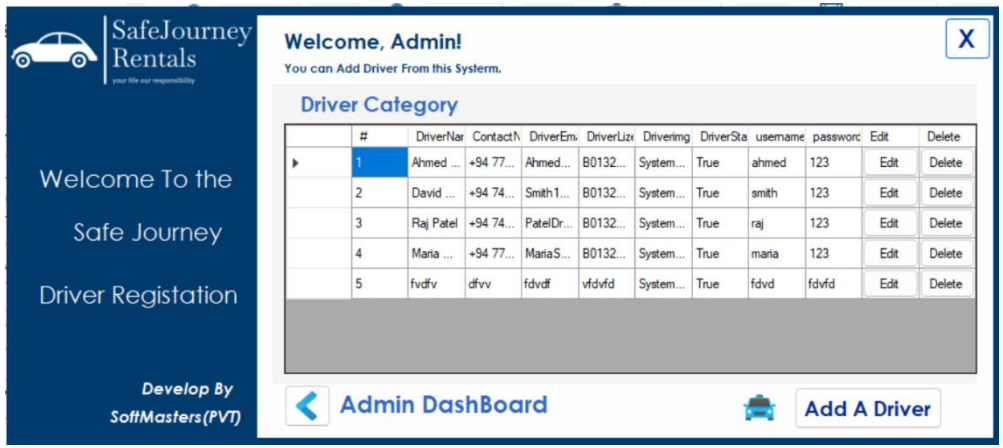
Update Booking Details.



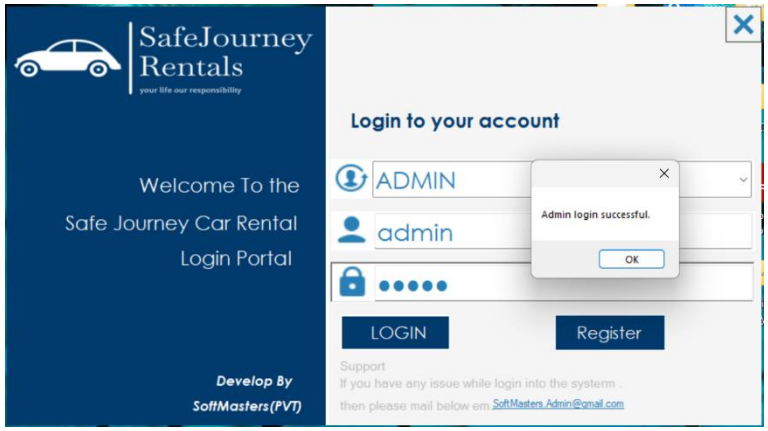
Order Confirmation (Driver)



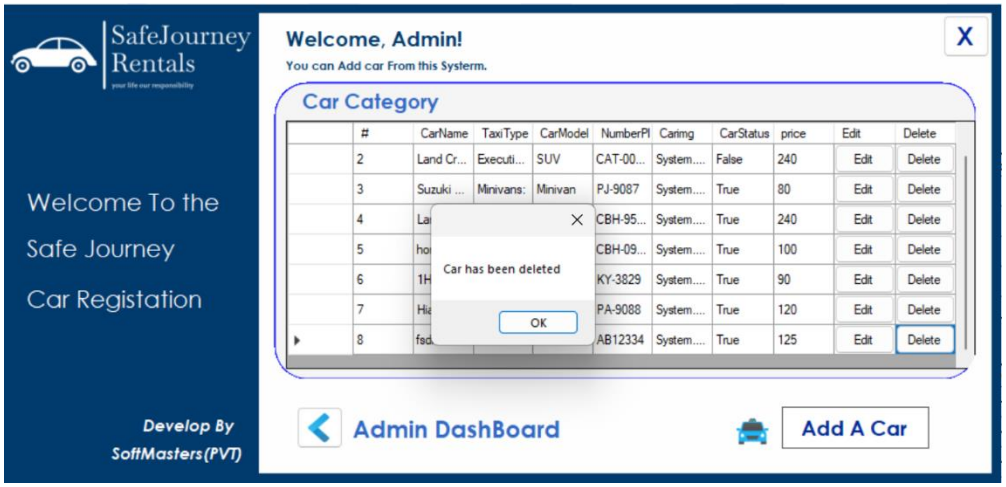
Load Drivers (Manage Drivers)



Login as Admin (Admin).

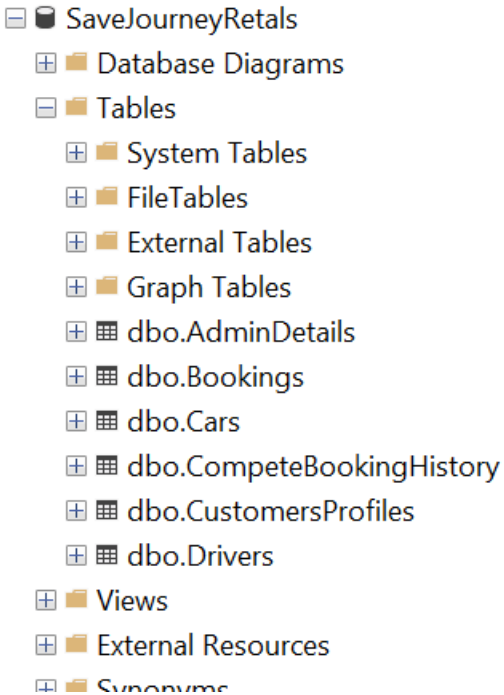


Delete Car details from the database. (Car)

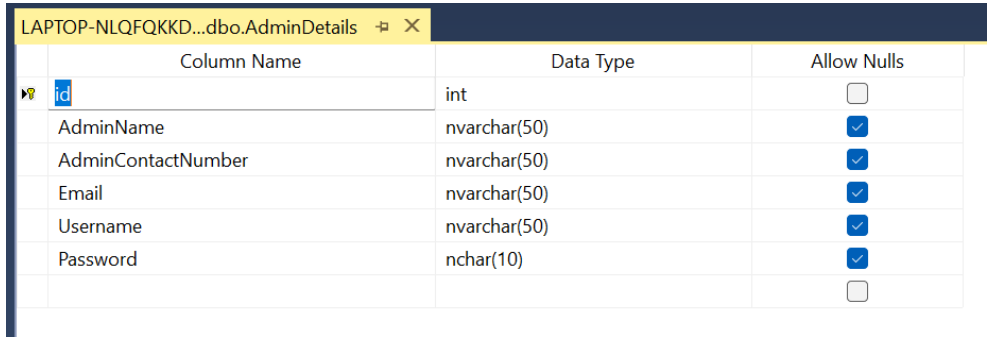


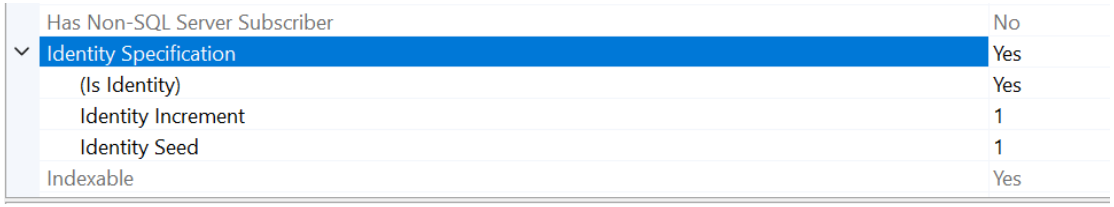
06.Images Of SQL Database

Data base tables.

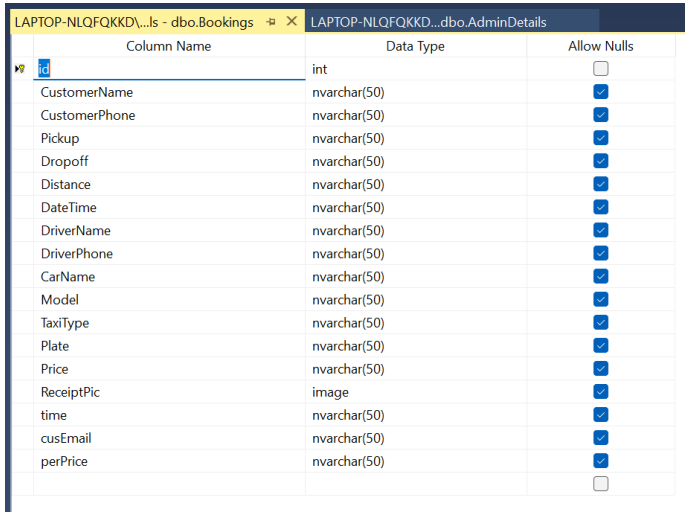


Admin Details.





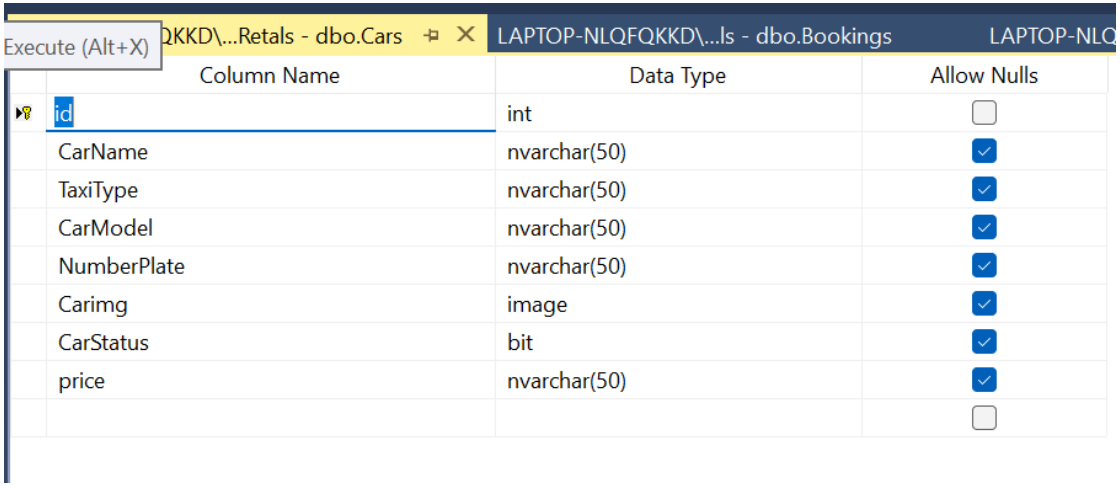
Bookings table.



LAPTOP-NLQFQKKD\...ls - dbo.Bookings | LAPTOP-NLQFQKKD...dbo.AdminDetails

Column Name	Data Type	Allow Nulls
id	int	☐
CustomerName	nvarchar(50)	☑
CustomerPhone	nvarchar(50)	☑
Pickup	nvarchar(50)	☑
Dropoff	nvarchar(50)	☑
Distance	nvarchar(50)	☑
DateTime	nvarchar(50)	☑
DriverName	nvarchar(50)	☑
DriverPhone	nvarchar(50)	☑
CarName	nvarchar(50)	☑
Model	nvarchar(50)	☑
TaxiType	nvarchar(50)	☑
Plate	nvarchar(50)	☑
Price	nvarchar(50)	☑
ReceiptPic	image	☑
time	nvarchar(50)	☑
cusEmail	nvarchar(50)	☑
perPrice	nvarchar(50)	☑
		☐

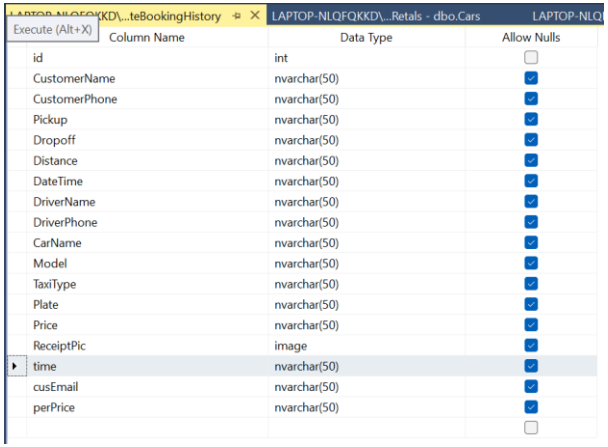
Cars table.



Execute (Alt+X) | QKKD\...Retals - dbo.Cars | LAPTOP-NLQFQKKD\...ls - dbo.Bookings | LAPTOP-NLQ

Column Name	Data Type	Allow Nulls
id	int	☐
CarName	nvarchar(50)	☑
TaxiType	nvarchar(50)	☑
CarModel	nvarchar(50)	☑
NumberPlate	nvarchar(50)	☑
Carimg	image	☑
CarStatus	bit	☑
price	nvarchar(50)	☑
		☐

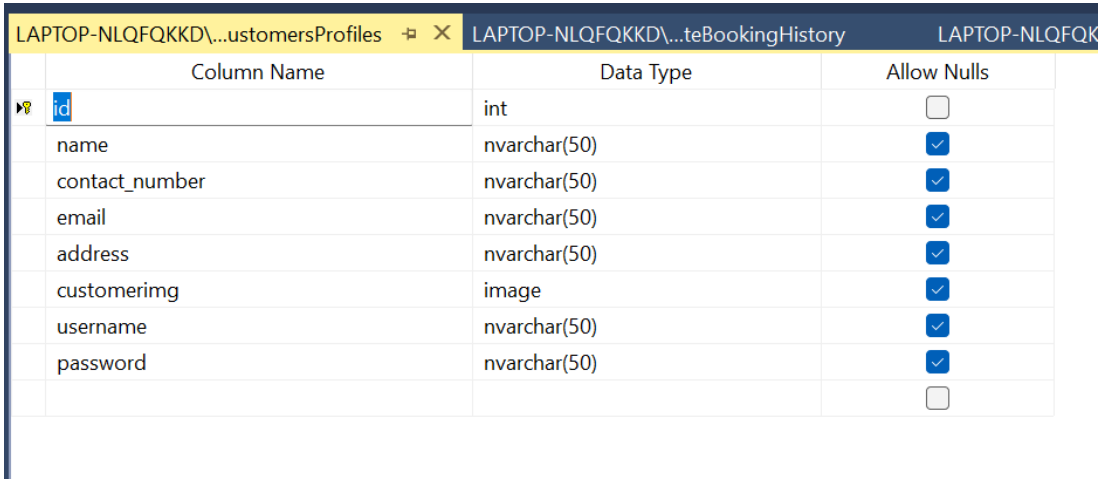
Complete Booking History table.



Execute (Alt+X) | ...teBookingHistory | LAPTOP-NLQFQKKD\...Retals - dbo.Cars | LAPTOP-NLQ

Column Name	Data Type	Allow Nulls
id	int	☐
CustomerName	nvarchar(50)	☑
CustomerPhone	nvarchar(50)	☑
Pickup	nvarchar(50)	☑
Dropoff	nvarchar(50)	☑
Distance	nvarchar(50)	☑
DateTime	nvarchar(50)	☑
DriverName	nvarchar(50)	☑
DriverPhone	nvarchar(50)	☑
CarName	nvarchar(50)	☑
Model	nvarchar(50)	☑
TaxiType	nvarchar(50)	☑
Plate	nvarchar(50)	☑
Price	nvarchar(50)	☑
ReceiptPic	image	☑
time	nvarchar(50)	☑
cusEmail	nvarchar(50)	☑
perPrice	nvarchar(50)	☑
		☐

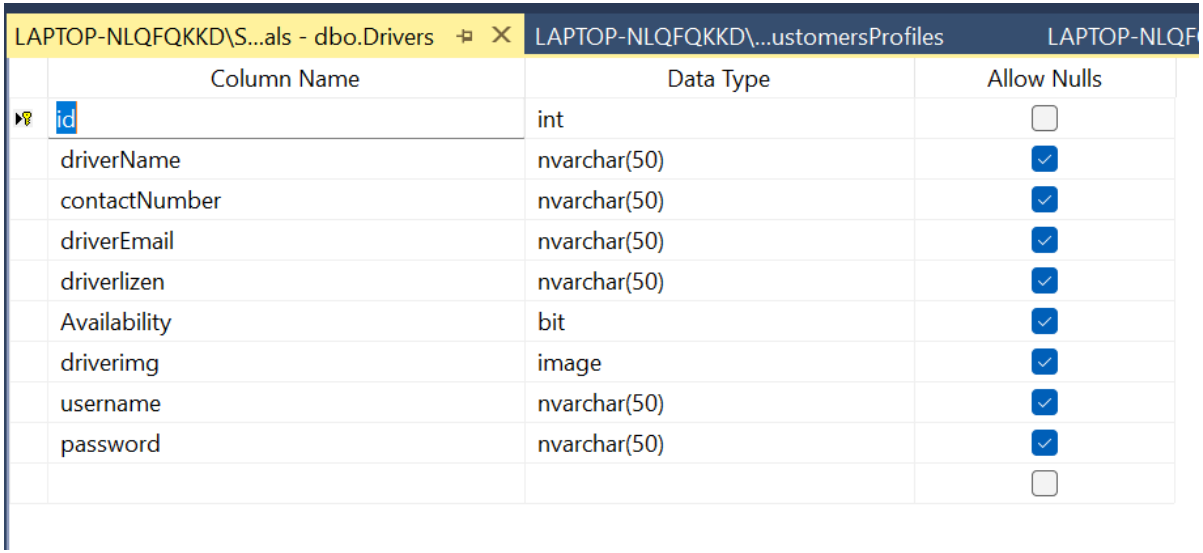
Customer Profiles table.



LAPTOP-NLQFQKKD\...ustomersProfiles | LAPTOP-NLQFQKKD\...teBookingHistory | LAPTOP-NLQFQK

Column Name	Data Type	Allow Nulls
id	int	☐
name	nvarchar(50)	☑
contact_number	nvarchar(50)	☑
email	nvarchar(50)	☑
address	nvarchar(50)	☑
customerimg	image	☑
username	nvarchar(50)	☑
password	nvarchar(50)	☑
		☐

Drivers table



Column Name	Data Type	Allow Nulls
id	int	☐
driverName	nvarchar(50)	☑
contactNumber	nvarchar(50)	☑
driverEmail	nvarchar(50)	☑
driverlizen	nvarchar(50)	☑
Availability	bit	☑
driverimg	image	☑
username	nvarchar(50)	☑
password	nvarchar(50)	☑
		☐

07. Conclusion

First of I have thank for our lecture MR. Pradeep sir who gave up successful guidance for provide a successful outcome.

In the future I like to upgrade my Safe Journey Cab Service with using ASP.NET, Guna and Nuget packages to make an industrial standard project.

In Safe journey cab service, I have implemented sent an email to customer for as a booking receipt. Then I implemented a method when customer selecting drivers and cars only, they can see available ones for based on his pickup date.

All so drivers can see the latest booking. Once he makes the booking job done then its available to the customer.