**ARCHIVE OF DOCUMENTARY PROJECTS**

Account views

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login

from django.contrib.auth.models import User

from .forms import UserForm

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

@login\_required

def profile(request):

    user = request.user

    context = {

        'user': user

    }

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

def user\_login(request):

    if request.method == 'POST':

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

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

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

        if user is not None:

            login(request, user)

            return redirect('documentary\_list')

        else:

            return render(request, 'account/login.html', {'error': 'Invalid username or password.'})

    return render(request, 'account/login.html')

def user\_signup(request):

    if request.method == 'POST':

        form = UserForm(request.POST)

        if form.is\_valid():

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

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

            User.objects.create\_user(username=username, password=password)

            return redirect('login')

    else:

        form = UserForm()

    return render(request, 'account/signup.html', {'form': form})

main system view

import json

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

from django.http import JsonResponse

from .models import Documentary, Comment, Reply

from .forms import CommentForm, ReplyForm, DocumentaryForm

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

from django.db import models

from django.db.models import Q

def home(request):

    return render(request, 'home.html')

@login\_required

def documentary\_list(request):

    query = request.GET.get('q')

    if query:

        # Perform a search across multiple fields using Q objects

        documentaries = Documentary.objects.filter(

            Q(title\_\_icontains=query) |

            Q(description\_\_icontains=query) |

            Q(author\_\_icontains=query) |

            Q(year\_\_icontains=query)  # Assuming year is an integer field

        )

    else:

        documentaries = Documentary.objects.all()

    context = {'documentaries': documentaries, 'query': query}

    if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

        return render(request, 'documentary/documentary\_list\_ajax.html', context)

    return render(request, 'documentary/documentary\_list.html', context)

@login\_required

def documentary\_details(request, documentary\_id):

    documentary = get\_object\_or\_404(Documentary, pk=documentary\_id)

    comments = documentary.comments.all()

    if request.method == 'POST':

        if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

            comment\_form = CommentForm(request.POST)

            if comment\_form.is\_valid():

                # Save the comment

                comment = comment\_form.save(commit=False)

                comment.documentary = documentary

                comment.user = request.user

                comment.save()

                return JsonResponse({'success': True})

            else:

                return JsonResponse({'success': False, 'errors': comment\_form.errors})

        else:

            comment\_form = CommentForm(request.POST)

            reply\_form = ReplyForm()

            if comment\_form.is\_valid():

                # Save the comment

                comment = comment\_form.save(commit=False)

                comment.documentary = documentary

                comment.user = request.user

                comment.save()

                return redirect('documentary\_details', pk=pk)

    else:

        comment\_form = CommentForm()

        reply\_form = ReplyForm()

    context = {

        'documentary': documentary,

        'comments': comments,

        'comment\_form': comment\_form,

        'reply\_form': reply\_form,

    }

    return render(request, 'documentary/documentary\_details.html', context)

@login\_required

def register\_documentary(request):

    if request.method == 'POST':

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

        if form.is\_valid():

            form.save()

            return JsonResponse({'success': True, 'message': 'Documentary added successfully'})

        else:

            return JsonResponse({'success': False, 'message': 'Failed to add documentary'})

    else:

        form = DocumentaryForm()

    return render(request, 'documentary/register\_documentary.html', {'form': form})

@login\_required

def submit\_comment(request, documentary\_id):

    documentary = get\_object\_or\_404(Documentary, id=documentary\_id)

    if request.method == 'POST':

        form = CommentForm(request.POST)

        if form.is\_valid():

            comment = form.save(commit=False)

            comment.documentary = documentary

            comment.user = request.user

            comment.save()

            return JsonResponse({'status': 'success', 'comment\_id': comment.id})

    return JsonResponse({'status': 'error'})

@login\_required

def documentary\_registered(request):

    if request.method == 'POST':

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

        if form.is\_valid():

            form.save()

            return JsonResponse({'success': True, 'message': 'Documentary added successfully'})

        else:

            return JsonResponse({'success': False, 'message': 'Failed to add documentary'})

    else:

        form = DocumentaryForm()

    return render(request, 'documentary/register\_documentary.html', {'form': form})

@login\_required

def submit\_reply(request, documentary\_id):

    comment\_id = request.POST.get('comment\_id')

    comment = get\_object\_or\_404(Comment, id=comment\_id)

    if request.method == 'POST':

        form = ReplyForm(request.POST)

        if form.is\_valid():

            reply = form.save(commit=False)

            reply.comment = comment

            reply.user = request.user

            reply.save()

            return JsonResponse({'status': 'success'})

    return JsonResponse({'status': 'error'})

models

from django.db import models

from django.contrib.auth.models import User

class Documentary(models.Model):

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

    description = models.TextField()

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

    year = models.IntegerField()

    pdf = models.FileField(upload\_to='documents/')

    class Meta:

        ordering = ['-id']

    def \_\_str\_\_(self):

        return self.title

class Comment(models.Model):

    documentary = models.ForeignKey(Documentary, on\_delete=models.CASCADE, related\_name='comments')

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

    content = models.TextField()

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

    def \_\_str\_\_(self):

        return f'Comment by {self.user.username} on {self.documentary.title}'

class Reply(models.Model):

    comment = models.ForeignKey(Comment, on\_delete=models.CASCADE, related\_name='replies')

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

    content = models.TextField()

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

    def \_\_str\_\_(self):

        return f'Reply by {self.user.username} to {self.comment}'

forms

from django import forms

from .models import Documentary, Comment, Reply

class DocumentaryForm(forms.ModelForm):

    class Meta:

        model = Documentary

        fields = ['title', 'description', 'year', 'author', 'pdf']

        widgets = {

            'title': forms.TextInput(attrs={'class': 'form-control'}),

            'description': forms.Textarea(attrs={'class': 'form-control'}),

            'year': forms.NumberInput(attrs={'class': 'form-control'}),

            'author': forms.TextInput(attrs={'class': 'form-control'}),

            'pdf': forms.FileInput(attrs={'class': 'form-control-file'}),

        }

class CommentForm(forms.ModelForm):

    class Meta:

        model = Comment

        fields = ['content']

        widgets = {

            'content': forms.Textarea(attrs={'class': 'form-control'}),

        }

class ReplyForm(forms.ModelForm):

    class Meta:

        model = Reply

        fields = ['content']

        widgets = {

            'content': forms.Textarea(attrs={'class': 'form-control'}),

        }