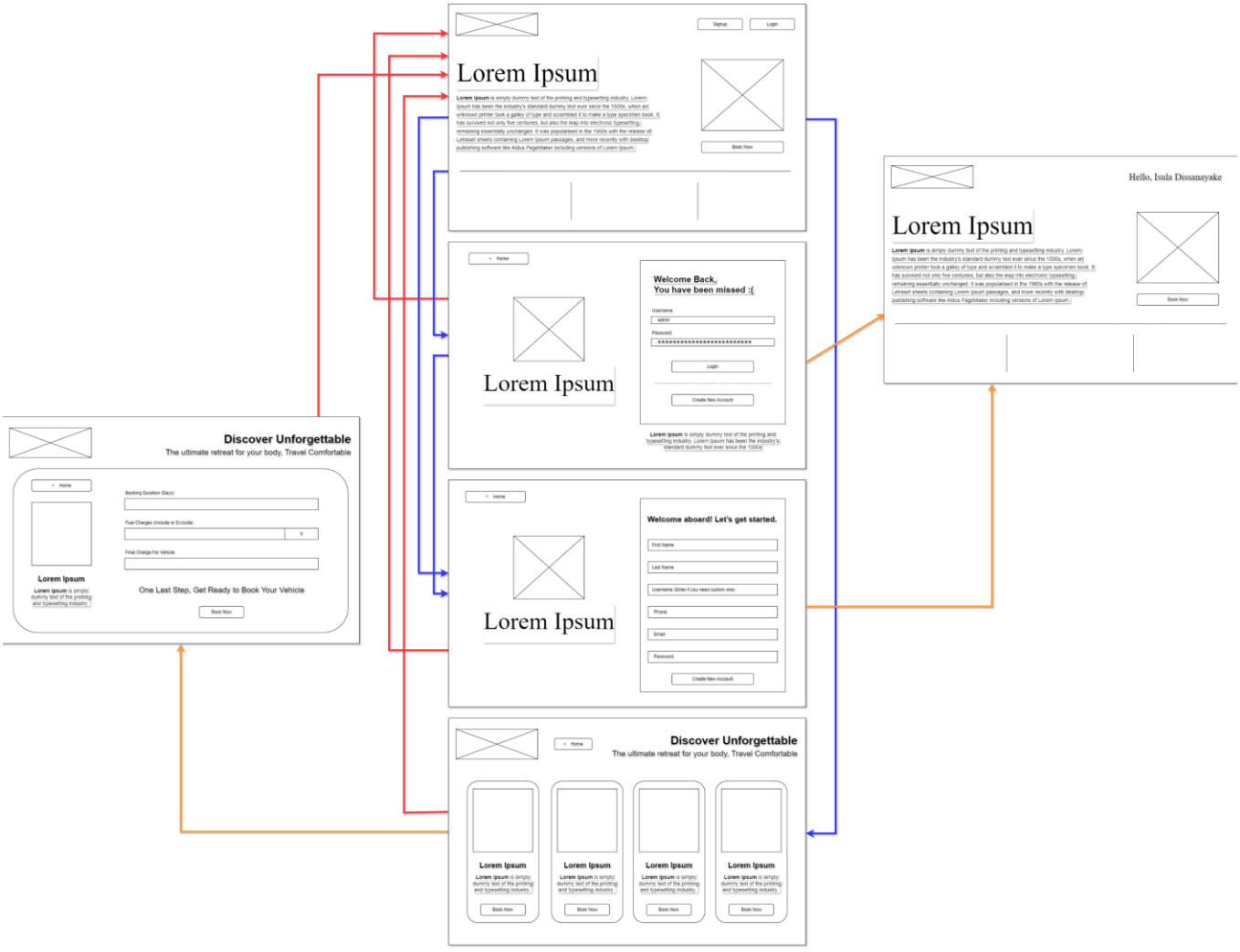
The Project - Vehicle Rental System

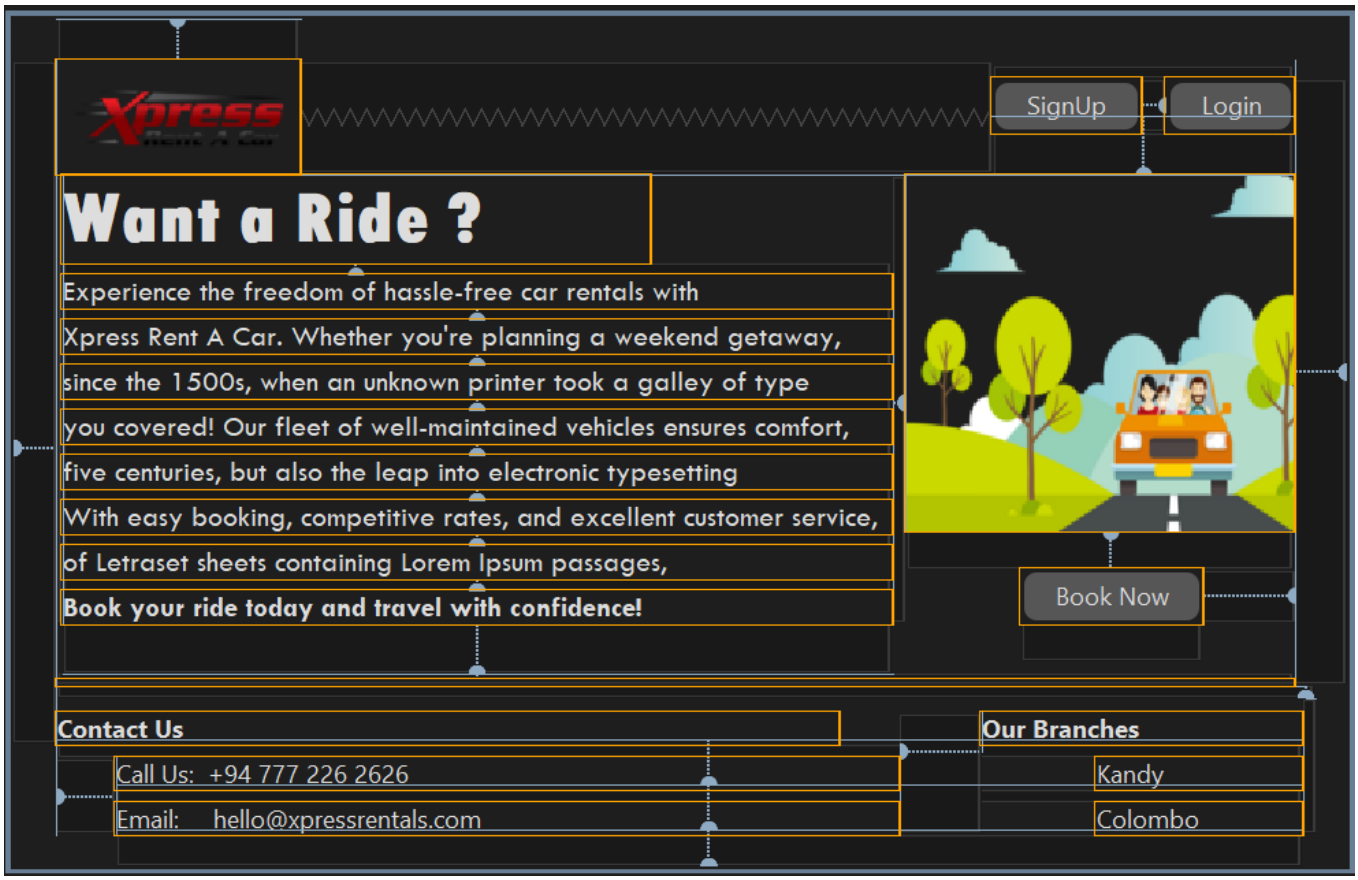
This project is based on vehicle rental system that allows user to select a vehicle (Bike, Car, Van or Jeep) & calculates the final charge based on the selected vehicle type and inserted duration.

Wireframe of The Project



Home page is accessible to every users but they can't hire a vehicle until the login to the system. If the user doesn't have a account, He/ She can sign up to the system through sign up page. Once the user login to the system, they can select a vehicle & hire that vehicle by inserting the duration that they would like to use that particular vehicle.

Home Page (Before login)



This page displays the text about the company, contact information etc. Allows users to navigate to other pages like login/ Sign up or vehicle selection.

JFrame: The main window of the application.

JButton: Used to create buttons such as “Sign up”, “Login”, “Book now”.

JLable: Used to display text & images.

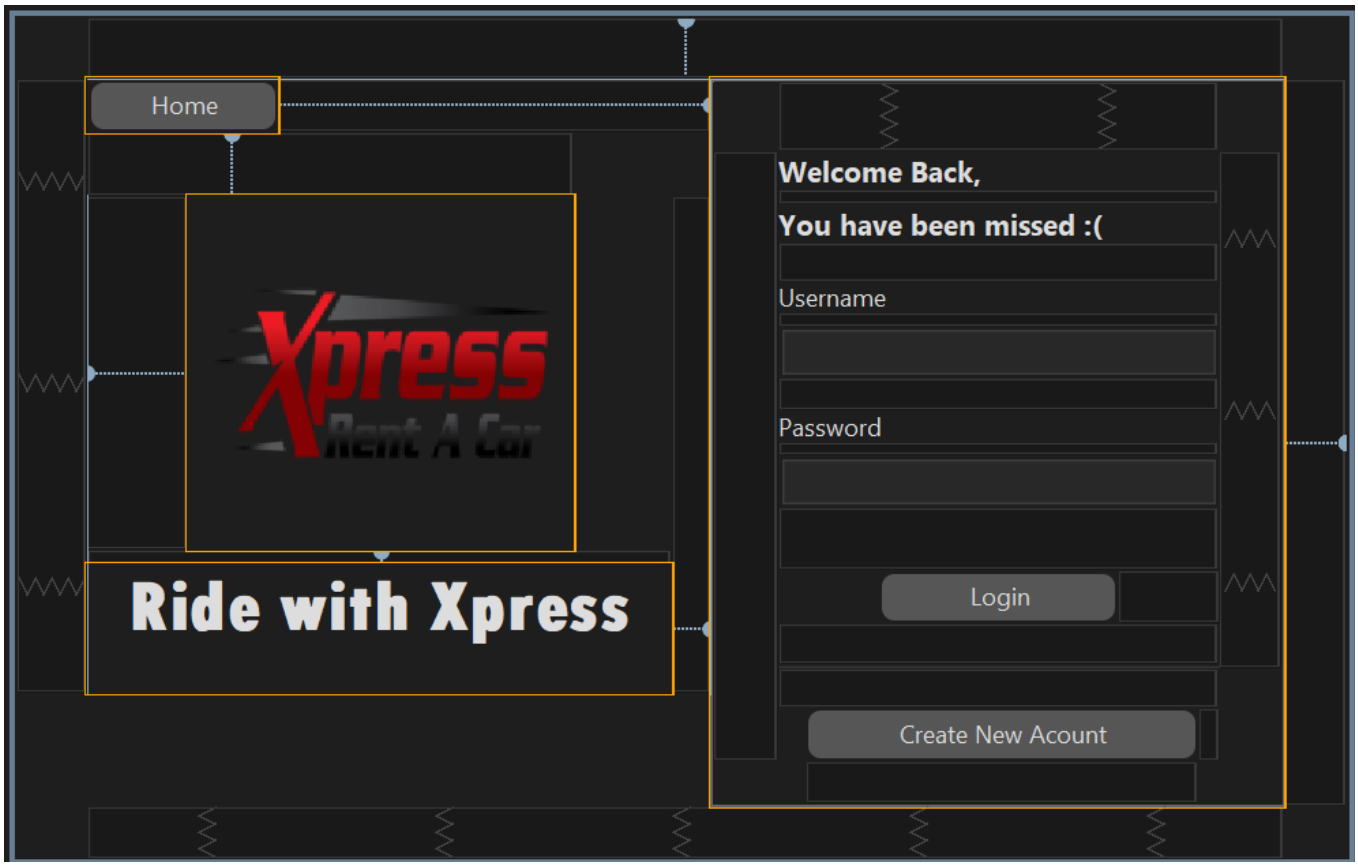
JPanel: Used to group components together.

The Code

```
private void btnSignUpActionPerformed(java.awt.event.ActionEvent
evt) {
        Signup obj01 = new Signup();
        obj01.setVisible(true);
        dispose();
    }
```

```
    private void btnLoginActionPerformed(java.awt.event.ActionEvent
evt) {
        Login obj01 = new Login();
        obj01.setVisible(true);
        dispose();
    }

    private void
btnBookNowActionPerformed(java.awt.event.ActionEvent evt) {
        VehicleSelection obj01 = new VehicleSelection();
        obj01.setVisible(true);
        dispose();
    }
```

Login Page



Manages user login/logout functionality. Allows user to login to the system. If user doesn't have an account, it links to the "Sign up" page through the "Create new account" Button. User can go back to the home page through "Home".

JFrame: The login window of system

JButton: Used to create buttons such as “Home”, “Login”, “Create new account”.

JLable: Used to display text & images.

JTextField: Used for user inputs such as “Username” & “Password”.

The Code

```
private void
btnCreateNewAccountActionPerformed(java.awt.event.ActionEvent evt)
{
        Signup obj01 = new Signup();
        obj01.setVisible(true);
        dispose();
    }

    private void btnLoginActionPerformed(java.awt.event.ActionEvent
evt) {
        String un= txtUsername.getText();
        String pw=txtPassword.getText();

        LoginManager.getInstance().login(un, pw);

        if (LoginManager.getInstance().isLoggedIn()) {
            HomePageLoggedIn obj01 = new HomePageLoggedIn();
            obj01.setVisible(true);
            dispose();
        } else {
            JOptionPane.showMessageDialog(null, "Username or
Password Invalid!", "Input Error", JOptionPane.ERROR_MESSAGE);
            funClear();
        }
    }
    public void funClear(){
```

```
        txtUsername.setText("");
        txtPassword.setText("");
    }
    private void
txtPasswordActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }


    private void
txtUsernameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }


    private void btnHomeActionPerformed(java.awt.event.ActionEvent
evt) {
        HomePage obj01 = new HomePage();
        obj01.setVisible(true);
        dispose();
    }
```

class LoginManager

Manages user login/logout functionality. Checks if the user is logged in. Add new users to the system & saves them to a file.

HashMap: Used to store details.

JOptionPane: For displaying messages like error or success messages.

Singleton Pattern: Ensures that only one instance of LoginManager exists throughout the application.

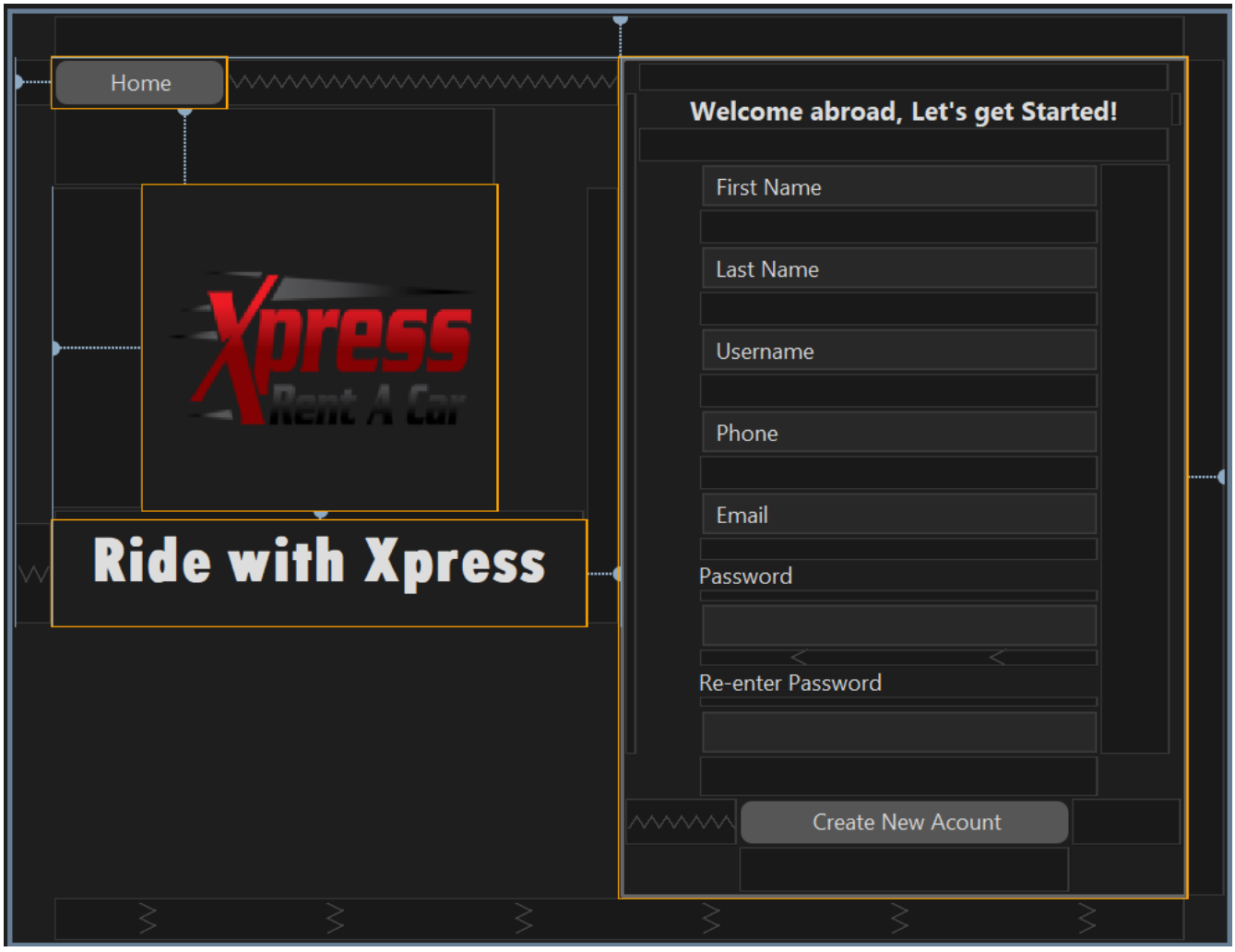
Encapsulation: Private fields with public methods (getters) for accessing & modifying data safely.

```java
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import javax.swing.JOptionPane;


public class LoginManager {
    private static LoginManager instance;
    boolean isLoggedIn = false;
    private Map<String, User> users;

    private LoginManager() {
        users = new HashMap<>();
    }

    public static synchronized LoginManager getInstance() {
        if (instance == null) {
            instance = new LoginManager();
        }
        return instance;
    }

    public void addUser(User user) {
        users.put(user.getUsername(), user);

        //Saving new user
        String filePath = "User Details Sign Up.txt";
        try (FileWriter writer = new FileWriter(filePath, true)) {
```

```
            writer.append("Username: " + user.getUsername() +
"\n");
            writer.append("Password: " + user.getPassword() +
"\n");
           writer.append("Re-Check Password: " +
user.getPasswordRecheck() + "\n");
            writer.append("First Name: " + user.getFirstName() +
"\n");
            writer.append("Last Name: " + user.getLastName() +
"\n");
            writer.append("Phone: " + user.getPhone() + "\n");
            writer.append("Email: " + user.getEmail() + "\n\n");


            JOptionPane.showMessageDialog(null, "User saved
successfully!");
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error saving User:
" + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    public boolean isUserExists(String username) {
        return users.containsKey(username);
    }
    public boolean login(String username, String password) {
        if ("admin".equals(username) && "1234".equals(password)) {
            isLoggedIn = true;
        }
        if (isUserExists(username)) {
            User user = users.get(username);
            return user.getPassword().equals(password);
        }
        if (users.containsKey(username)) {
```

```
            User user = users.get(username);
            if (user.getPassword().equals(password)) {
                isLoggedIn = true;
                return true;
            } else {
                return false;
            }
        }
        return false;
    }
    public boolean isLoggedIn() {
        return isLoggedIn;
    }
    public void logout() {
        isLoggedIn = false;
    }
}
```

Sign up page



Allows users to create new account/ Sign up to the system who don't have an account. It links to the home page through "Home" button.

JFrame: The Sign in window of system

JButton: Used to create buttons such as "Home", "Create new account".

JLable: Used to display text & images.

JTextField: Used for user inputs such as "First name", "Last name", "Username" & "Password" etc.

<u>The Code</u>

```
    public Signup() {
        initComponents();    //To initialize GUI components, like
buttons
        btnCreateNewAccount.setEnabled(false);
    }
```

```
private void
btnCreateNewAccountActionPerformed(java.awt.event.ActionEvent evt)
{
        String fName= txtFirstName.getText();
        String lName=txtLastName.getText();
        String uName=txtUsername.getText();
        String pNumber= txtPhone.getText();
        String email=txtEmail.getText();
        String pw=txtPassword.getText();
        String pwr=txtPasswordReCheck.getText();

        //cheacking inputs
        if (fName.isEmpty() || lName.isEmpty() || uName.isEmpty()
|| pNumber.isEmpty() || email.isEmpty() || pw.isEmpty() ||
pwr.isEmpty()) {
            JOptionPane.showMessageDialog(this, "All fields are
required!", "Input Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        if ("admin".equals(uName)) {
            JOptionPane.showMessageDialog(this, "Admin account
already exists.", "Input Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
      if (!pNumber.matches("\\d{10}")) {
                    throw new IllegalArgumentException("Invalid
phone number. Please enter a 10-digit number.");
                }
      if (!email.contains("@") || !email.contains(".")) {
                    throw new IllegalArgumentException("Invalid
email address. Please enter a valid email.");
                }
```

```
     if (!pw.equals(pwr)){
throw new IllegalArgumentException("passwords are not matching,
please re-check.");}


        //Sending instance to login manager
        LoginManager.getInstance().login(uName, pw);


        //adding new user
        User newUser = new User(uName, pw, pwr, fName, lName,
pNumber, email);
        LoginManager.getInstance().addUser(newUser);


        //Seting login manager after signup
        LoginManager.getInstance().isLoggedIn = true;


        throw new IllegalArgumentException ("Account created
successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);
        HomePageLoggedIn obj01 = new HomePageLoggedIn();
        obj01.setVisible(true);
        dispose();
    }


    private void
btnHomePageActionPerformed(java.awt.event.ActionEvent evt) {
        HomePage obj01 = new HomePage();
        obj01.setVisible(true);
        dispose();
    }


    private void
txtFirstNameActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add your handling code here:
    }

    private void
txtUsernameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void
txtLastNameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txtEmailActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void
txtUsername4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void txtPhoneActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void
txtPasswordActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
    private void txtFirstNameMouseClicked(java.awt.event.MouseEvent
evt) {
        if (!_firstNameCleared) {
            txtFirstName.setText("");
            _firstNameCleared = true;
        }
    }

    private void txtLastNameMouseClicked(java.awt.event.MouseEvent
evt) {
        if (!_lastNameCleared) {
            txtLastName.setText("");
            _lastNameCleared = true;
        }
    }

    private void txtUsernameMouseClicked(java.awt.event.MouseEvent
evt) {
        if (!_usernameCleared) {
            txtUsername.setText("");
            _usernameCleared = true;
        }
    }

    private void txtPhoneMouseClicked(java.awt.event.MouseEvent
evt) {
        if (!_phoneCleared) {
            txtPhone.setText("");
            _phoneCleared = true;
        }
```

```
    }

    private void txtEmailMouseClicked(java.awt.event.MouseEvent
evt) {
        if (!_emailCleared) {
            txtEmail.setText("");
            _emailCleared = true;
        }
    }
}
```

class User

Represents a user with attributes like “Username”, “Password” etc. It provides methods to get these attributes & save them to a file.

JOptionPane: For display messages like error/Success.

Encapsulation is used to Private fields with public getter & setter methods.

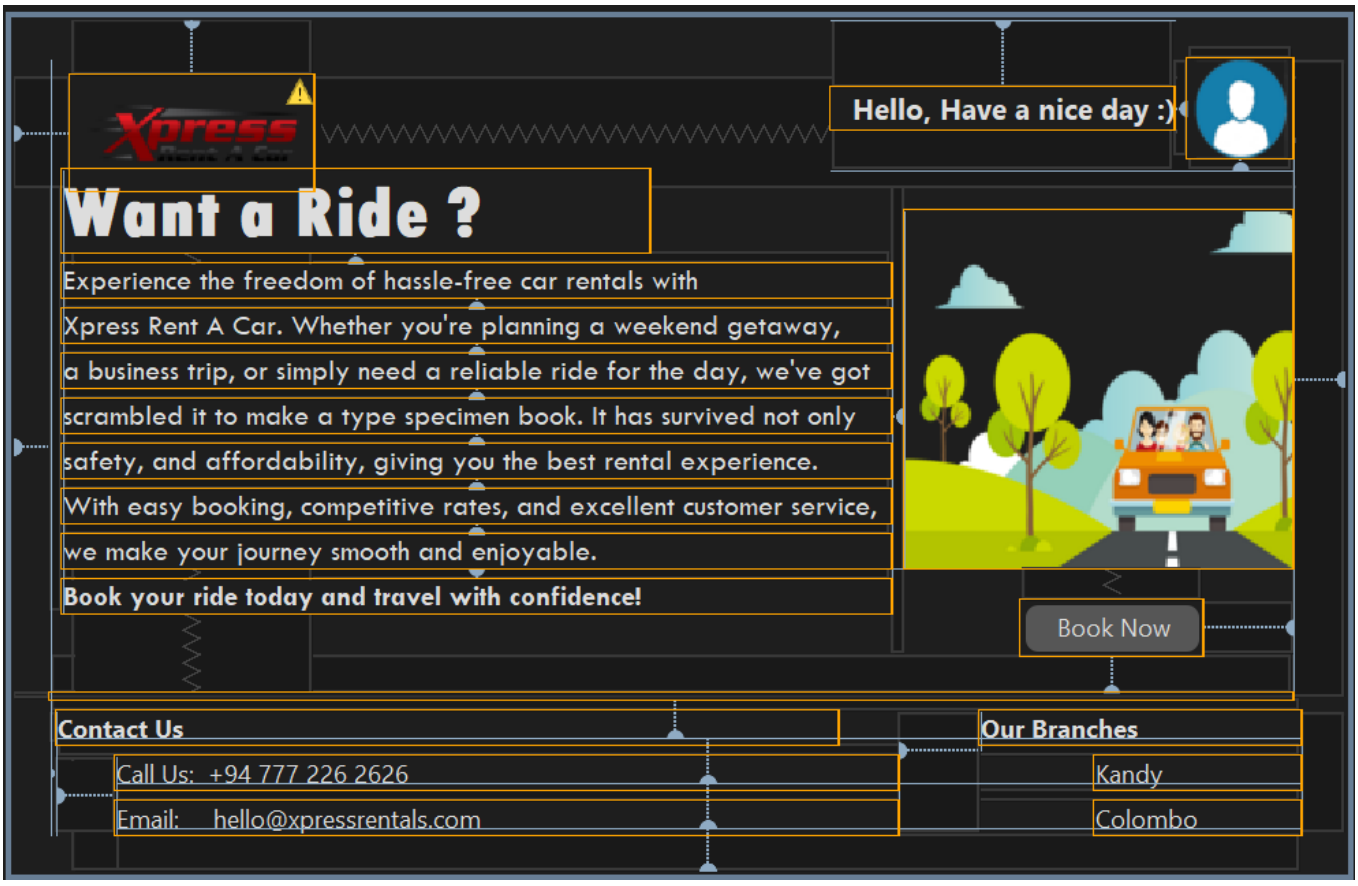
The Code

```
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;



public class User {
    private String username;
    private String password;
    private String passwordReCheck;
    private String firstName;
    private String lastName;
    private String phone;
    private String email;
```

```
    public User(String username, String password, String
passwordReCheck, String firstName, String lastName, String phone,
String email) {
        this.username = username;
        this.password = password;
        this.firstName = firstName;
        this.lastName = lastName;
        this.phone = phone;
        this.email = email;
        this.passwordReCheck = passwordReCheck;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getPhone() {
```

```
        return phone;
    }

    public String getEmail() {
        return email;
    }

    public String getpasswrdReCheck(){
        return passwerdReCheck;
    }

    public void saveToFile(String filePath) {
        try (FileWriter writer = new FileWriter(filePath, true)) {
            writer.append("Username: " + username + "\n");
            writer.append("Password: " + password + "\n");
            writer.append("First Name: " + firstName + "\n");
            writer.append("Last Name: " + lastName + "\n");
            writer.append("Phone: " + phone + "\n");
            writer.append("Email: " + email + "\n\n");

            JOptionPane.showMessageDialog(null, "File saved
successfully!");
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error saving File:
" + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Home Page (After logged in)



Displays the same list of text as Home page (Before login) but there is no login or sign up button (Cause this page is only appeared once the user logged in to the system)

Allows user to hire a vehicle as their preference by inserting the number of days(Duration)

JFrame: The main window of the application.

JButton: Used to create buttons such as “Book now”.

JLable: Used to display text & images.

JPanel: Used to group components together.

Inheritance concept was used to inherit the home page, & the concept of encapsulation was used to make the attributes private.

The Code

```
private void btnBookNowActionPerformed(java.awt.event.ActionEvent
evt) {
        VehicleSelection obj01 = new VehicleSelection();
        obj01.setVisible(true);
        dispose();
    }
```

Vehicle selection page

Allows users to select a vehicle (Bike,Car,Van or Jeep) according to their preference.

JFrame: The vehicle booking window of the application.

JButton: Used to create buttons such as “Home”, “Book now”.

JLable: Used to display text & images.

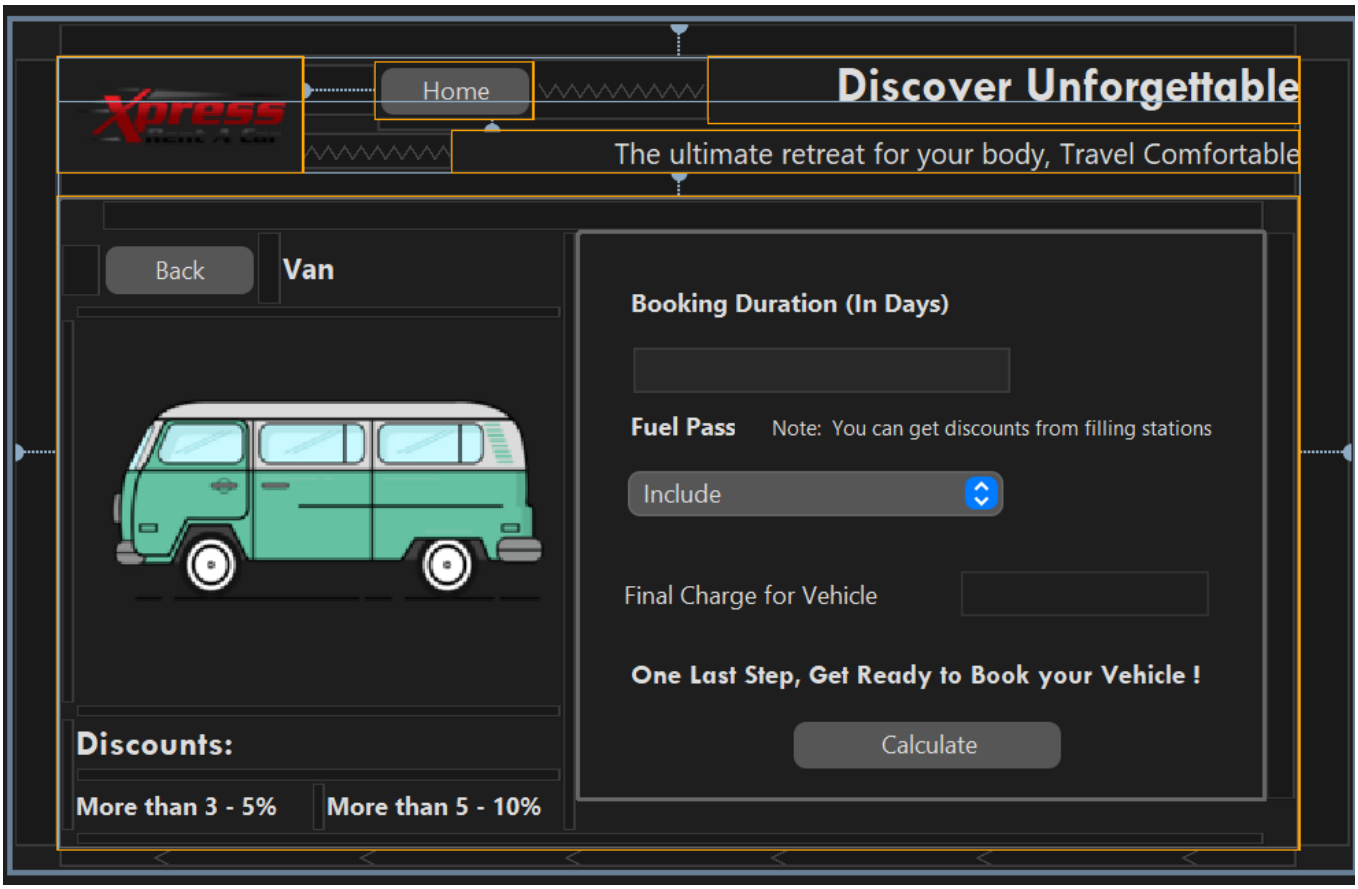
The Code

```
private void btnHomeActionPerformed(java.awt.event.ActionEvent evt)
{
        if (!LoginManager.getInstance().isLoggedIn()) {
            HomePage obj01 = new HomePage();
            obj01.setVisible(true);
            dispose();
            return;
        }
        HomePageLoggedIn obj01 = new HomePageLoggedIn();
        obj01.setVisible(true);
        dispose();
    }

    private void
btnBookNowVehicle01ActionPerformed(java.awt.event.ActionEvent evt)
{
        if (!LoginManager.getInstance().isLoggedIn()) {
            JOptionPane.showMessageDialog(null, "Please log in
first.", "Access Denied", JOptionPane.ERROR_MESSAGE);
            return;
        }

        CostCalculationVehicle01 obj01 = new
CostCalculationVehicle01();
        obj01.setVisible(true);
        dispose();
    }

    private void
btnBookNowVehicle02ActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        if (!LoginManager.getInstance().isLoggedIn()) {
            JOptionPane.showMessageDialog(null, "Please log in
first.", "Access Denied", JOptionPane.ERROR_MESSAGE);
            return;
        }

        CostCalculationVehicle02 obj01 = new
CostCalculationVehicle02();
        obj01.setVisible(true);
        dispose();
    }

    private void
btnBookNowVehicle03ActionPerformed(java.awt.event.ActionEvent evt)
{
        if (!LoginManager.getInstance().isLoggedIn()) {
            JOptionPane.showMessageDialog(null, "Please log in
first.", "Access Denied", JOptionPane.ERROR_MESSAGE);
            return;
        }

        CostCalculationVehicle03 obj01 = new
CostCalculationVehicle03();
        obj01.setVisible(true);
        dispose();
    }

    private void
btnBookNowVehicle04ActionPerformed(java.awt.event.ActionEvent evt)
{
        if (!LoginManager.getInstance().isLoggedIn()) {
            JOptionPane.showMessageDialog(null, "Please log in
first.", "Access Denied", JOptionPane.ERROR_MESSAGE);
```

```
            return;
        }

        CostCalculationVehicle04 obj01 = new
CostCalculationVehicle04();
        obj01.setVisible(true);
        dispose();
    }
```

Cost Calculation Page (Ex: Van)



The user is able to calculate the charge for vehicle by inserting the duration & selecting an option for fuel pass (Include/ Exclude). User can back to the Vehicle selection page by clicking on back button & also user can back to the home page by clicking on home page.

JFrame: The cost calculation window of the application.

JButton: Used to create buttons such as “Home”, “Back”, “Calculate”.

JComboBox: Used to select “Fuel pass”

JLable: Used to display text & images.

The Code

```
private void btnCalculateActionPerformed(java.awt.event.ActionEvent
evt) {
        if(txtBookingDuration.getText().equals("")){
            JOptionPane.showMessageDialog(rootPane, "Please fill
Booking Duration!!");
        }
        else{
            txtFInalChargeForVehicle.setText(""+
VehicleBookingVan.calculateFinalCharge(txtBookingDuration.getText()
, cmbFuelPass.getSelectedItem()));

            //Saving to text
            String bookingDuration = txtBookingDuration.getText();
            Object fuelPass = cmbFuelPass.getSelectedItem();
            double finalCharge =
VehicleBookingVan.calculateFinalCharge(bookingDuration, fuelPass);
            saveToFile(bookingDuration, fuelPass.toString(),
String.valueOf(finalCharge));
        }
    }
    public static class VehicleBookingVan {

        //Using private fields to encapsulate the data
        private int bookingDuration;
        private boolean FuelPassIncluded;

        // Constructor for data inputs
        public VehicleBookingVan(String bookingDuration, String
fuelPass) {
```

```
            this.bookingDuration =
Integer.parseInt(bookingDuration);
            this.FuelPassIncluded = "Include".equals(fuelPass);
        }

        // Calculating the final charge
        public static double calculateFinalCharge(String
bookingDuration, Object fuelPass) {
            //Van's charge
            int vanCharge = 2500;
            boolean isFuelPassIncluded =
"Include".equals(fuelPass);

            VehicleBookingVan booking = new
VehicleBookingVan(bookingDuration, fuelPass.toString());
            double fuelPassCharge = booking.getFuelPassCharge();
            double bookingDiscount = booking.getBookingDiscount();

            return ((vanCharge * booking.getBookingDuration()) +
fuelPassCharge) * bookingDiscount;
        }

        // Using a Getter for  get booking duration
        public int getBookingDuration() {
            return bookingDuration;
        }

        // Using a Getter for calculate discount by using the
duration
        public double getBookingDiscount() {
            if (bookingDuration <= 3) {
                return 1.0;
```

```
            } else if (bookingDuration <= 5) {
                return 0.95;
            } else {
                return 0.10;
            }
        }

        // Using a Getter for fuel pass to check weather include or
exclude
        public double getFuelPassCharge() {
            return FuelPassIncluded ? 1500 : 0;
        }
    }



    private void btnBackActionPerformed(java.awt.event.ActionEvent
evt) {
        VehicleSelection obj01 = new VehicleSelection();
        obj01.setVisible(true);
        dispose();
    }

    private void btnHomeActionPerformed(java.awt.event.ActionEvent
evt) {
        HomePageLoggedIn obj01 = new HomePageLoggedIn();
        obj01.setVisible(true);
        dispose();
    }

    private void saveToFile(String bookingDuration, String
fuelPass, String finalCharge) {
```

```
        try (FileWriter writer = new
FileWriter("booking_info_Van.txt", true)) {
            writer.write("Booking Duration: " + bookingDuration +
"\n");
            writer.write("Fuel Pass: " + fuelPass + "\n");
            writer.write("Final Charge: " + finalCharge + "\n\n");


            JOptionPane.showMessageDialog(rootPane, "Information
saved to booking_info_Van.txt!");
        }
        catch (IOException e) {
            JOptionPane.showMessageDialog(rootPane, "Error saving
file: " + e.getMessage());
        }
    }
```

Indexes

- KIC-DCSAI-242-F-006
- KIC-DCSAI-242-F-038
- KIC-DCSAI-242-F-040
- KIC-DCSAI-242-F-049