**Lab 10 Part 9**

**Sign Up, Sign In & Sign Out**

In this exercise we will add authentication to the online shopping project to enable a user to sign up, sign in and sign out.

**Step 1: Sign Up**

Run the server and log in to Django Admin. Create a new **group** called **Customer**, click Save and then log out. When a new user signs up, they will be part of a group of users called Customer and the user itself will be based on the **CustomUser** model that was already provided in the starter code.

Graphical user interface, application

Description automatically generated

Copy-paste the following imports to **accounts/views.py**:

from django.shortcuts import render, redirect

from django.urls import reverse\_lazy

from django.views.generic.edit import CreateView

from django.contrib.auth.models import Group

from .forms import CustomUserCreationForm

from .models import CustomUser

Copy-paste the following code into **accounts/views.py:**

class SignUpView(CreateView):

    form\_class = CustomUserCreationForm

    template\_name = 'registration/signup.html'

    def post(self, request, \*args, \*\*kwargs):

        form = CustomUserCreationForm(request.POST)

        if form.is\_valid():

            form.save()

            username = form.cleaned\_data.get('username')

            signup\_user = CustomUser.objects.get(username=username)

            customer\_group = Group.objects.get(name='Customer')

            customer\_group.user\_set.add(signup\_user)

            return redirect('login')

        else:

            return render(request, self.template\_name, {'form' : form })

The first two lines of code in the class-based view are needed to do the following:

* Specify which form class to use. We use the **CustomUserCreationForm** that was created in **forms.py** as part of the **accounts** app
* Template is named **signup.html** & is in the registration folder

To build the logic to assign a new user to the Customer group we override the Django method **post**.

In this method we do the following:

* We create a form instance and populate it with data from the request: form = CustomUserCreationForm(request.POST) This is called “binding data to the form” (it is now a bound form).
* We call the form’s is\_valid() method; if it is not True, we go back to the template with the form. This time the form is no longer empty (unbound) so the HTML form will be populated with the data previously submitted, where it can be edited and corrected as required.
* If the form data is valid then we save the form
  + We find all the validated form data in its cleaned\_data attribute i.e. the user name. Whatever the data submitted with a form, once it has been successfully validated by calling is\_valid() (and is\_valid() has returned True), the validated form data will be in the form.cleaned\_data dictionary. This data will have been nicely converted into Python types for you.
  + We assign a variable signup\_user to contain the new custom user object
  + We assign a variable customer\_group to contain the customer group object
  + Add the new user to the group & redirect the user to the sign in page
* If the form is not valid then we call the render function which contains the request object, the template and a dictionary called form. In other words, we show the form again to the user.

Create a **urls.py** file in the **accounts** app folder and copy-paste the following code:

from django.urls import path

from shop import views

from .views import SignUpView

urlpatterns = [

    path('create/', SignUpView.as\_view(), name='signup'),

]

Create a folder called **templates** in the **accounts** app and create a new folder inside the templates folder called **registration**. Inside this folder create a template called **signup.html**.

Copy-paste the following code:

{% extends "base.html" %}

{% load static %}

{% load crispy\_forms\_tags %}

{% block metadescription %}

Sign up to get a new account with us. The perfect

cushion store has a variety of stunning and comfy cushions.

{% endblock %}

{% block title %}

Create a New Account - Perfect Cushion Store

{% endblock %}

{% block content %}

<div>

  {% if not form.is\_valid %}

  <div class="mx-auto">

    <br />

    <h1 class="my\_title text-center">Create a new account</h1>

    <br />

    <div class="col-9 col-sm-9 col-md-6 col-lg-5 mx-auto bg-light">

      <br />

      <p>Enter the following information to create a new account.</p>

      <form method="post">

        <p>{{ form | crispy }}</p>

        {% csrf\_token %}

        <button type="submit" class="btn btn-secondary">

          Register Account

        </button>

      </form>

      <br />

    </div>

  </div>

  {% endif %}

</div>

<br />

{% endblock %}

Open **custom.css** and copy-paste the following code:

.errorlist li {

color: red;

font-weight: bold;

}

This class controls the behaviour of the form when a user tries to create an account with a username that already exists. This class will highlight the error in red.

Open **onlineshop/urls.py** and add in the following code:

Text

Description automatically generated

If it is not installed, install the django crispy forms app:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**pip install crispy-bootstrap5**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Add the following entries to **settings.py**:

Text

Description automatically generated



**Step 2: Log In & Log out**

The built-in auth app already provides views and urls for log in and log out. When you log in or log out of a site, you are immediately redirected to a subsequent page. We need to tell Django where to send users in each case. The LOGIN\_REDIRECT\_URL and LOGOUT\_REDIRECT\_URL settings do that. Add these two lines at the bottom of the settings.py file.



To use the urls for log in and log out, add the following code to **shop\_project/urls.py**:

Text

Description automatically generated

Create a new template called **login.html** inside the **accounts/templates/registration** folder.

Copy-paste the following code:

{% extends "base.html" %}

{% load static %}

{% load crispy\_forms\_tags %}

{% block metadescription %}

    Sign in and enjoy the variety of stunning and comfy cushions.

{% endblock %}

{% block title %}

    Login - Perfect Cushion Store

{% endblock %}

{% block content %}

    <div class="container">

        <br>

        <h1 class="my\_title text-center">Login or Create a New Account</h1>

        <br>

        <div class="row">

            <div class="col-12 col-sm-12 col-md-12 col-lg-6 text-center my\_custom\_div">

                <br>

                <h2 class="my\_title">New Customer?</h2>

                <br>

                <a href="{% url 'signup' %}" class="btn btn-secondary">Create New Account</a>

            </div>

            <div class="col-12 col-sm-12 col-md-12 col-lg-6 bg-light">

                <div>

                    <br>

                    <h2 class="my\_title">Registered Customers Only</h2>

                    <form method="post">

                        {% csrf\_token %}

                        <p>{{ form | crispy }}</p>

                        <button type="submit" class="btn btn-secondary">Login</button>

                    </form>

                    <br>

                </div>

            </div>

        </div>

        <br>

    </div>

    <br>

{% endblock %}

Open **custom.css** and add in the following code:

.my\_custom\_div {

margin-bottom: 20px !important;

}

We will now add a top nav bar so that we can see Sign Up, Sign In, Sign Out links at the top of our page.

Create a new template called **top-navbar.html** inside the **shop/templates** folder & add in the following code:

<div class="container">

    <div class="my\_custom\_menu">

        <ul>

            {% if user.is\_authenticated %}

            <li>

                Welcome, <a href="#" class="dropdown-toggle" data-bs-toggle="dropdown"

                aria-haspopup="true" aria-expanded="false">{{ user.username }}

                <i class="fas fa-user-circle"></i></a>

            </li>

            <li>

                <a href="{% url 'logout' %}">Sign Out <i class="fas fa-sign-out-alt"></i></a>

            </li>

            {% else %}

            <li>

                <a href="{% url 'signup' %}">Sign Up <i class="fas fa-user-plus"></i></a>

            </li>

            <li>

                <a href="{% url 'login' %}">Sign In <i class="fas fa-sign-in-alt"></i></a>

            </li>

            {% endif %}

        </ul>

    </div>

</div>

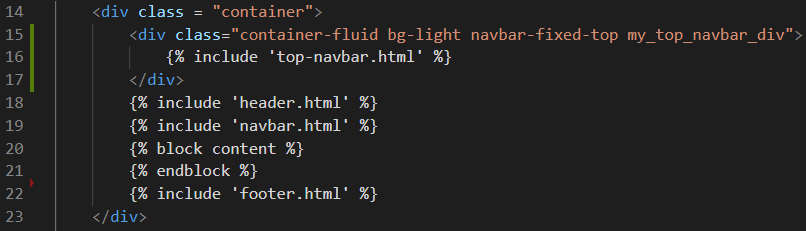
Open **base.html** and copy-paste the following code to the location shown below:

        <div class="container-fluid bg-light navbar-fixed-top

my\_top\_navbar\_div">

            {% include 'top-navbar.html' %}

        </div>



Open **custom.css** and add in the following code:

.my\_top\_navbar\_div {

height: 60px;

line-height: 50px;

}

.my\_custom\_menu {

text-align: right !important;

}

.my\_custom\_menu ul li {

list-style: none;

color: #000;

text-transform: uppercase;

letter-spacing: .2em;

font-size: 12px;

padding-right: 20px;

display: inline;

}

.my\_custom\_menu ul li a {

text-decoration: none;

color: #000;

list-style: none;

text-transform: uppercase;

letter-spacing: .2em;

font-size: 12px;

}

Run the server and you should see the new navigation bar at the top of the screen. If you were already logged in then clear the browser cache:



Try to sign up and log in with the new account details you created earlier & you should see a welcome message in the top nav bar.



**Step 3: Commit your changes and push your code to the lab 10 repo**

git add -A

git commit -m “lab 10 part 9 commit”

git push -u origin main