**Django.**

**NOTE: Always remember to safe all your work on VsCode**

from terminal. pip install django==2.1

Ctrl + L ( to clean up the terminal)

django-admin startproject pyshop . (the dot . means to create one folder)

(wsgi: webserver gateway interface.)

python3 manage.py runserver

django-admin startproject django\_project

getting default browser

python3 manage.py runserver

starting an app

python manage.py startapp blog

in Vscode🡪views.py

from django.http import HttpResponse

create home in 🡪 view.py

def home(request):

return HttpResponse('<h1>Blog Home</h1>')

within the blog folder 🡪 create URLS.PY

copy from url.py in folder into url.py

from django.urls import path

urlpatterns = [

path('admin/', admin.site.urls), /Users/femi/django\_project/blog/views.py

/Users/femi/django\_project/blog/url.py

/Users/femi/django\_project/django\_project/urls.py

]

Then enter into urls.py

from . import views

change the urlpatterns

path('', views.home, name='blog-home'),

include blogs.url in the url.py folder

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('blog/', include('blog.urls')),

]

In views.py enter:

def about(request):

return HttpResponse('<h1>Blog About</h1>')

in urls.py blog enter:

path('about/', views.about, name='blog-about'),

]

CHECK by opening localhost/8000/blog/about.

Also run python3 manage.py runserver on the TERMINAL.

In blog FOLDER 🡪 create Template 🡪, then create a blog Folder inside the templates.

🡪blog.html and about.html file in the template/blog folder

In home.html create

<html>

<head>

<title></title>

</head>

<body>

<h1>Blog home!</h1>

</body>

</html>

Open app.py in blog module, copy 🡪

BlogConfig(

Open settings.py and paste into installed app the copied words (always come here to install app anytime new apps are added to the program)

INSTALLED\_APPS = [

'blog.apps.BlogConfig'

Open blog view, change

def home(request):

return HttpResponse('<h1>Blog Home</h1>')

TO🡪

def home(request):

return render(request, 'blog/home.html')

open about.html and enter

<html>

<head>

<title></title>

</head>

<body>

<h1>About Page</h1>

</body>

</html>

Open views.py and change

def about(request):

return HttpResponse('<h1>Blog About</h1>')

TO🡪

def about(request):

return render(request, 'blog/about.html')

and remove

from django.http import HttpResponse

from view.py since its no longer in use

**DUMMY POST.**

In view. Py

from django.shortcuts import render

posts = [

{

'author': 'CoreyMS',

'title': 'Blog Post 1',

'content': 'First post content',

'date\_posted': 'August 27, 2018'

},

{

'author': 'Jane Doe',

'title': 'Blog Post 2',

'content': 'Second post content',

'date\_posted': 'August 28, 2018'

}

]

def home(request):

context = {

'posts': posts

}

return render(request, 'blog/home.html', context)

def about(request):

return render(request, 'blog/about.html')

In home.html in template/blog

<html>

<head>

<title></title>

</head>

<body>

{% for post in posts %}

<h1>{{ post.title }}</h1>

<p>By {{ post.author }} on {{ post.date\_posted }}</p>

<p>{{ post.content }}</p>

{% endfor %}

</body>

</html>

And about.html

<html>

<head>

{% if title%}

<title> Django Blog - {{ title }} </title>

{% else %}

<title></title>

{% endif %}

</head>

<body>

<h1>About Page</h1>

</body>

</html>

If there is a title for our project

<html>

<head>

{% if title%}

<title> Django Blog - {{ title }}</title>

{% else %}

<title></title>

{% endif %}

</head>

Create file base.html in template/blog. Copy from about and paste into base.html

<html>

<head>

{% if title%}

<title> Django Blog - {{ title }} </title>

{% else %}

<title> Django Blog </title>

{% endif %}

</head>

<body>

{% block content %}{% endblock %}

</body>

</html>

Edit home.html to 🡪

{% extend "blog/base.html" %}

{% block content %}

{% for post in posts %}

<h1>{{ post.title }}</h1>

<p>By {{ post.author }} on

{{ post.date\_posted }}</p>

<p>{{ post.content }}</p>

{% endfor %}

{% endblock content %}

Copy from home.html 🡪about.html and edit to

{% extend "blog/base.html" %}

{% block content %}

<h1>About Page</h1>

{% endblock content %}

🡪 <https://getbootstrap.com/docs/5.1/getting-started/introduction/#starter-template>

Copy the bootstap CSS and java script between the head and body, and paste into base.html and edit

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

{% if title %}

<title>Django Blog - {{ title }}</title>

{% else %}

<title>Femi</title>

{% endif %}

</head>

<body>

<div class="container">

{% block content %}{% endblock %}

</div>

<!-- Optional JavaScript; choose one of the two! -->

<!-- Option 1: Bootstrap Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->

<!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script>

-->

</body>

</html>

Copy from snipet file into base.html

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

{% if title %}

<title>Django Blog - {{ title }}</title>

{% else %}

<title>Femi</title>

{% endif %}

</head>

<body>

</head>

<body>

<header class="site-header">

<nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">

<div class="container">

<a class="navbar-brand mr-4" href="{% url 'blog-home' %}">Django Blog</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false" aria-label="Toggle navigation">

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

</button>

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

<div class="navbar-nav mr-auto">

<a class="nav-item nav-link" href="{% url 'blog-home' %}">Home</a>

<a class="nav-item nav-link" href="{% url 'blog-about' %}">About</a>

</div>

<!-- Navbar Right Side -->

<div class="navbar-nav">

<a class="nav-item nav-link" href="#">Login</a>

<a class="nav-item nav-link" href="#">Register</a>

</div>

</div>

</div>

</nav>

</header>

<main role="main" class="container">

<div class="row">

<div class="col-md-8">

{% block content %}{% endblock %}

</div>

<div class="col-md-4">

<div class="content-section">

<h3>Our Sidebar</h3>

<p class='text-muted'>You can put any information here you'd like.

<ul class="list-group">

<li class="list-group-item list-group-item-light">Latest Posts</li>

<li class="list-group-item list-group-item-light">Announcements</li>

<li class="list-group-item list-group-item-light">Calendars</li>

<li class="list-group-item list-group-item-light">etc</li>

</ul>

</p>

</div>

</div>

</div>

</main>

<!-- Optional JavaScript; choose one of the two! -->

<!-- Option 1: Bootstrap Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->

<!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script>

-->

</body>

</html>

Open a static folded/ blog folder/main.css 🡪 in blog folder. Copy and paste snipet

body {

background: #fafafa;

color: #333333;

margin-top: 5rem;

}

h1, h2, h3, h4, h5, h6 {

color: #444444;

}

ul {

margin: 0;

}

.bg-steel {

background-color: #5f788a;

}

.site-header .navbar-nav .nav-link {

color: #cbd5db;

}

.site-header .navbar-nav .nav-link:hover {

color: #ffffff;

}

.site-header .navbar-nav .nav-link.active {

font-weight: 500;

}

.content-section {

background: #ffffff;

padding: 10px 20px;

border: 1px solid #dddddd;

border-radius: 3px;

margin-bottom: 20px;

}

.article-title {

color: #444444;

}

a.article-title:hover {

color: #428bca;

text-decoration: none;

}

.article-content {

white-space: pre-line;

}

.article-img {

height: 65px;

width: 65px;

margin-right: 16px;

}

.article-metadata {

padding-bottom: 1px;

margin-bottom: 4px;

border-bottom: 1px solid #e3e3e3

}

.article-metadata a:hover {

color: #333;

text-decoration: none;

}

.article-svg {

width: 25px;

height: 25px;

vertical-align: middle;

}

.account-img {

height: 125px;

width: 125px;

margin-right: 20px;

margin-bottom: 16px;

}

.account-heading {

font-size: 2.5rem;

}

Go to 🡪 base.html

{% load static %}

Enter

<link rel="stylesheet" type="text/css" href="{% static 'blog/main.css' %}">

Restart terminal and check web browser.

Open home.html

{% extends "blog/base.html" %}

{% block content %}

{% for post in posts %}

<article class="media content-section">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ post.author }}</a>

<small class="text-muted">{{ post.date\_posted }}</small>

</div>

<h2><a class="article-title" href="#">{{ post.title }}</a></h2>

<p class="article-content">{{ post.content }}</p>

</div>

</article>

{% endfor %}

{% endblock content %}

**Creating a new user (Admin)**

On terminal

python3 manage.py makemigrations

this is to inspect our table.

To create table

python3 manage.py migrate

to create superuser

python3 manage.py createsuperuser

> python manage.py shell

$ from django.contrib.auth.models import User

$ User.objects.get(username="joebloggs", is\_superuser=True).delete()

**Run server on terminal and also start localhost:8000/admin. Enter username and password**

Building database using ORM(Object relation map)

Open model.py

from django.db import models

from django.utils import timezone

from django.contrib.auth.models import User

class Post(models.Model):

title = models.CharField(max\_length =100)

content = models.TextField()

date\_posted = models.DateTimeField(default=timezone.now)

author = models.ForeignKey(User, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.title

on terminal:

**python3 manage.py makemigrations:** run this on the terminal to see the data base created.

**python3 manage.py sqlmigrate blog 0001:** to see the created model post and tables.

**python3 manage.py migrate:** to make changes to our database.

**python3 manage.py shell:** to work with models interactively line by line

1.**from** **blog.models** **import** Post

2.**from** **django.contrib.auth.models** **import** User

(User.objects.all()

Out[**4**]: <QuerySet [<User: femi>]>) this create users already created in the database

User.objects.filter(username=’CoreyMS’) : command use to get the user by name

3. user = User.objects.filter(username='femi').first()

:putting the line in a variable user.

4. post\_1 = Post(title='Blog 1', content='First Post Content!', author=user)

: to make a post by the user

post\_2.save()

On model.py

def \_\_str\_\_(self):

return self.title

to return the username on queries

Next🡪 on terminal

python3 manage.py shell

In [4]: exit

(base) femi@Femis-MBP django\_project % python3 manage.py shell

Python 3.8.8 (default, Apr 13 2021, 12:59:45)

Type 'copyright', 'credits' or 'license' for more information

IPython 7.22.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from blog.models import Post

In [2]: from django.contrib.auth.models import User

In [3]: Post.objects.all()

Out[3]: <QuerySet [<Post: Blog 1>]>

And also renter **user = User.objects.filter(username='femi').first()** on the terminal

post🡪

🡪 post\_2 = Post(title='Blog 2', content='Second Post Content', author\_id=u

🡪 ...: ser.id)

post\_2.save()

🡪 turning the first post to a variable

post = Post.objects.first()

to get the date post was made

post.date\_posted

to see user object

post.author

to sse also get the email

post.author.email

to see what a user has creaated

user.post\_set

to see all post a user authored

user.post\_set.all()

creating another post by the same user

user.post\_set.create(title='Blog 3', content='Third Post Content!')

to import the new post into our site 🡪 VsCode 🡪 view.py

add and redirect the new post

from .models import Post

from

def home(request):

context = {

'posts': posts

}

return render(request, 'blog/home.html', context)

To

def home(request):

context = {

'posts': Post.objects.all()

}

return render(request, 'blog/home.html', context)

rerun server

python manage.py runserver

changing the time to month/day/year.

in template/blog/home.html add

<small class="text-muted">{{ post.date\_posted|date:"F d, Y" }}</small>

Registering our post on the admin site

Open admin.py

from django.contrib import admin

from .models import Post

admin.site.register(Post)

to show all migrations

python3 manage.py showmigrations

{ no such table: main.auth\_user\_\_old 🡪

-----------------------------------🡪

Solution:

pip install Django --upgrade

Delete the SQLite db,

Migrate again python manage.py makemigrations and then python manage.py migrate

Start the server python manage.py runserver }

Creating user profile:

python3 manage.py startapp users

add the app in the settings.py

INSTALLED\_APPS = [

'blog.apps.BlogConfig',

'users.apps.UsersConfig',

Creating views for users

Users/views.py

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

def register(request):

form = UserCreationForm()

return render(request, 'users/register.html', {'form': form})

open a new templates folder/users folder/register files

{% extends "blog/base.html" %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4"> Join Today</legend>

{{ form }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info" type="submit">Sign Up</button>

</div>

</form>

<div class="border-top pt-3">

<small class="text-muted">

Already Have An Account? <a class="ml-2" href="#">Sign In</a>

</small>

</div>

</div>

{% endblock content %}

In Django\_project/urls.py 🡪

from django.contrib import admin

from django.urls import path, include

from users import views as user\_views

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('', include('blog.urls')),

]

🡪templates/users/register

from django.shortcuts import render, redirect

from django.contrib.auth.forms import UserCreationForm

from django.contrib import messages

def register(request):

if request.method == 'POST':

form = UserCreationForm(request.POST)

if form.is\_valid():

form.save()

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

messages.success(request, f' Account created for {username}!')

return redirect('blog-home')

else:

form = UserCreationForm()

return render(request, 'users/register.html', {'form': form})

go to base.py to include the changes on our page

<main role="main" class="container">

<div class="row">

<div class="col-md-8">

{% if messages %}

{% for message in messages %}

<div class="alert alert-{{ message.tags }}">

{{ message }}

</div>

{% endfor %}

{% endif %}

{% block content %}{% endblock %}

</div>

<div class="col-md-4">

<div class="content-section">

<h3>Our Sidebar</h3>

<p class='text-muted'>You can put any information here you'd like.

<ul class="list-group">

<li class="list-group-item list-group-item-light">Latest Posts</li>

<li class="list-group-item list-group-item-light">Announcements</li>

<li class="list-group-item list-group-item-light">Calendars</li>

<li class="list-group-item list-group-item-light">etc</li>

</ul>

</p>

</div>

</div>

</div>

</main>

Runserver on terminal and got to home/register. Enter new user to make sure its created.

Adding new field to user creation forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from django.forms import fields

class UserRegisterForm(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

🡪views.py: import the forms.py

from django.shortcuts import render, redirect

from django.contrib import messages

from .forms import UserRegisterForm

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

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

messages.success(request, f' Account created for {username}!')

return redirect('blog-home')

else:

form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

on terminal:

install pip install Django-crispy-forms

crispy forms app: to give better class to the user page. Giving a tag in a bootstrap faction

to tell Django its an install app 🡪 open settings.py on vscode

INSTALLED\_APPS = [

'blog.apps.BlogConfig',

'users.apps.UsersConfig',

'crispy\_forms',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

Also add 🡪 at the bottom

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

Load into 🡪 reqister.py

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

Also change the form in register.py

<fieldset class="form-group">

<legend class="border-bottom mb-4"> Join Today</legend>

{{ form|crispy }}

</fieldset>

Creating default user page

🡪 Django\_project/urls.py

from django.contrib import admin

from django.contrib.auth import views as auth\_views

from django.urls import path, include

from users import views as user\_views

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='users/logout.html'), name='logout'),

path('', include('blog.urls')),

]

Create path in templates/users/login.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4"> Log In</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

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

</div>

</form>

<div class="border-top pt-3">

<small class="text-muted">

Need An Account? <a class="ml-2" href="{% url 'register' %}">Sign Up Now</a>

</small>

</div>

</div>

{% endblock content %}

Open settings.py add

LOGIN\_REDIRECT\_URL = 'blog-home'

🡪 users/views.py(change success message)

messages.success(request, f'Your account has been created! You are now able to log in {username}!')

return redirect('login')

🡪 Django\_project/urls.py

from django.contrib import admin

from django.contrib.auth import views as auth\_views

from django.urls import path, include

from users import views as user\_views

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='users/logout.html'), name='logout'),

path('', include('blog.urls')),

]

Create templates/users/logout.html

{% extends "blog/base.html" %}

{% block content %}

<h2>You have been logged out</h2>

<div class="border-top pt-3">

<small class="text-muted">

<a href="{% url 'login' %}">Log In Again</a>

</small>

</div>

{% endblock content %}

To add logout page.

🡪 base.html

<!-- Navbar Right Side -->

<div class="navbar-nav ">

{% if user.is\_authenticated %}

<a class="nav-item nav-link" href="{% url 'profile' %}">Profile</a>

<a class="nav-item nav-link" href="{% url 'logout' %}">Logout</a>

{% else %}

<a class="nav-item nav-link" href="{% url 'login' %}">Login</a>

<a class="nav-item nav-link" href="{% url 'register' %}">Register</a>

{% endif %}

</div>

Creating a page for users profile

🡪 users/view.py

from django.shortcuts import render, redirect

from django.contrib import messages

from .forms import UserRegisterForm

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

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

messages.success(request, f'Your account has been created! You are now able to log in {username}!')

return redirect('login')

else:

form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

def profile(request):

return render(request, 'users/profile.html')

to create profile templates.

Templates/users/profile.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<h1>{{ user.username }}</h1>

{% endblock content %}

Include in in our project url.py

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('profile/', user\_views.profile, name='profile'),

restricting user to log in before view profile

user/views.py

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from .forms import UserRegisterForm

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

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

messages.success(request, f'Your account has been created! You are now able to log in {username}!')

return redirect('login')

else:

form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

@login\_required

def profile(request):

return render(request, 'users/profile.html')

🡪 settings.py add

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC\_URL = '/static/'

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

LOGIN\_REDIRECT\_URL = 'blog-home'

LOGIN\_URL = 'login'

Adding profile picture

🡪 users/models.py

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

image = models.ImageField(default='default.jpg', upload\_to='profile\_pics')

def \_\_str\_\_(self):

return f'{self.user.username} Profile'

whenever we make changes to our database, we must run migration on terminal

run: pip install Pillow

python3 manage.py makemigrations

python3 manage.py migrate

🡪registering with the admin page: 🡪users/admin.py

from django.contrib import admin

from .models import Profile

admin.site.register(Profile)

searching for user in python shell:

run python manage.py shell

**from** **django.contrib.auth.models** **import** User

user =User.onjects.filter(username=’TestUser’).first()

to check user profile image:

user.profile.image

moving our profile pics to another location and easy access

🡪project/settings.py

STATIC\_URL = '/static/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

LOGIN\_REDIRECT\_URL = 'blog-home'

LOGIN\_URL = 'login'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

Add media.

New media is created in our vscode page and project folder. The old profile\_pics can be deleted.

Displaying all image in templates.

Users/template/profiles.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<div class="media">

<img class="rounded-circle account-img" src="{{ user.profile.image.url }}">

<div class="media-body">

<h2 class="account-heading">{{ user.username }}</h2>

<p class="text-secondary">{{ user.email }}</p>

</div>

</div>

<!-- FORM HERE -->

</div>

{% endblock content %}

🡪project/url.py

from django.contrib import admin

from django.contrib.auth import views as auth\_views

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

from users import views as user\_views

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('profile/', user\_views.profile, name='profile'),

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='users/logout.html'), name='logout'),

path('', include('blog.urls')),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

🡪media

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.dispatch import receiver

from .models import Profile

@receiver(post\_save, sender=User)

def create\_profile(sender, instance, created, \*\*kwargs):

if created:

Profile.objects.create(user=instance)

@receiver(post\_save, sender=User)

def save\_profile(sender, instance, \*\*kwargs):

instance.profile.save()

🡪users/apps.py

from django.apps import AppConfig

class UsersConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'users'

def ready(self):

import users.signals

creating update:

users/forms.py

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from .models import Profile

class UserRegisterForm(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

class UserUpdateForm(forms.ModelForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email']

class ProfileUpdateForm(forms.ModelForm):

class Meta:

model = Profile

field = ['image']

🡪 user/views.py

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

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

messages.success(request, f'Your account has been created! You are now able to log in {username}!')

return redirect('login')

else:

form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

@login\_required

def profile(request):

if request.method == 'POST':

u\_form = UserUpdateForm(request.POST, instance=request.user)

p\_form = ProfileUpdateForm(request.POST, request.FILES, instance=request.user.profile)

if u\_form.is\_valid() and p\_form.is\_valid():

u\_form.save()

p\_form.save()

messages.success(request, f'Your account has been update!')

return redirect('profile')

else:

u\_form = UserUpdateForm(instance=request.user)

p\_form = ProfileUpdateForm(instance=request.user.profile)

context = {

'u\_form': u\_form,

'p\_form': p\_form

}

return render(request, 'users/profile.html', context)

🡪user/profile.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<div class="media">

<img class="rounded-circle account-img" src="{{ user.profile.image.url }}">

<div class="media-body">

<h2 class="account-heading">{{ user.username }}</h2>

<p class="text-secondary">{{ user.email }}</p>

</div>

</div>

<form method="POST" enctype="multipart/form-data">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Profile Info</legend>

{{ u\_form|crispy }}

{{ p\_form|crispy }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info" type="submit">Update</button>

</div>

</form>

</div>

{% endblock content %}

🡪 user/forms.py

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from .models import Profile

class UserRegisterForm(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

class UserUpdateForm(forms.ModelForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email']

class ProfileUpdateForm(forms.ModelForm):

class Meta:

model = Profile

fields = ['image']

🡪profile mode to set image size. User/models.py

from django.db import models

from django.contrib.auth.models import User

from PIL import Image

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

image = models.ImageField(default='default.jpg', upload\_to='profile\_pics')

phone\_number = models.IntegerField(blank=True, null=True)

def \_\_str\_\_(self):

return f'{self.user.username} Profile'

def save(self):

super().save()

img = Image.open(self.image.path)

if img.height > 300 or img.width > 300:

output\_size = (300, 300)

img.thumbnail(output\_size)

img.save(self.image.path)

🡪blog/home.html

{% extends "blog/base.html" %}

{% block content %}

{% for post in posts %}

<article class="media content-section">

<img class="rounded-circle article-img" src="{{ post.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ post.author }}</a>

<small class="text-muted">{{ post.date\_posted|date:"F d, Y" }}</small>

</div>

<h2><a class="article-title" href="#">{{ post.title }}</a></h2>

<p class="article-content">{{ post.content }}</p>

</div>

</article>

{% endfor %}

{% endblock content %}

Displaying, delete and updating new post

Blog/views.py

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):

model = Post

success\_url = '/'

def test\_func(self):

post = self.get\_object()

if self.request.user == post.author:

return True

return False

def about(request):

return render(request, 'blog/about.html', {'title': 'About'})

url.py

from django.urls import path

from .views import (

PostListView,

PostDetailView,

PostCreateView,

PostUpdateView,

PostDeleteView

)

from . import views

urlpatterns = [

path('', PostListView.as\_view(), name='blog-home'),

path('post/<int:pk>/', PostDetailView.as\_view(), name='post-detail'),

path('post/new/', PostCreateView.as\_view(), name='post-create'),

path('post/<int:pk>/update/', PostUpdateView.as\_view(), name='post-update'),

path('post/<int:pk>/delete/', PostDeleteView.as\_view(), name='post-delete'),

path('about/', views.about, name='blog-about'),

]

🡪templates/Blog/post\_Detail.html

{% block content %}

<article class="media content-section">

<img class="rounded-circle article-img" src="{{ object.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ object.author }}</a>

<small class="text-muted">{{ object.date\_posted|date:"F d, Y" }}</small>

{% if object.author == user %}

<div>

<a class="btn btn-secondary btn-sm mt-1 mb-1" href="{% url 'post-update' object.id %}">Update</a>

<a class="btn btn-danger btn-sm mt-1 mb-1" href="{% url 'post-delete' object.id %}">Delete</a>

</div>

{% endif %}

</div>

<h2 class="article-title">{{ object.title }}</h2>

<p class="article-content">{{ object.content }}</p>

</div>

</article>

{% endblock content %}

Updating links for post (using class base views)

🡪templates/blog/home.html

{% extends "blog/base.html" %}

{% block content %}

{% for post in posts %}

<article class="media content-section">

<img class="rounded-circle article-img" src="{{ post.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ post.author }}</a>

<small class="text-muted">{{ post.date\_posted|date:"F d, Y" }}</small>

</div>

<h2><a class="article-title" href="{% url 'post-detail' post.id %}">{{ post.title }}</a></h2>

<p class="article-content">{{ post.content }}</p>

</div>

</article>

{% endfor %}

{% endblock content %}

🡪templates/blog/post\_form.html (using class base views)

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4"> Blog Post</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info" type="submit">Post</button>

</div>

</form>

</div>

{% endblock content %}

🡪blog/models.py (using class base views)

from django.db import models

from django.utils import timezone

from django.contrib.auth.models import User

from django.urls import reverse

class Post(models.Model):

title = models.CharField(max\_length=100)

content = models.TextField()

date\_posted = models.DateTimeField(default=timezone.now)

author = models.ForeignKey(User, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.title

def get\_absolute\_url(self):

return reverse('post-detail', kwargs={'pk': self.pk})

Templates to confirm delete request

🡪blog/post\_confrim\_delete.html

{% extends "blog/base.html" %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Delete Post</legend>

<h2>Are You Sure You Want To Delete The Post {{ object.title }}</h2>

</fieldset>

<div class="form-group">

<button class="btn btn-outline-danger" type="submit">Yes, Delete</button>

<a class="btn btn-outline-secondary" href="{% url 'post-detail' object.id %}">Cancel</a>

</div>

</form>

</div>

{% endblock content %}

Adding links to the navigation bar update/delete and new post

Blog/templates/blog/base.html

{% load static %}

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<link rel="stylesheet" type="text/css" href="{% static 'blog/main.css' %}">

{% if title %}

<title>Django Blog - {{ title }}</title>

{% else %}

<title>Femi</title>

{% endif %}

</head>

<body>

<header class="site-header">

<nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">

<div class="container">

<a class="navbar-brand mr-4" href="{% url 'blog-home' %}">Django Blog</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false" aria-label="Toggle navigation">

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

</button>

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

<div class="navbar-nav mr-auto">

<a class="nav-item nav-link" href="{% url 'blog-home' %}">Home</a>

<a class="nav-item nav-link" href="{% url 'blog-about' %}">About</a>

</div>

<!-- Navbar Right Side -->

<div class="navbar-nav ">

{% if user.is\_authenticated %}

<a class="nav-item nav-link" href="{% url 'post-create' %}">New Post</a>

<a class="nav-item nav-link" href="{% url 'profile' %}">Profile</a>

<a class="nav-item nav-link" href="{% url 'logout' %}">Logout</a>

{% else %}

<a class="nav-item nav-link" href="{% url 'login' %}">Login</a>

<a class="nav-item nav-link" href="{% url 'register' %}">Register</a>

{% endif %}

</div>

</div>

</div>

</nav>

</header>

<main role="main" class="container">

<div class="row">

<div class="col-md-8">

{% if messages %}

{% for message in messages %}

<div class="alert alert-{{ message.tags }}">

{{ message }}

</div>

{% endfor %}

{% endif %}

{% block content %}{% endblock %}

</div>

<div class="col-md-4">

<div class="content-section">

<h3>Our Sidebar</h3>

<p class='text-muted'>You can put any information here you'd like.

<ul class="list-group">

<li class="list-group-item list-group-item-light">Latest Posts</li>

<li class="list-group-item list-group-item-light">Announcements</li>

<li class="list-group-item list-group-item-light">Calendars</li>

<li class="list-group-item list-group-item-light">etc</li>

</ul>

</p>

</div>

</div>

</div>

</main>

<!-- Optional JavaScript; choose one of the two! -->

<!-- Option 1: Bootstrap Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->

<!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script>

-->

</body>

</html>

Pagination : working with pagination class

🡪Django shell

**🡪from** **django.core.paginator** **import** Paginator

sample post: Paginator objects

🡪posts = ['1','2','3','4','5']

Making two post per page

🡪p = Paginator(posts, 2)

Knowing number of pages

🡪p.num\_pages

Looping over a page

🡪**for** page **in** p.page\_range:

...: print(page)

Looking at a specific page

🡪p1 = p.page(1)

🡪p1

We can get page as a number

🡪p1.number

Looking at post on a page

🡪p1.object\_list

Checking previous page

🡪p1.has\_previous()

To get if page has a next page

🡪p1.has\_next()

To get next page as a number

🡪P1.next\_page\_number()

Importing json thru Django.shell

🡪 **import** **json**

**🡪 from** **blog.models** **import** Post

🡪 **with** open('posts.json') **as** f:

...: posts\_json = json.load(f)

🡪 **for** post **in** posts\_json:

post = Post(title=post['title'], content=post['content'], author\_id=po

...: st['user\_id'])

...: post.save()

Including paginator in our projects--vxcode

🡪blog/views.py

from django.shortcuts import render, get\_object\_or\_404

from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin

from django.contrib.auth.models import User

from django.views.generic import (

ListView,

DetailView,

CreateView,

UpdateView,

DeleteView,

)

from .models import Post

def home(request):

context = {

'posts': Post.objects.all()

}

return render(request, 'blog/home.html', context)

class PostListView(ListView):

model = Post

template\_name = 'blog/home.html' # <app>/<model>\_<viewtype>.html

context\_object\_name = 'posts'

ordering = ['-date\_posted']

paginate\_by = 5 # number of pages

class UserPostListView(ListView):

model = Post

template\_name = 'blog/user\_posts.html' # <app>/<model>\_<viewtype>.html

context\_object\_name = 'posts'

paginate\_by = 5

def get\_queryset(self):

user = get\_object\_or\_404(User, username=self.kwargs.get('username'))

return Post.objects.filter(author=user).order\_by('-date\_posted')

class PostDetailView(DetailView):

model = Post

class PostCreateView(LoginRequiredMixin, CreateView):

model = Post

fields = ['title', 'content']

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):

model = Post

fields = ['title', 'content']

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

def test\_func(self):

post = self.get\_object()

if self.request.user == post.author:

return True

return False

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):

model = Post

success\_url = '/'

def test\_func(self):

post = self.get\_object()

if self.request.user == post.author:

return True

return False

def about(request):

return render(request, 'blog/about.html', {'title': 'About'})

🡪home.html

{% endfor %}

{% if is\_paginated %}

{% if page\_obj.has\_previous %}

<a class="btn btn-outline-info mb-4" href="?page=1">First</a>

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.previous\_page\_number }}">Previous</a>

{% endif %}

{% for num in page\_obj.paginator.page\_range %}

{% if page\_obj.number == num %}

<a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>

{% elif num > page\_obj.number|add:'-3' and num < page\_obj.number|add:'3' %}

<a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>

{% endif %}

{% endfor %}

{% if page\_obj.has\_next %}

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.next\_page\_number }}">Next</a>

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.paginator.num\_pages }}">Last</a>

{% endif %}

{% endif %}

{% endblock content %}

Blog/url.py

from django.urls import path

from .views import (

PostListView,

PostDetailView,

PostCreateView,

PostUpdateView,

PostDeleteView,

UserPostListView

)

from . import views

urlpatterns = [

path('', PostListView.as\_view(), name='blog-home'),

path('user/<str:username>', UserPostListView.as\_view(), name='user-posts'),

path('post/<int:pk>/', PostDetailView.as\_view(), name='post-detail'),

path('post/new/', PostCreateView.as\_view(), name='post-create'),

path('post/<int:pk>/update/', PostUpdateView.as\_view(), name='post-update'),

path('post/<int:pk>/delete/', PostDeleteView.as\_view(), name='post-delete'),

path('about/', views.about, name='blog-about'),

]

Create new templates:

Blog/user\_posts.html

{% extends "blog/base.html" %}

{% block content %}

<h1 class="mb-3">Posts by {{ view.kwargs.username }} ({{ page\_obj.paginator.count }})</h1>

{% for post in posts %}

<article class="media content-section">

<img class="rounded-circle article-img" src="{{ post.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="{% url 'user-posts' post.author.username %}">{{ post.author }}</a>

<small class="text-muted">{{ post.date\_posted|date:"F d, Y" }}</small>

</div>

<h2><a class="article-title" href="{% url 'post-detail' post.id %}">{{ post.title }}</a></h2>

<p class="article-content">{{ post.content }}</p>

</div>

</article>

{% endfor %}

{% if is\_paginated %}

{% if page\_obj.has\_previous %}

<a class="btn btn-outline-info mb-4" href="?page=1">First</a>

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.previous\_page\_number }}">Previous</a>

{% endif %}

{% for num in page\_obj.paginator.page\_range %}

{% if page\_obj.number == num %}

<a class="btn btn-info mb-4" href="?page={{ num }}">{{ num }}</a>

{% elif num > page\_obj.number|add:'-3' and num < page\_obj.number|add:'3' %}

<a class="btn btn-outline-info mb-4" href="?page={{ num }}">{{ num }}</a>

{% endif %}

{% endfor %}

{% if page\_obj.has\_next %}

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.next\_page\_number }}">Next</a>

<a class="btn btn-outline-info mb-4" href="?page={{ page\_obj.paginator.num\_pages }}">Last</a>

{% endif %}

{% endif %}

{% endblock content %}

Also change the author route in the following:

Home.html

<a class="mr-2" href="{% url 'user-posts' post.author.username %}">{{ post.author }}</a>

And

Post\_detail.html

<a class="mr-2" href="url 'user-posts' object.author.username ">{{ object.author }}</a>

Resetting password and notification for password request:

🡪project/url.py

from django.contrib import admin

from django.contrib.auth import views as auth\_views

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

from users import views as user\_views

urlpatterns = [

path('admin/', admin.site.urls),

path('register/', user\_views.register, name='register'),

path('profile/', user\_views.profile, name='profile'),

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='users/logout.html'), name='logout'),

path('password-reset/',

auth\_views.PasswordResetView.as\_view(

template\_name='users/password\_reset.html'

),

name='password\_reset'),

path('password-reset/done/',

auth\_views.PasswordResetDoneView.as\_view(

template\_name='users/password\_reset\_done.html'

),

name='password\_reset\_done'),

path('password-reset-confirm/<uidb64>/<token>/',

auth\_views.PasswordResetConfirmView.as\_view(

template\_name='users/password\_reset\_confirm.html'

),

name='password\_reset\_confirm'),

path('password-reset-complete/',

auth\_views.PasswordResetCompleteView.as\_view(

template\_name='users/password\_reset\_complete.html'

),

name='password\_reset\_complete'),

path('', include('blog.urls')),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

]

🡪user/password\_reset\_complete.html

{% extends "blog/base.html" %}

{% block content %}

<div class="alert alert-info">

Your password has been set.

</div>

<a href="{% url 'login' %}">Sign In Here</a>

{% endblock content %}

🡪user/password\_reset\_confirm.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Reset Password</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info" type="submit">Reset Password</button>

</div>

</form>

</div>

{% endblock content %}

🡪user/password\_reset\_done.html

{% extends "blog/base.html" %}

{% block content %}

<div class="alert alert-info">

An email has been sent with instructions to reset your password

</div>

{% endblock content %}

🡪user/password\_reset.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Reset Password</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info" type="submit">Request Password Reset</button>

</div>

</form>

</div>

{% endblock content %}

🡪settings.py (to add email password)

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_HOST\_USER = os.environ.get('EMAIL\_USER')

EMAIL\_HOST\_PASSWORD = os.environ.get('EMAIL\_PASS')

Login template/login.html

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4"> Log In</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

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

<small class="text-muted ml-2">

<a href="{% url 'password\_reset' %}">Forgot Password?</a>

</small>

</div>

</form>

<div class="border-top pt-3">

<small class="text-muted">

Need An Account? <a class="ml-2" href="{% url 'register' %}">Sign Up Now</a>

</small>

</div>

</div>

{% endblock content %}