OSL Mini Project

*Instagram Look-a-Like Webapp*

Introduction:

We have created an Instagram Look-a-like web application. Users can do the following things:

1. Register to the website to get your credentials for the webapp.
2. Login to gain permissions to use the web app.
3. View posts from all users on the home-page.
4. View the post details and comments on the page dedicated to the post and also comment.
5. View a person’s profile along with all their posts on their profile page.
6. Upload your photo with a mind-blowing caption.
7. Like photos from the home-page, profile page and details page of the post.

Implementation:

* We have used Django (Python Web-Framework) for the backend(sever side) development.
* We have used SQLite3 database.
* The Frontend was developed from Scratch using Bootstrap.
* The webapp is responsive and can be run on various devices from mobile phones to computers.
* We have taken care of form validations for User Registrations and Logins which include ‘unique usernames and emails’, ‘non-blank first names’, etc.
* We have implemented additional security against password thefts by hashing the passwords.
* Cross-Site Request Forgery attacks are prevented by using CSRF Tokens provided by Django itself, wherever required.
* The posts are displayed in anti-chronological order while the comments on the posts are displayed in chronological order of being saved in the database.
* Models of the webapp include Post, Comment, User and Profile.
* The author of a Post and Comments are a Foreign Key to the Model User, Comments are Foreign Keys to Posts, Likes on Posts are linked using by using Many to Many Fields in Django which in the database, is saved using an intermediate table.
* Photos have been saved in the media directory further classified on the basis of username.

Source Code:

Models.py

from datetime import datetime

# Creating a custom path for storing the user photos

# Example : /MEDIA\_ROOT/photos/1234567890/abc.jpg

def path(instance, filename):

return 'photos/{0}/{1}'.format(instance.author.username, filename)

class Profile(models.Model):

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

bio = models.TextField(max\_length=500, blank=True, null=True)

display = models.FileField(blank=True, null=True)

class Post(models.Model):

author = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='post\_written')

caption = models.TextField(max\_length=500)

photo = models.FileField(upload\_to=path)

likes = models.ManyToManyField(User, related\_name='posts\_liked', blank=True)

time = models.DateTimeField(default=datetime.now)

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

return self.author.username + " - " + str(self.time)

class Comment(models.Model):

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

post = models.ForeignKey(Post, on\_delete=models.CASCADE)

text = models.CharField(max\_length=200)

time = models.DateTimeField(default=datetime.now)

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

return self.author.username + " - " + self.post.id + " - " + self.time

Views.py

from django.http import JsonResponse, HttpResponseRedirect

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

from django.views.decorators.csrf import ensure\_csrf\_cookie

from django.contrib.auth import authenticate, login, logout

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

from .models import User, Profile, Post, Comment

from .forms import UserLoginForm, UserRegisterForm, PostUploadForm

from django.conf import settings

# Create your views here.

@ensure\_csrf\_cookie

def loginView(request):

print(request.user)

if not request.user.id == None:

return redirect('project:homepage')

if request.method == 'POST':

form = UserLoginForm(request.POST)

username = request.POST.get('username')

password = request.POST.get('password')

user = authenticate(username=username, password=password)

if user is not None:

if user.is\_active:

login(request, user)

return redirect('project:homepage')

else:

error\_message = 'User is inactive.'

return render(request, 'project/login.html', {'error\_message': error\_message, 'form': form})

error\_message = 'Invalid Login'

return render(request, 'project/login.html', {'error\_message': error\_message, 'form': form})

if request.method == 'GET':

form = UserLoginForm(None)

return render(request, 'project/login.html', {'form': form})

def registerView(request):

if not request.user.id == None:

return redirect('project:homepage')

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

user = form.save(commit=False)

password = form.cleaned\_data['password']

user.set\_password(password)

user.save()

return redirect('project:login')

error\_message = 'Invalid Form'

return render(request, 'project/register.html', {'form': form, 'error\_message': error\_message})

if request.method == 'GET':

form = UserRegisterForm(None)

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

@login\_required

def homePage(request):

posts = Post.objects.exclude(author=request.user).order\_by('-time')

return render(request, 'project/homepage.html', {'posts': posts})

@login\_required

def postDetail(request, pk):

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

comments = Comment.objects.filter(post=post).all().order\_by('time')

return render(request, 'project/postDetail.html', {'post': post, 'comments': comments})

@login\_required

def like(request, pk):

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

user = request.user

try:

if user in post.likes.all():

post.likes.remove(user)

else:

post.likes.add(user)

except (KeyError, Post.DoesNotExist):

return JsonResponse({'success': False})

return JsonResponse({'success': True})

@login\_required

def uploadPost(request):

if request.method == 'POST':

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

if form.is\_valid():

post = form.save(commit=False)

post.author = request.user

post.save()

return redirect('project:detail', post.id)

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

else:

form = PostUploadForm(None)

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

@login\_required

def uploadComment(request, pk):

if request.method == 'POST':

user = get\_object\_or\_404(User, id=request.user.id)

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

content = request.POST.get('comment')

comment = Comment(author=user, text=content, post=post)

comment.save()

return redirect('project:detail', pk=pk)

@login\_required

def profile(request, username):

user = get\_object\_or\_404(User, username=username)

posts = Post.objects.filter(author=user).all().order\_by('-time')

return render(request, 'project/profile.html', {'user': user, 'posts': posts})

@login\_required

def logout\_view(request):

logout(request)

return redirect('project:homepage')

Urls.py (from the app):

from django.urls import path

from . import views

app\_name = 'project'

urlpatterns = [

path("login/", views.loginView, name="login"),

path("register/", views.registerView, name="register"),

path("home/", views.homePage, name="homepage"),

path("post/<int:pk>/", views.postDetail, name="detail"),

path("profile/<str:username>/", views.profile, name="profile"),

path("post/<int:pk>/like/", views.like, name="like"),

path("post/<int:pk>/comment/", views.uploadComment, name="comment"),

path("post/upload/", views.uploadPost, name="uploadPost"),

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

]

Urls.py (from the project):

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

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

path('app/', include('project.urls')),

]

if settings.DEBUG:

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

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

Forms.py:

from django.contrib.auth.models import User

from django import forms

from .models import Post

class UserLoginForm(forms.ModelForm):

password = forms.CharField(widget=forms.PasswordInput)

class Meta:

model = User

fields = ['username', 'password']

class UserRegisterForm(forms.ModelForm):

password = forms.CharField(widget=forms.PasswordInput)

email = forms.CharField(widget=forms.EmailInput)

class Meta:

model = User

fields = ['first\_name', 'last\_name', 'username', 'email', 'password']

class PostUploadForm(forms.ModelForm):

photo = forms.FileField()

caption = forms.CharField()

class Meta:

model = Post

fields = ['photo', 'caption']

Snippet of a Post in Frontend:

<div class="card-deck">

<div class="card">

<a href="{% url 'project:detail' post.id %}">

<div class="card-img-top">

<img class="postImage" src="{{ post.photo.url }}" alt="Card image cap">

</div></a>

<div class="card-body">

<img class="like {% if request.user in post.likes.all %}liked{% endif %}" id="{{ post.id }}" src="../../static/project/img/{% if not request.user in post.likes.all %}InstagramLike.png{% else %}Liked.png{% endif %}" height="30px" width="30px" style="float: right">

<p class="text-muted likecount" style="float: right; margin-top: 2px;">{{ post.likes.count }}</p>

<a href="{% url 'project:profile' post.author.username %}"><p class="card-text" style="font-weight: bold">{{ post.author.username }}</p></a>

<span class="caption text-muted">{{ post.caption }}</span>

<p class="card-text"><small class="text-muted">{{ post.time|date:"M d, Y" }}</small></p>

</div>

</div>

Screenshots:





