DIRECTORY H:\ZZZ.python PROJECTS\2. thug coding blog\ThugCodingBlog  
  
style.css  
/\* styles.css \*/

/\* Overlay background \*/

.img-lightbox-overlay {

  position: fixed;

  top: 0;

  left: 0;

  width: 100vw;

  height: 100vh;

  background-color: rgba(0,0,0,0.85);

  display: none;

  justify-content: center;

  align-items: center;

  z-index: 9999;

}

/\* Fullscreen image \*/

.img-lightbox-overlay img {

  max-width: 90vw;

  max-height: 90vh;

  border-radius: 8px;

  box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);

  cursor: zoom-out;

}

/\* Clearfix to prevent layout issues from floating images \*/

.card::after {

  content: "";

  display: table;

  clear: both;

}

.blog-post-img {

  max-width: 50%;  /\* Adjust as needed \*/

}

/\* Ensure all buttons use the same blue color \*/

.btn-primary,

.btn-success,

.btn-secondary,

.btn-link {

    background-color: #007BFF !important;

    border-color: #007BFF !important;

    color: white !important;

}

/\* Hover effect for buttons \*/

.btn-primary:hover,

.btn-success:hover,

.btn-secondary:hover,

.btn-link:hover {

    background-color: #0069D9 !important;

    border-color: #0062CC !important;

    color: white !important;

}

/\* Remove underline from btn-link on hover \*/

.btn-link:hover {

    text-decoration: none !important;

}

  .card-body h5.card-title a {

    color: #333;

    text-decoration: none;

  }

  .card-body h5.card-title a:hover {

    text-decoration: underline;

  }

  .card-text {

    color: #555;

  }

.card {

    padding: 0;

    border: 1px solid #ddd;

    border-radius: 10px;

    overflow: hidden;

}

.card-body {

    padding: 1rem;

}

  .btn-primary {

    background-color: #007bff;

    border-color: #007bff;

    transition: all 0.2s ease-in-out;

  }

  .btn-primary:hover {

    background-color: #0056b3;

    border-color: #004085;

  }

ThugCodingBlog\blogproject\blog\templates\blog\base.html  
  
<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title %}Thug Coding® Blog{% endblock %}</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body>

    <!-- Navbar -->

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

        <a class="navbar-brand" href="{% url 'home' %}">Thug Coding® Blog</a>

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

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

        </button>

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

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

                <li class="nav-item">

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

                </li>

                {% if user.is\_authenticated %}

                    <li class="nav-item">

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

                    </li>

                {% else %}

                    <li class="nav-item">

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

                    </li>

                {% endif %}

            </ul>

        </div>

    </nav>

    <!-- Page Content -->

    <div class="container mt-4">

        {% block content %}

        {% endblock %}

    </div>

    <!-- Bootstrap JS -->

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

    <!-- Lightbox overlay -->

    <div class="img-lightbox-overlay" id="lightbox" onclick="this.style.display='none'">

        <img id="lightbox-img" src="" alt="fullscreen">

    </div>

    <!-- Lightbox script -->

    <script>

      document.querySelectorAll('.blog-post-img').forEach(img => {

        img.addEventListener('click', () => {

          const lightbox = document.getElementById('lightbox');

          const lightboxImg = document.getElementById('lightbox-img');

          lightboxImg.src = img.src;

          lightbox.style.display = 'flex';

        });

      });

    </script>

</body>

</html>

ThugCodingBlog\blogproject\blog\templates\blog\file\_list.html

<!DOCTYPE html>

<html>

<head>

    <title>Uploaded Files</title>

</head>

<body>

    <h1>Uploaded Files</h1>

    <ul>

        {% for file in files %}

        <li>

            <a href="{{ file.file.url }}">{{ file.title }}</a>

        </li>

        {% endfor %}

    </ul>

    <a href="{% url 'upload\_file' %}">Upload New File</a>

</body>

</html>

ThugCodingBlog\blogproject\blog\templates\blog\home.html

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

{% block title %}Home{% endblock %}

{% block content %}

<h1 class="my-4">Blog Home</h1>

{% for post in posts %}

<div class="card mb-4 p-3">

    {% if post.image %}

        <img src="{{ post.image.url }}" class="blog-post-img" alt="{{ post.title }}">

    {% endif %}

    <div class="card-body">

        <h5 class="card-title">

            <a href="{% url 'post\_detail' post.pk %}">{{ post.title }}</a>

        </h5>

        <p class="card-text">{{ post.content|truncatewords:30 }}</p>

        <a href="{% url 'post\_detail' post.pk %}" class="btn btn-primary">Read More →</a>

    </div>

</div>

{% endfor %}

<h2 class="my-4">Uploaded Files</h2>

<ul>

    {% for file in files %}

        <li><a href="{{ file.file.url }}">{{ file.title }}</a></li>

    {% endfor %}

</ul>

{% endblock %}

ThugCodingBlog\blogproject\blog\templates\blog\login.html

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

{% block title %}Login{% endblock %}

{% block content %}

    <h2>Login</h2>

    <form method="post">

        {% csrf\_token %}

        {{ form.as\_p }}

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

    </form>

{% endblock %}

ThugCodingBlog\blogproject\blog\templates\blog\post\_detail.html

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

{% block title %}{{ post.title }}{% endblock %}

{% block content %}

<div class="container mt-4">

    <div class="card mb-4 p-3">

        {% if post.image %}

            <img src="{{ post.image.url }}" class="blog-post-img" alt="{{ post.title }}">

        {% endif %}

        <div class="card-body">

            <h1 class="card-title">{{ post.title }}</h1>

            <p><strong>Author:</strong> {{ post.author }}</p>

            <p><strong>Created:</strong> {{ post.created\_at }}</p>

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

            <a href="{% url 'home' %}" class="btn btn-primary">Back to Home</a>

        </div>

    </div>

    <hr>

    <h3>Comments</h3>

    {% for comment in comments %}

        <div class="mb-3">

            <strong>{{ comment.author }}</strong>

            <small class="text-muted">{{ comment.created\_at }}</small>

            <p>{{ comment.text }}</p>

        </div>

    {% empty %}

        <p>No comments yet.</p>

    {% endfor %}

    {% if user.is\_authenticated %}

        <hr>

        <h4>Leave a comment</h4>

        <form method="post">

            {% csrf\_token %}

            {{ form.as\_p }}

            <button type="submit" class="btn btn-primary">Post Comment</button>

        </form>

    {% else %}

        <p><a href="{% url 'login' %}">Log in</a> to post a comment.</p>

    {% endif %}

</div>

{% endblock %}

ThugCodingBlog\blogproject\blog\templates\blog\post\_form.html

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

{% block title %}{{ form.instance.pk|default:"New" }} Post{% endblock %}

{% block content %}

<h1>{{ form.instance.pk|default:"New" }} Post</h1>

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

    {% csrf\_token %}

    {{ form.as\_p }}

    <button type="submit" class="btn btn-success">Save</button>

</form>

<a href="{% url 'home' %}" class="btn btn-link">Back to home</a>

{% endblock %}

ThugCodingBlog\blogproject\blog\templates\blog\register.html

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

{% block title %}Sign Up{% endblock %}

{% block content %}

  <h2>Sign Up</h2>

  <form method="post">

    {% csrf\_token %}

    {{ form.as\_p }}

    <button type="submit" class="btn btn-success">Register</button>

  </form>

  <p class="mt-3">Already have an account? <a href="{% url 'login' %}">Login</a></p>

{% endblock %}

ThugCodingBlog\blogproject\blog\templates\blog\upload.html

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

{% block title %}{{ form.instance.pk|default:"New" }} Post{% endblock %}

{% block content %}

<h1>{{ form.instance.pk|default:"New" }} Post</h1>

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

    {% csrf\_token %}

    {{ form.as\_p }}

    <button type="submit" class="btn btn-success">Save</button>

</form>

<a href="{% url 'home' %}" class="btn btn-link">Back to home</a>

{% endblock %}

ThugCodingBlog\blogproject\blog\admin.py

from django.contrib import admin

from .models import Post, UploadedFile, Comment

# Enhanced display for comments in admin

@admin.register(Comment)

class CommentAdmin(admin.ModelAdmin):

    list\_display = ('author', 'post', 'text', 'created\_at')

    search\_fields = ('text', 'author\_\_username', 'post\_\_title')

    list\_filter = ('created\_at', 'author')

# Register other models as usual

admin.site.register(Post)

admin.site.register(UploadedFile)

ThugCodingBlog\blogproject\blog\apps.py

from django.apps import AppConfig

class BlogConfig(AppConfig):

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

    name = 'blog'

ThugCodingBlog\blogproject\blog\forms.py

from django import forms

from .models import Post, UploadedFile, Comment

class PostForm(forms.ModelForm):

    class Meta:

        model = Post

        fields = ['title', 'author', 'content', 'image']

class UploadFileForm(forms.ModelForm):

    class Meta:

        model = UploadedFile

        fields = ['title', 'file']

class CommentForm(forms.ModelForm):

    class Meta:

        model = Comment

        fields = ['text']

ThugCodingBlog\blogproject\blog\models.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=200)

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

    content = models.TextField()

    created\_at = models.DateTimeField(default=timezone.now)

    updated\_at = models.DateTimeField(auto\_now=True)

    image = models.ImageField(upload\_to='images/', blank=True, null=True)

    def \_\_str\_\_(self):

        return self.title

class UploadedFile(models.Model):

    title = models.CharField(max\_length=255)

    file = models.FileField(upload\_to='uploads/')

    uploaded\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return self.title

# Create your models here.

class Comment(models.Model):

    post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='comments')

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

    text = models.TextField()

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"{self.author.username} on {self.post.title}"

ThugCodingBlog\blogproject\blog\tests.py  
  
from django.test import TestCase

# Create your tests here.

ThugCodingBlog\blogproject\blog\urls.py

from django.urls import path

from . import views

urlpatterns = [

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

    path('post/<int:pk>/', views.post\_detail, name='post\_detail'),

    path('post/new/', views.create\_post, name='create\_post'),

    path('post/<int:pk>/edit/', views.edit\_post, name='edit\_post'),

    path('logout/', views.logout\_view, name='logout'),

    path('upload/', views.upload\_file, name='upload\_file'),

    path('files/', views.file\_list, name='file\_list'),

    path('register/', views.register, name='register'),

]

ThugCodingBlog\blogproject\blog\views.py  
  
from django.shortcuts import render, get\_object\_or\_404, redirect

from django.core.paginator import Paginator

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

from django.contrib.auth import logout, login

from django.contrib.auth.forms import UserCreationForm

from .models import Post, UploadedFile, Comment

from .forms import PostForm, UploadFileForm, CommentForm

def home(request):

    post\_list = Post.objects.all().order\_by('-created\_at')

    paginator = Paginator(post\_list, 5)

    page\_number = request.GET.get('page')

    posts = paginator.get\_page(page\_number)

    files = UploadedFile.objects.all()

    return render(request, 'blog/home.html', {'posts': posts, 'files': files})

def post\_detail(request, pk):

    post = get\_object\_or\_404(Post, pk=pk)

    comments = post.comments.all().order\_by('-created\_at')

    if request.method == 'POST' and request.user.is\_authenticated:

        form = CommentForm(request.POST)

        if form.is\_valid():

            comment = form.save(commit=False)

            comment.author = request.user

            comment.post = post

            comment.save()

            return redirect('post\_detail', pk=post.pk)

    else:

        form = CommentForm()

    return render(request, 'blog/post\_detail.html', {

        'post': post,

        'comments': comments,

        'form': form

    })

@login\_required

def create\_post(request):

    if request.method == 'POST':

        form = PostForm(request.POST, request.FILES)

        if form.is\_valid():

            post = form.save(commit=False)

            post.author = request.user

            post.save()

            return redirect('post\_detail', pk=post.pk)

    else:

        form = PostForm()

    return render(request, 'blog/post\_form.html', {'form': form})

@login\_required

def edit\_post(request, pk):

    post = get\_object\_or\_404(Post, pk=pk)

    if request.method == 'POST':

        form = PostForm(request.POST, request.FILES, instance=post)

        if form.is\_valid():

            form.save()

            return redirect('post\_detail', pk=post.pk)

    else:

        form = PostForm(instance=post)

    return render(request, 'blog/post\_form.html', {'form': form})

def logout\_view(request):

    logout(request)

    return redirect('home')

@login\_required

def upload\_file(request):

    if request.method == 'POST':

        form = UploadFileForm(request.POST, request.FILES)

        if form.is\_valid():

            form.save()

            return redirect('file\_list')

    else:

        form = UploadFileForm()

    return render(request, 'blog/upload.html', {'form': form})

@login\_required

def file\_list(request):

    files = UploadedFile.objects.all()

    return render(request, 'blog/file\_list.html', {'files': files})

def register(request):

    if request.method == 'POST':

        form = UserCreationForm(request.POST)

        if form.is\_valid():

            user = form.save()

            login(request, user)

            return redirect('home')

    else:

        form = UserCreationForm()

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

ThugCodingBlog\blogproject\blogproject\settings.py

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-=)8-9\_ivj6=xc%vnno9kigarl27=ssu%kb6bh%t79\_isetcc+5'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS =  []

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'blog',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'blogproject.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'blogproject.wsgi.application'

# Database

# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC\_URL = '/static/'

# Default primary key field type

# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

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

# Media files

MEDIA\_URL = '/media/'

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

LOGIN\_REDIRECT\_URL = 'home'

LOGIN\_URL = 'login'

LOGOUT\_REDIRECT\_URL = 'home'

ThugCodingBlog\blogproject\blogproject\urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

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

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('blog.urls')),

    # ✅ Add these if missing

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

    path('logout/', auth\_views.LogoutView.as\_view(next\_page='/'), name='logout'),

]

if settings.DEBUG:

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