IFB299 IT Project Design and Development
Personal Portfolio

(Team No. 70)
(Kaushal Kishorbhai Limbasiya – N10032029)
(GitHub: IFB299-Group-70-S.K.R.A.M)

Artefact 1 – (Edit Car Information)

Edit car information is one of the main functionality use to edit car which can be only done by staff. At the same time, it also updates the car information store in the database. It also has functionality implemented by Riley Duggan about user verification. Car details can only be edited when staff member is logged in. It uses POST method to update car details to database.

```
def detail(request, car_id):
    carInfo = Cars.objects.filter(id=car_id)
    context = {'CarInfo': carInfo}

    if not UserVerification.StaffLoggedIn(request):
        return render(request, 'testApp/showcaroriginal.html', context)
    post = Cars.objects.get(pk = car_id)
    if request.method == 'POST':
        if (request.POST.get('carname') and request.POST.get('carmodel') and
        request.POST.get('carseries') and request.POST.get('carseriesyear') and
        request.POST.get('carpricenew') and request.POST.get('carenginesize') and request.POST.get('carfuelsystem')
        and request.POST.get('cartankcapacity') and request.POST.get('carpower') and request.POST.get('carseatingcapacity')
        and request.POST.get('carstandardtransmission') and request.POST.get('carbodytype')
        and request.POST.get('cardrive') and request.POST.get('wheelbase')):
            f = forms.CharField()
            g = forms.IntegerField()
            post.car_makename = f.clean(request.POST.get('carname'))
            post.car_model = f.clean(request.POST.get('carmodel'))
            post.car_series = f.clean(request.POST.get('carseries'))
            post.car_seriesyear = g.clean(request.POST.get('carseriesyear'))
            post.car_pricenew = g.clean(request.POST.get('carpricenew'))
            post.car_enginesize = f.clean(request.POST.get('carenginesize'))
            post.car_fuelsystem = f.clean(request.POST.get('carfuelsystem'))
            post.car_tankcapacity = f.clean(request.POST.get('cartankcapacity'))
            post.car_power = f.clean(request.POST.get('carpower'))
            post.car_seatingcapacity = g.clean(request.POST.get('carseatingcapacity'))
            post.car_standardtransmission = f.clean(request.POST.get('carstandardtransmission'))
            post.car_bodytype = f.clean(request.POST.get('carbodytype'))
            post.car_drive = f.clean(request.POST.get('cardrive'))
            post.car_wheelbase = f.clean(request.POST.get('wheelbase'))
            post.save()
            messages.add_message(request, messages.INFO, 'Succefully updated car info')
    return render(request, 'testApp/showcaroriginalMikeUpdate.html', context)
```

Artefact 2 – (Edit User Function)

This function is similar to customer sign up. It helps to edit information of logged in customer. This edited customer information is then updated to the database. With the help of POST method, it updates customer information in the database. This function also has user verification which only allows to edit customer information only by logged in customer. This user verification was implemented by Riley Duggan.

```
def editUser(request):
    if not UserVerification.CustomerLoggedIn(request):
        messages.add_message(request, messages.INFO, 'You MUST be logged in to access that page')
        return redirect("/accounts/login/")
    userProfile = Profile.objects.get(user = request.user)
    post = Customers.objects.get(pk = userProfile.customerid_id)
    if request.method == 'POST':
        if (request.POST.get('firstname') and request.POST.get('Phonenumber') and request.POST.get('Homeaddress')
        and request.POST.get('DOB') and request.POST.get('youremail') and request.POST.get('Gender') and request.POST.get('Occupation')):
            f = forms.CharField()
            post.name = f.clean(request.POST.get('firstname'))
            post.phone = f.clean(request.POST.get('Phonenumber'))
            post.address = f.clean(request.POST.get('Homeaddress'))
            post.dob = f.clean(request.POST.get('DOB'))
            post.occupation = f.clean(request.POST.get('Occupation'))
            post.gender = f.clean(request.POST.get('Gender'))
            post.email = f.clean(request.POST.get('youremail'))
            post.save()
            messages.add_message(request, messages.INFO, 'Succefully updated your information')
        else:
            messages.add_message(request, messages.INFO, 'Failed to updated your information')
    dob = post.dob
    if dob[2] == "-":
        if int(dob[6:8]) < 20:
            dob = "20" + dob[6:8] + "-" + dob[3:5] + "-" + dob[0:2]
        else:
            dob = "19" + dob[6:8] + "-" + dob[3:5] + "-" + dob[0:2]

    context = {'customer': post, 'dob': dob}
    return render(request,'testApp/MikeUserLandingPage.html', context)
```

Artefact 3 – (Edit customer HTML template to check functionality)

Staff members can edit car details using this HTML template. This template shows the car details which is previously stored in the database using value attribute in HTML. This data is displayed on text field and can be edited by staff member. Car detail depends on car id which is stored in the database. It has also got small button called Update which sends updated information to the database. When Update button is clicked it runs the function ‘detail’ from views.py.

```
<form action = "/CarInfo/{{car.id}}/" method = "POST" >
 {% csrf_token %}
 <label for="car_makename"><b>Car make name</b></label>
 <input id = "car_makename" type="text" placeholder="car.car_makename" name="carname" required value="{{car.car_makename}}" maxlength="255">
 <br>
 <label for="car_model"><b>Car Model</b></label>
 <input id = "car_model" type="text" placeholder="car.car_model" name="carmodel" required value="{{car.car_model}}" maxlength="255">
 <br>
 <label for="car_series"><b>Car Series</b></label>
 <input id = "car_series" type="text" placeholder="car_series" name="carseries" required value="{{car.car_series}}" maxlength="255">
 <br>
 <label for="car_seriesyear"><b>Car Series Year</b></label>
 <input id = "car_seriesyear" type="text" placeholder="car_seriesyear" name="carseriesyear" required value="{{car.car_seriesyear}}">
 <br>
 <label for="car_pricenew"><b>Car price new</b></label>
 <input id = "car_pricenew" type="text" placeholder="car_pricenew" name="carpricenew" required value="{{car.car_pricenew}}">
 <br>
 <label for="car_enginesize"><b>Car engine size</b></label>
 <input id = "car_enginesize" type="text" placeholder="car_enginesize" name="carenginesize" required value="{{car.car_enginesize}}" maxlength="255">
 <br>
 <label for="car_fuelsystem"><b>Car fuel system</b></label>
 <input id = "car_fuelsystem" type="text" placeholder="car_fuelsystem" name="carfuelsystem" required value="{{car.car_fuelsystem}}" maxlength="255">
 <br>
 <label for="car_tankcapacity"><b>Car tank capacity</b></label>
 <input id = "car_tankcapacity" type="text" placeholder="car_tankcapacity" name="cartankcapacity" required value="{{car.car_tankcapacity}}" maxlength="255">
 <br>
 <label for="car_power"><b>Car power</b></label>
 <input id = "car_power" type="text" placeholder="car_power" name="carpower" required value="{{car.car_power}}" maxlength="255">
 <br>
 <label for="car_seatingcapacity"><b>Car seating capacity</b></label>
 <input id = "car_seatingcapacity" type="text" placeholder="car_seatingcapacity" name="carseatingcapacity" required value="{{car.car_seatingcapacity}}">
 <br>
 <label for="car_standardtransmission"><b>Car standard transmission</b></label>
 <input id = "car_standardtransmission" type="text" placeholder="car_standardtransmission" name="carstandardtransmission" required value="{{car.car_standardtransmission}}" maxlength="255">
 <br>
 <label for="car_bodytype"><b>Car BOdytype</b></label>
 <input id = "car_bodytype" type="text" placeholder="car_bodytype" name="carbodytype" required value="{{car.car_bodytype}}" maxlength="255">
 <br>
 <label for="car_drive"><b>Car Drive</b></label>
 <input id = "car_drive" type="text" placeholder="car_drive" name="cardrive" required value="{{car.car_drive}}" maxlength="3">
 <br>
 <label for="car_wheelbase"><b>Car Wheelbase</b></label>
 <input id = "car_wheelbase" type="text" placeholder="car_wheelbase" name="wheelbase" required value="{{car.car_wheelbase}}" maxlength="255">
 <br>
 <input type="submit" value="Update">
</form>
```

Artefact 4 – (Update Customer Information HTML template for testing)

Customers can edit their details using this HTML template. This template shows the customer details which is previously stored in the database using value attribute in HTML. This data is displayed on text field and can be edited by customers who are already signed up. Customer information depends on customer id which is stored in the database. It has also got small button called Update which sends updated information to the database. When Update button is clicked it runs the function ‘editUser’ from views.py.

```
<form action = "/editUser/" method = "POST">
    {% csrf_token %}
    <label for ="name">Name:</label><br>
    <input id="name" type="text" name="Name" maxlength="225" value="{{customer.name}}" required>
    <br>
    <label for ="phone">Phone:</label><br>
    <input id = "phone" type="Phone" name="Phone" maxlength="20" value="{{customer.phone}}" required>
    <br>
    <label for ="address">Address:</label><br>
    <input id="address" type="text" name="Address" maxlength="255" value="{{customer.address}}" required>
    <br>
    <label for ="dob">DOB:</label><br>
    <input id ="dob" type="date" name="DOB" maxlength="20" value = "{{dob}}" required>
    <br>
    <label for ="occupation">Occupation:</label><br>
    <input id="occupation" type="text" name="Occupation" maxlength="255" value="{{customer.occupation}}">
    <br>
    <label for ="gender">Gender:</label><br>
    <input id="gender" type="radio" name="Gender" value="M" maxlength="3"> Male<br>
    {%if customer.gender == M%}
    <input id="gender" type="radio" name="Gender" value="F" maxlength="3" > Female
    {%else%}
    <input id="gender" type="radio" name="Gender" value="F" maxlength="3" checked = "checked" > Female
    {%endif%}
    <br>
    <label for ="email">Email:</label><br>
    <input id="email" type="email" name="Email" placeholder="abcd123@example.com" maxlength="255" value = "{{customer.email}}" required>
    <br>
    <br>
    <input type="submit" value="Update">
</form>
```

Artefact 5 – (Unit testing for edit Customer view.)

Unit testing is one of the best way to test the working of the code. This unit test is used to check edit customer view. It includes the logged in and logged out test for the customer, template test when customer is logged in and logged out. It also tests template redirection when staff member or board member logs in. Apart from this it also tests context of ‘customer’ and ‘dob’. If the any of the test fails it shows error code of the test failed.

```python
class test_EditCustomersView(TestCase):
    @classmethod
    def setUpTestData(cls):
        CreateUsers()

    def test_LoggedOut(self):
        response = self.client.post('/editUser/')
        self.assertRedirects(response, '/accounts/login/', status_code=302, target_status_code=200, msg_prefix='', fetch_redirect_response=True)

    def test_LoggedInAsCustomer(self):
        self.client.login(username="customer", password="1234")
        response = self.client.get('/editUser/')
        self.assertEqual(response.status_code, 200)

    def test_Context(self):
        self.client.login(username="customer", password="1234")
        response = self.client.get('/editUser/')
        self.assertEqual(response.status_code, 200)
        customer = response.context['customer']
        dob = response.context['dob']
        self.assertIsInstance(customer, Customers)
        self.assertIsInstance(dob, str)


    def test_LoggedInAsBM(self):
        self.client.login(username="BM", password="1234")
        response = self.client.get('/accounts/login/')
        self.assertEqual(response.status_code, 200)


    def test_LoggedInAsStaff(self):
        self.client.login(username="staff", password="1234")
        response = self.client.get('/accounts/login/')
        self.assertEqual(response.status_code, 200)
```