# Lab 5 Part (2)

## User Accounts

In this exercise we will look at how to authenticate users in our web application. Keep working on the project that you created in **lab 5**.

Django comes with a powerful, built-in user authentication system that we can use. Whenever you create a new project, by default Django installs the **auth** app, which provides us with a **User** object containing:

- username
- password
- email
- first_name
- last_name

We will use this **User** object to implement log in, log out, and sign up in our blog application.

## Log in

Django provides us with a default view for a log in page via **LoginView**. We need to make the following changes to our application to use this view:

- add a urlpattern for the auth system

- create a **log in** template

- make a small update to the **settings.py** file

We need to update the **studentproject/urls.py** file. We will add a path to this file which defines the accounts/ URL where we will place our **log in** and **log out** pages.

1. Add the line of code shown at line 6 below:

**Code**

```
17    from django.contrib import admin
18    from django.urls import path, include
19
20    urlpatterns = [
21        path('admin/', admin.site.urls),
22        path('accounts/', include('django.contrib.auth.urls')),
23        path('', include('student.urls')),
24    ]
```

By default, Django will look within a **templates** directory called **registration** for a file called **login.html** for a log in form. We need to create a new directory called **registration** and the requisite file within it.

2. In VS Code create a folder inside the **templates** folder called **registration**.
3. Create an empty file called **login.html** and add in the following code:

**Code**

```
templates > registration > <> login.html > ...
  1    {% extends 'base.html' %}
  2    {% block title %}
  3      Login Page
  4    {% endblock title %}
  5    {% block content %}
  6      <h1>Log In</h1>
  7      <form action="" method="POST">
  8          {% csrf_token %}
  9          {{ form.as_p }}
 10          <input type="submit" value="Log In"/>
 11      </form>
 12    {% endblock content %}
```

Line 7: We use HTML **<form></form>** tags and specify the **POST** method since we are sending data to the server (we would use **GET** if we were requesting data, such as in a search engine form).

Line 8: We add **{% csrf_token %}** for security concerns, namely, to prevent a XSS Attack.

Line 9: The form's contents are outputted between paragraph tags thanks to **{{ form.as_p }}** and then we add a "**Log In**" button.

**Re-direct User**

The final step is to specify where to redirect the user upon a successful log in. We can set this with the **LOGIN_REDIRECT_URL** setting.

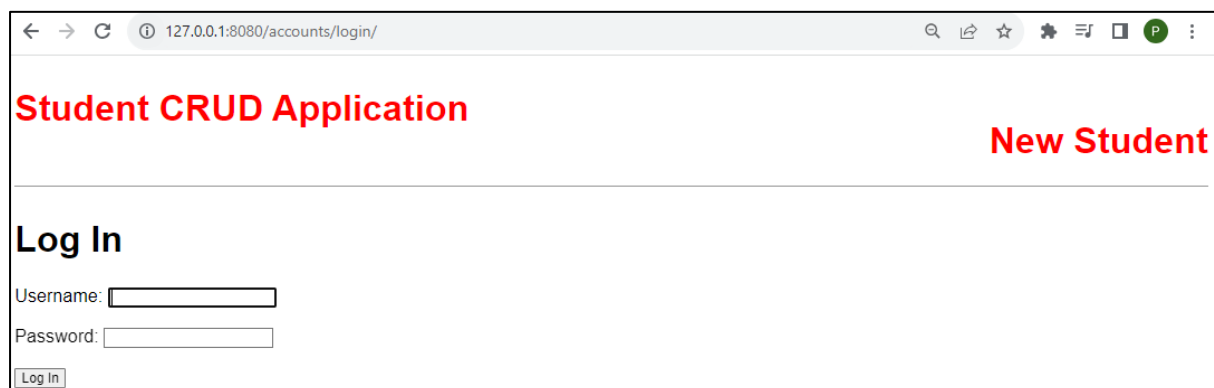1. At the bottom of the **settings.py** file add the following:

**Code**

```
126    LOGIN_REDIRECT_URL = 'home'
```

Now the user will be redirected to the **'home'** template which is our homepage.

**2.** Start up the Django server again with the command **python manage.py runserver** and navigate to our log in page: **http://127.0.0.1:8080/accounts/login/**

You will see the following:



3. Enter your username and password for your superuser account and press the **Log In** button.

You will be redirected to the homepage. Notice that we didn't add any view logic or create a database model because the Django **auth** system provided both for us automatically.

**Updated base.html**

We will update our **base.html** template so we display a message to users whether they are logged in or not. We can use the **is_authenticated** attribute for this.

1. Update the **base.html** file with the following code starting beneath the closing **</header>** tag.

**Code**

```
17        </header>
18        {% if user.is_authenticated %}
19            <p>Hi {{user.username }}</p>
20        {% else %}
21            <p>You are not logged in </p>
22            <a href="{% url 'login' %}">Log In</a>
23        {% endif %}
24        <body>
25            {% block content %}
26            {% endblock content %}
27        </body>
28   </html>
```

If the user is logged in, we say hello to them by name, if not we provide a link to a newly created log in page.

2. Run the server and access the site at: **http://127.0.0.1:8080/**
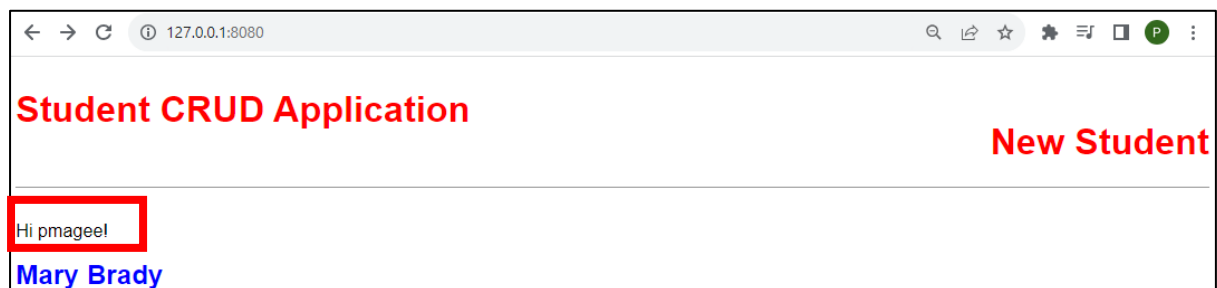


It worked. You should see your username on the page.

**Log out link**

Next, we will add a log out link that redirects the user to the homepage.

1. In the **base.html** file add a one-line **{% url 'logout' %}** link for logging out just below our user greeting.

**Code**

```
17        </header>
18        {% if user.is_authenticated %}
19            <p>Hi {{user.username }}</p>
20            <a href="{% url 'logout' %}">Log Out</a>
21        {% else %}
22            <p>You are not logged in </p>
23            <a href="{% url 'login' %}">Log In</a>
24        {% endif %}
25        <body>
26            {% block content %}
27            {% endblock content %}
28        </body>
29    </html>
```

That is all we need to do as the necessary view is provided to us by the Django **auth** app. We do need to specify where to redirect a user upon log out though.

2. Update **settings.py** to provide a redirect link which is called **LOGOUT_REDIRECT_URL**. Add this line of code just beneath the log in redirect as shown below:

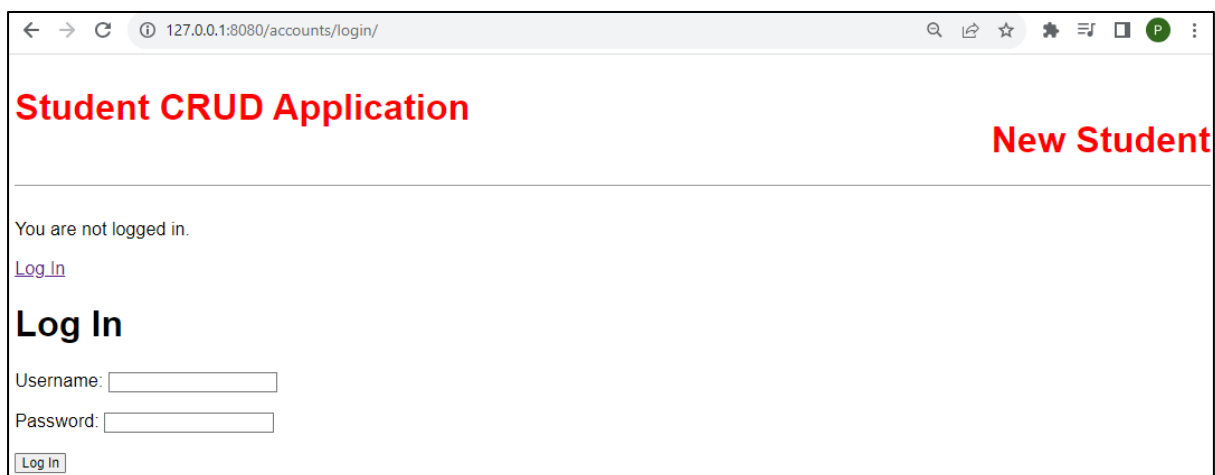**Code**

```
settings.py ×

blog_project > settings.py

125    # blog_project/settings.py
126    LOGIN_REDIRECT_URL = 'home'
127    LOGOUT_REDIRECT_URL = 'home' # new
```

3. Refresh the homepage & you will see the page now has a "log out" link for logged in users.

1. Click the logout link and see that it takes you back to the homepage with a login link.
2. Try logging in and out a few times with your user account.



**Sign-up**

We need to write our own view for a sign-up page to register new users, but Django provides us with a form class, **UserCreationForm**, to make things easier. By default, it comes with three fields: username, password1, and password2.

There are many ways to organize your code and URL structure for a user authentication system. Here we will create a dedicated new app called **accounts**, for our sign-up page.

In Windows Command Line use the following command to create a new app called **accounts**

_____

**python manage.py startapp accounts**

_____

1. Add the following line of code to the **INSTALLED_APPS** setting in our **settings.py** file.

**Code**

```
33   INSTALLED_APPS = [
34       'django.contrib.admin',
35       'django.contrib.auth',
36       'django.contrib.contenttypes',
37       'django.contrib.sessions',
38       'django.contrib.messages',
39       'django.contrib.staticfiles',
40       'blog',
41       'accounts',
42   ]
```

**Create the view**

1. Open the **accounts/views.py** file.
2. Delete the import at the top of the file on line 1.
3. Add the following code to the file. This new view uses the built-in **UserCreationForm** (see import on line 1) and generic **CreateView** (see import on line 3).

**Code**

```python
accounts > 🐍 views.py
1   from django.contrib.auth.forms import UserCreationForm
2   from django.urls import reverse_lazy
3   from django.views import generic
4
5
6   class SignUpView(generic.CreateView):
7       form_class = UserCreationForm
8       success_url = reverse_lazy('login')
9       template_name = 'registration/signup.html'
```

4. Create a new file called **signup.html** in the **templates/registration** directory & populate it with the following code:

**Code**

```
templates > registration > <> signup.html > ...
   1    {% extends 'base.html' %}
   2    {% block title %}
   3      Sign Up Page
   4    {% endblock title %}
   5
   6    {% block content %}
   7      <h1>Sign Up</h1>
   8        <form action="" method="POST">
   9          {% csrf_token %}
  10          {{ form.as_p }}
  11          <input type="submit" value="Sign Up"/>
  12        </form>
  13    {% endblock content %}
```

This format is very similar to what we have done before. We extend our base template at the top, place our logic between **<form></form>** tags, use the **csrf_token** for security, display the form's content in paragraph tags with **form.as_p**, and include a **submit** button.

**Configure the URLs**

**1.** Add the following line of code which adds a new URL path in **student_project/urls.py** pointing to this new app.

**Code**

```
blogproject > 🐍 urls.py
   1    from django.contrib import admin
   2    from django.urls import path, include # new
   3
   4    urlpatterns = [
   5        path('admin/', admin.site.urls),
   6        path('accounts/', include('django.contrib.auth.urls')),
   7        path('accounts/', include('accounts.urls')), # new
   8        path('', include('blog.urls')), # new
   9    ]
```
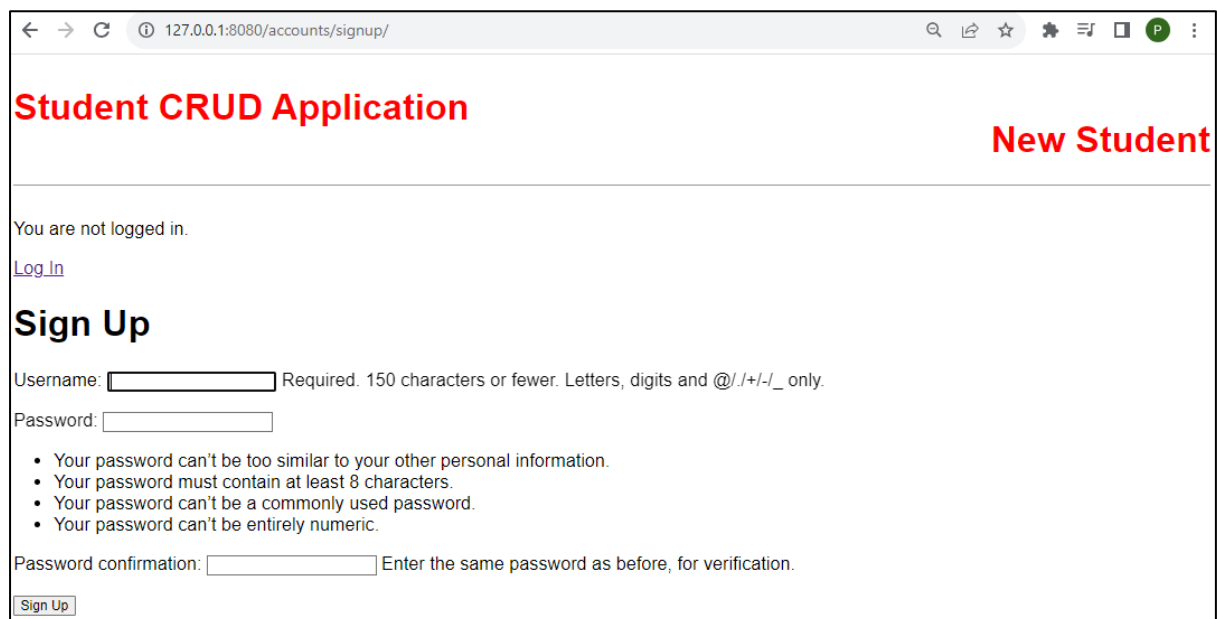
The order of our urls matters here because Django reads this file from top-to-bottom. Therefore, when we request the **/accounts/signup** url, Django will first look in **auth**, not find it, and then proceed to the **accounts** app.

2. In VS Code, create a new file inside the accounts folder called **urls.py** and add the following code into it.

**Code**

```
accounts > 🐍 urls.py > ...
   1    from django.urls import path
   2    from .views import SignUpView
   3
   4    urlpatterns = [
   5        path('signup/',SignUpView.as_view(), name='signup'),
   6
   7    ]
```

5. Start up the local server with the command **python manage.py runserver** and navigate to the newly created page: http://127.0.0.1:8080/accounts/signup/



Notice there is a lot of extra text that Django includes by default.

6. Create a new user and click **Sign up** which will redirect you to the log in page. Then after logging in successfully with your new user and password, you will be redirected to the homepage with a personalized "Hi username" greeting.
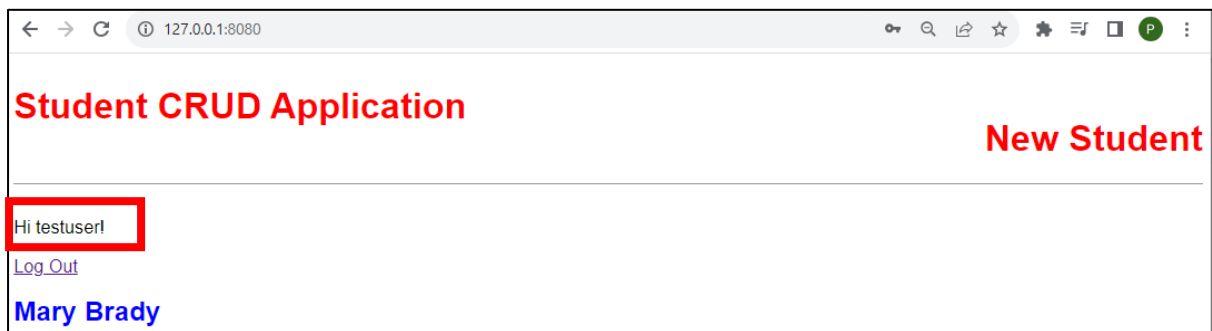
Run the following git commands to update the **lab 5** local and remote repositories:

**git add -A**

**git commit -m "lab 5 part 2 commit"**

**git push -u origin main**