**Lab 5 Part (3)**

**Bootstrap**

In this exercise we are going to improve the layout of our website. Instead of writing all the HTML and CSS code ourselves, we will use Bootstrap, the most popular framework for building responsive, mobile-first projects. Rather than write all our own code for common website layout features, we can instead rely on Bootstrap to do this for us. This means with only a small amount of code on our part we can quickly have great looking websites. And if we want to make custom changes as a project progresses, it’s easy to override Bootstrap where needed, too. When you want to focus on the functionality of a project and not the design, Bootstrap is a great choice.

**Pages app**

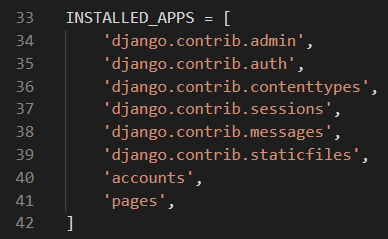
In the previous lab we displayed our homepage by including view logic in our urls.py file. While this approach works, it doesn’t scale as a website grows over time. It is also probably somewhat confusing to Django newcomers. Instead we can and should create a dedicated pages app for all our static pages. This will keep our code nice and organized going forward.

On the command line use the startapp command to create a new pages app. If the

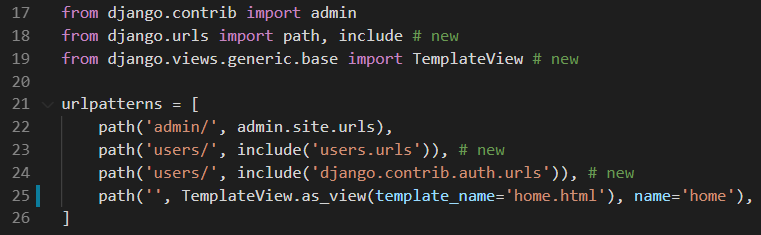
server is still running you may need to type Control+BREAK first to quit it.

(env) djangoprojects\lab5>python manage.py startapp pages

Open settings.py and update it with the name of this new app.



Now we can update our urls.py file inside the newspaperproject directory. Go ahead and remove the import of TemplateView. We will also update the ‘ ' route to include the pages app.

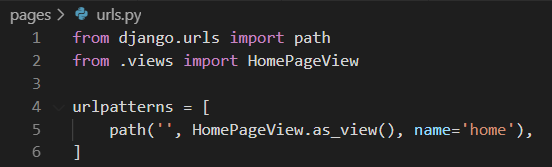


Remove line 19

Replace the code on line 25 with the code shown below



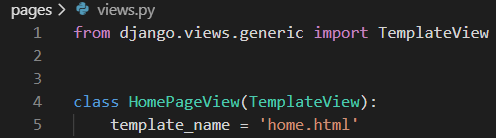
Create a new file urls.py inside the pages directory and add the following code to it:



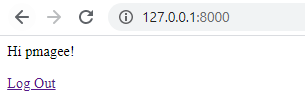
The views.py code should look familiar at this point. We’re using Django’s TemplateView generic class-based view which means we only need to specify our template\_name to use it.

**Views**

Open the file pages/views.py: and replace the code with the following:



We already have an existing home.html template. Let’s confirm it still works as expected with our new url and view. Start up the local server python manage.py runserver and navigate to the homepage at http://127.0.0.1:8000/ to confirm it remains unchanged.



**Bootstrap**

There are two ways to add Bootstrap to a project: you can download all the files and

serve them locally or rely on a Content Delivery Network (CDN). The second approach

is simpler to implement provided you have a consistent internet connection so that’s

what we’ll use here.

Bootstrap comes with a starter template that includes the basic files needed. Notably

there are four that we incorporate:

* Bootstrap.css
* jQuery.js
* Popper.js
* Bootstrap.js

Here’s what the updated base.html file should look like. Generally you should type all code examples yourself but as this is one is quite long, it’s ok to copy and paste here. Overwrite the base.html file with the code provided here.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}Newspaper App{% endblock title %}</title>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

</head>

<body>

<h1>Hello World</h1>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

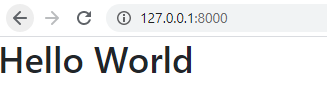
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script>

</body>

</html>

If you start the server again with python manage.py runserver and refresh the homepage at http://127.0.0.1:8000/ you will see that only the font size has changed at

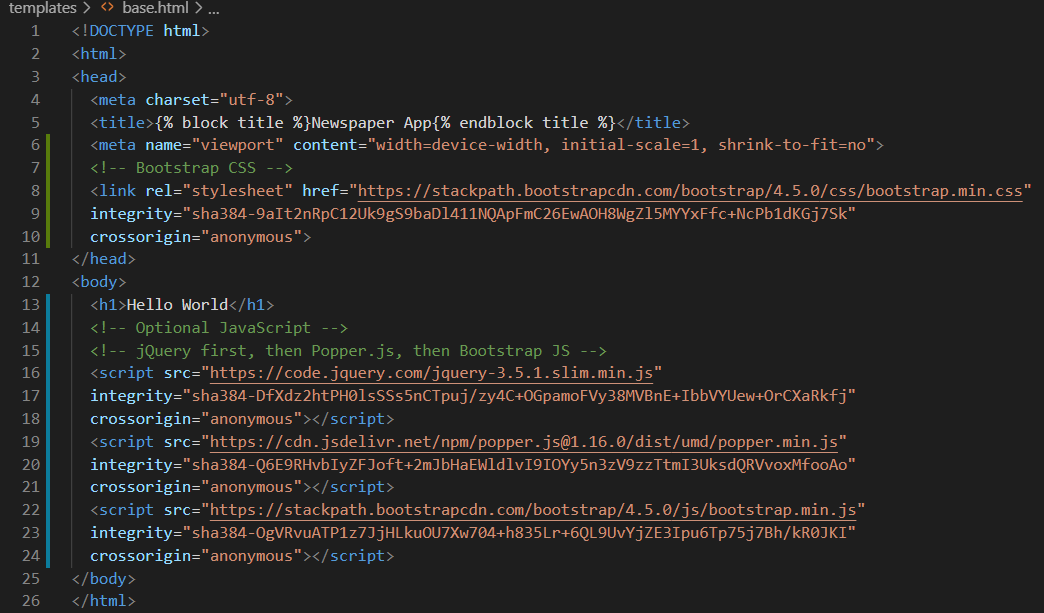
the moment.



Next we will add a navigation bar at the top of the page which contains our links for the homepage, log in, log out, and sign up. Notably we can use the if/else tags in the

Django templating engine to add some basic logic. We want to show a “log in” and “sign up” button to users who are logged out, but a “log out” and “change password” button to users logged in.

Delete the <h1> tag with “Hello World” i.e. line 13 and copy paste the code given on the next page into the base.html file at this location.



<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<a class="navbar-brand" href="{% url 'home' %}">Newspaper</a>

<button class="navbar-toggler" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

{% if user.is\_authenticated %}

<ul class="navbar-nav ml-auto">

<li class="nav-item">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"

role="button" data-toggle="dropdown" aria-haspopup="true"

aria-expanded="false">

{{ user.username }}

</a>

<div class="dropdown-menu dropdown-menu-lg-right"

aria-labelledby="navbarDropdown">

<a class="dropdown-item"

href="{% url 'password\_change'%}">Change password</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="{% url 'logout' %}">

Log Out</a>

</div>

</li>

</ul>

{% else %}

<form class="form-inline ml-auto">

<a href="{% url 'login' %}" class="btn btn-outline-secondary">

Log In</a>

<a href="{% url 'signup' %}" class="btn btn-primary ml-2">

Sign up</a>

</form>

{% endif %}

</div>

</nav>

<main>

<div class="container">

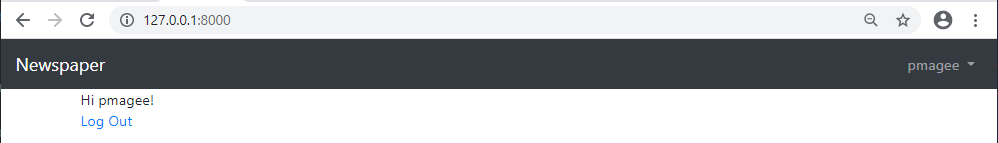
{% block content %}

{% endblock content %}

</div>

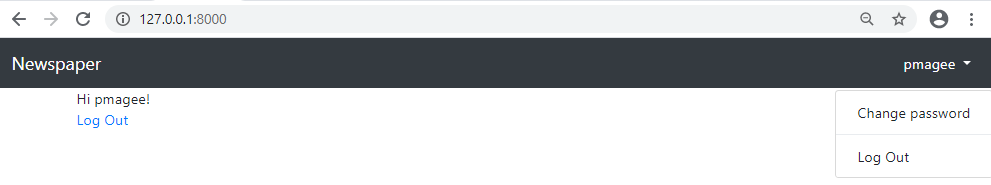
</main>

If you refresh the homepage at http://127.0.0.1:8000/ our new nav has appeared:



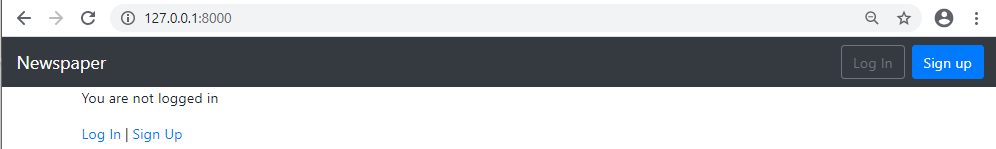
We have also added in our {% block content %} tags so the user greeting has returned, as has our “Newspaper App” in the title.

Click on the username in the upper right hand corner–pmagee in my case–to see the nice dropdown menu Bootstrap provides.

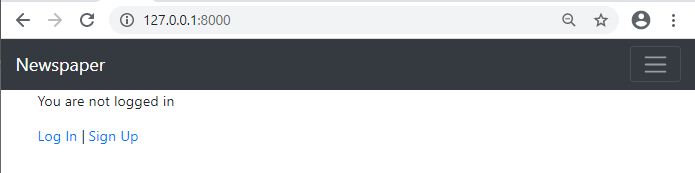


If you click on the “Log Out” link then our nav bar changes offering links to either “Log

In” or “Sign Up.”

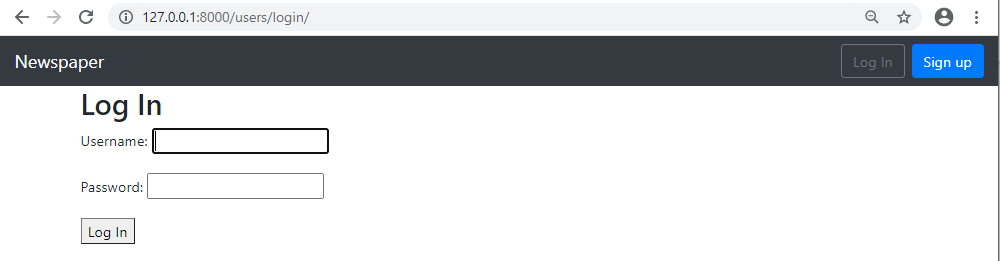


If you shrink the size of your browser window Bootstrap automatically resizes and adjusts so it looks good on a mobile device, too.

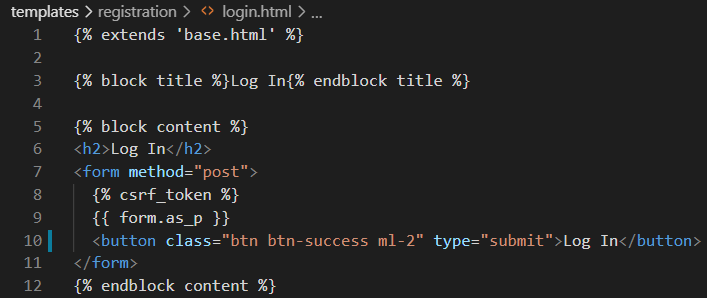


If you click on the “Log Out” button and then “Log In” from the top nav you can also

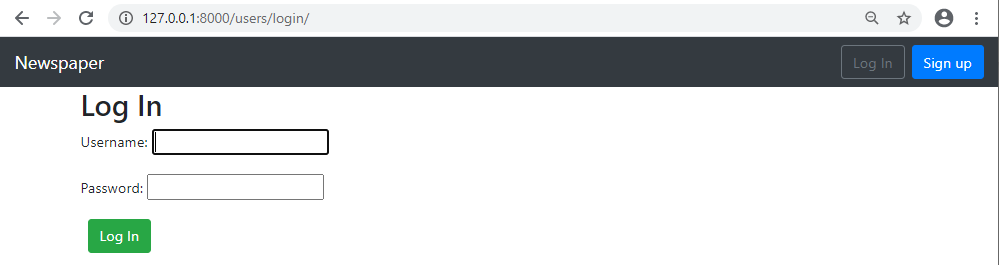
see that our log in page http://127.0.0.1:8000/users/login looks better too.



We can use Bootstrap to add some nice styling such as making the “Log in” button green and inviting. Change the “button” line in templates/registration/login.html as follows.



Now refresh the page to see our new button.

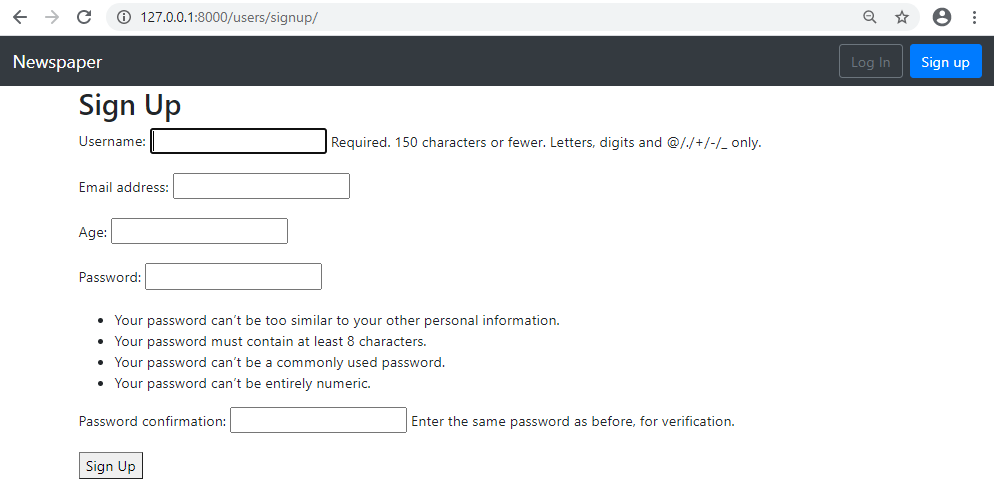


**Sign Up Form**

Our sign up page at http://127.0.0.1:8000/users/signup/ has Bootstrap stylings but

also distracting helper text. For example after “Username” it says “Required. 150

characters or fewer. Letters, digits and @/./+/-/\_ only.”



This helper text comes from Django. We will now look at a way to improve the layout of this helper text in our form. To do this will use a 3rd party application django-crispy-forms to help us to manage django forms.

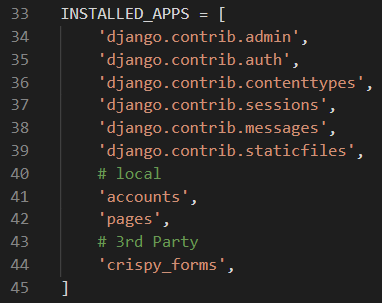
Stop the local server with Control+BREAK. Then use pip to install the package in our project.

**Command Line**

(env) djangoprojects\lab5>pip install django-crispy-forms

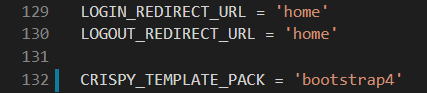
Add the new app to our INSTALLED\_APPS list in the settings.py file. As the number of apps starts to grow, it is helpful to distinguish between 3rd party apps and local apps we have added. Here’s what the code looks like now.

**Code**



Since we are using Bootstrap4 we should also add that config to our settings.py file. This goes on the bottom of the file.

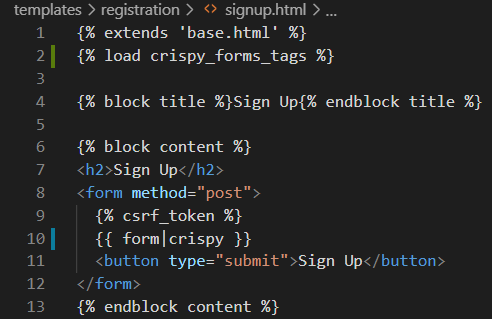
Code



Now in our signup.html template we can quickly use crispy forms. First we load

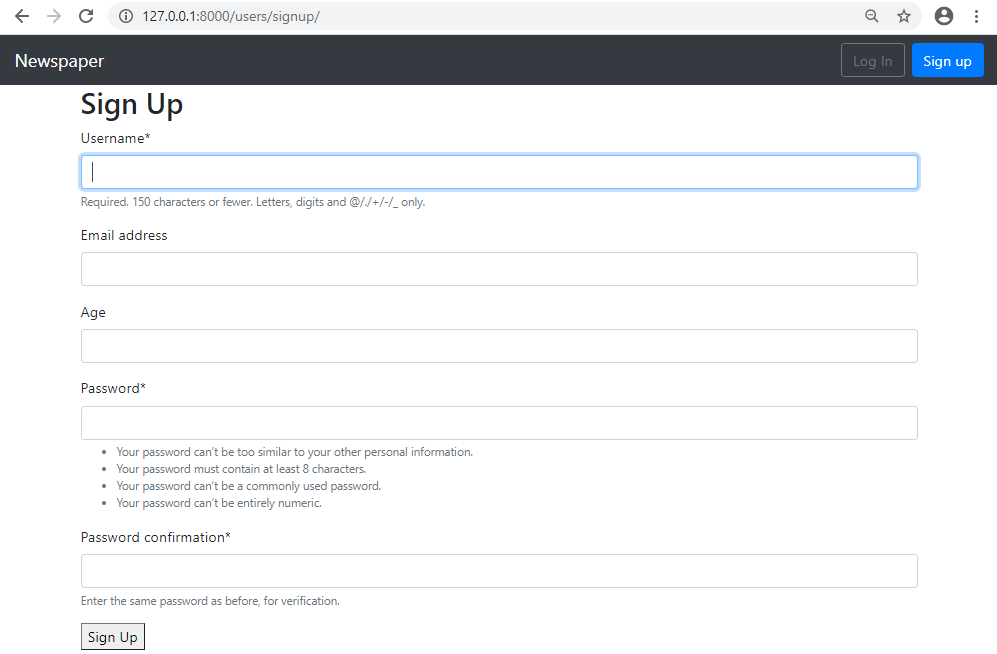
crispy\_forms\_tags at the top and then replace {{ form.as\_p }} for {{ form|crispy

}}.

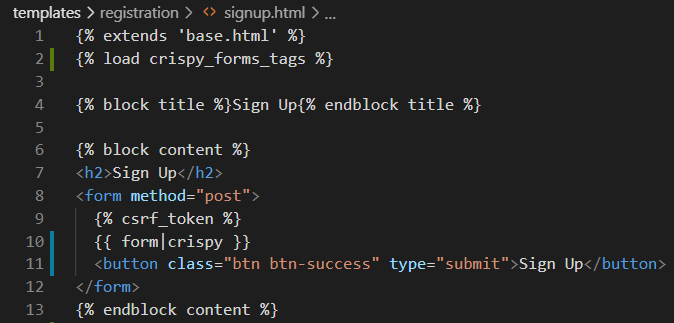


If you start up the server again with python manage.py runserver and refresh the sign

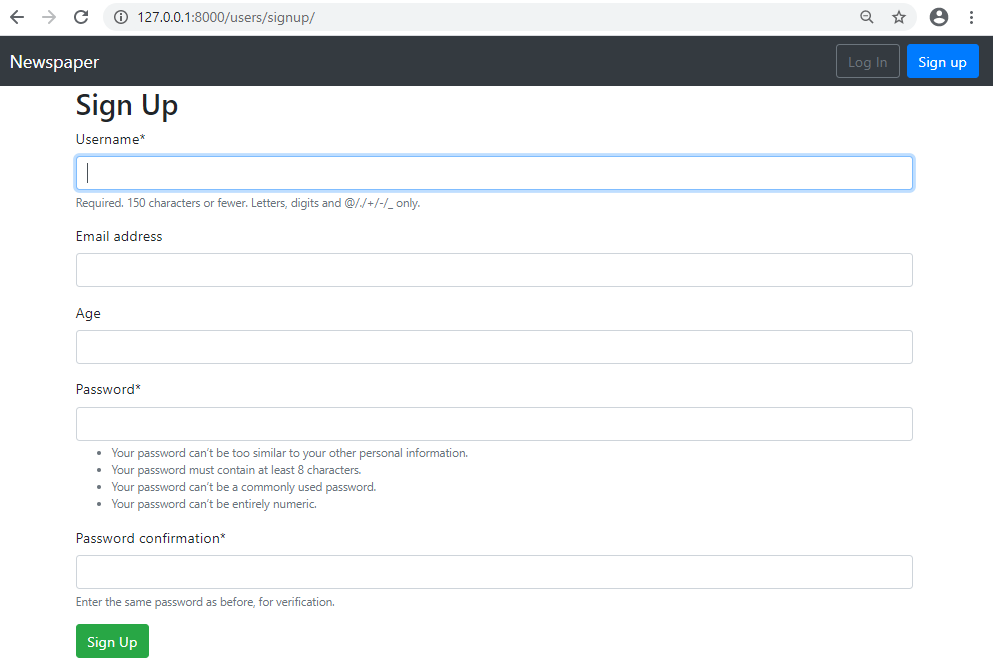
up page we can see the new changes.



The form now looks much better. Next we will change the colour of our “Sign Up” button to make it green? Bootstrap has all sorts of button styling options we can choose from. Let’s use the “success” one which has a green background and white text. Update the signup.html file on the line for the sign up button.



Refresh the page and you can see our updated work.



Run the following git commands to update the local and remote repositories:

(env) djangoprojects\lab5>git add -A

(env) djangoprojects\lab5>git commit -m “lab 5 part 3 commit”

(env) djangoprojects\lab5>git push -u origin master