**SAMPLE CODE**

**Admin Views:**

from django.shortcuts import render

from django.contrib import messages

from users.models import UserRegistrationModel, UserActionsModel

# Create your views here.

def AdminLoginCheck(request):

if request.method == 'POST':

usrid = request.POST.get('loginid')

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

print("User ID is = ", usrid)

if usrid == 'admin' and pswd == 'admin':

return render(request, 'admins/AdminHome.html')

else:

messages.success(request, 'Please Check Your Login Details')

return render(request, 'AdminLogin.html', {})

def AdminHome(request):

return render(request, 'admins/AdminHome.html')

def RegisterUsersView(request):

data = UserRegistrationModel.objects.all()

return render(request,'admins/viewregisterusers.html',{'data':data})

def ActivaUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).update(status=status)

data = UserRegistrationModel.objects.all()

return render(request,'admins/viewregisterusers.html',{'data':data})

def BlockUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'waiting'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).update(status=status)

data = UserRegistrationModel.objects.all()

return render(request, 'admins/viewregisterusers.html', {'data': data})

def UserActionsViewbyAdmin(request):

data = UserActionsModel.objects.all()

return render(request, 'admins/allactionviews.html', {'data': data})

**Base.html:**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

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

<meta name="description" content="">

<meta name="author" content="">

<link href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500,700" rel="stylesheet">

<title>Human Action</title>

<!-- Additional CSS Files -->

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

<link rel="stylesheet" type="text/css" href="{% static 'css/font-awesome.css' %}">

<link rel="stylesheet" href="{% static 'css/templatemo-softy-pinko.css' %}">

<!-- Custom Styles -->

<style>

body {

font-family: 'Montserrat', sans-serif;

color: #2d2d2d;

background-color: #db2534;

margin: 0;

padding: 0;

}

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

color: #343a40;

}

.header-area {

background-color: #17a2b8;

padding: 15px 0;

position: sticky;

top: 0;

width: 100%;

z-index: 1000;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.main-nav {

display: flex;

justify-content: space-between;

align-items: center;

}

.main-nav .nav {

list-style: none;

display: flex;

margin: 0;

padding: 0;

}

.main-nav .nav li {

margin: 0 15px;

}

.main-nav .nav li a {

color: #b11717;

font-weight: 400;

text-decoration: none;

padding: 10px 20px;

transition: color 0.3s;

}

.main-nav .nav li a:hover {

color: #ffc107;

}

.welcome-area:before {

content: "";

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

}

.header-text {

position: relative;

z-index: 2;

}

.header-text h1 {

font-size: 48px;

font-weight: 700;

margin-bottom: 20px;

}

.header-text p {

font-size: 18px;

font-weight: 300;

color: #1c8f2b;

max-width: 800px;

margin: 0 auto;

}

#preloader {

background-color: #3bbb56;

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

z-index: 9999;

display: flex;

align-items: center;

justify-content: center;

}

.jumper > div {

width: 15px;

height: 15px;

background-color: #17a2b8;

margin: 3px;

border-radius: 100%;

animation: jump 0.6s infinite alternate;

}

@keyframes jump {

to {

transform: translateY(-20px);

}

}

.jumper {

display: flex;

}

</style>

</head>

<body>

<!-- Preloader Start -->

<div id="preloader">

<div class="jumper"><div></div><div></div><div></div></div>

</div>

<!-- Preloader End -->

<!-- Header Area Start -->

<header class="header-area">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="main-nav">

<center>

<a href="#" class="logo">

<h5 style="color: #ca1c1c;">Human Action Recognition</h5>

</a>

<ul class="nav">

<li><a href="{% url 'index' %}" style="color: rgb(78, 197, 78);">Home</a></li>

<li><a href="{% url 'UserLogin' %}" style="color: red;">User</a></li>

<li><a href="{% url 'AdminLogin' %}" style="color: red;">Admin</a></li>

<li><a href="{% url 'UserRegister' %}" style="color: red;">Registration</a></li>

</ul>

</center>

<a class='menu-trigger'><span>Menu</span></a>

</nav>

</div>

</div>

</div>

</header>

<!-- Header Area End -->

<!-- Welcome Area Start -->

<div class="welcome-area" id="welcome">

<div class="header-text">

<div class="container">

<div class="row">

<div class="offset-xl-3 col-xl-6 offset-lg-2 col-lg-8 col-md-12 col-sm-12">

<h1><strong>Light-Weight Deep Learning Model for Human Action Recognition in Videos</strong></h1>

<p>The aim of this project is to develop a model for human actions such as running, jogging, walking, clapping, handwaving, and boxing. A series of videos is given for the layout, where an individual executes an event in each video. The action performed in that particular video will be the label of the video. This relationship must be learned by the model, and the label of an input (video) which it has never seen can then be predicted. Technically, despite descriptions of these acts, the model would need to learn to distinguish between various human behaviors. There may be many content identification programs that can work on tasks like active object tracking for identifying an item such as a vehicle or a human from a CCTV picture and learning the patterns in the movement of humans when we can create a pattern that will guide us (humans) to perform a variety of activities.</p>

</div>

</div>

</div>

</div>

</div>

<!-- Welcome Area End -->

{% block contents %}

{% endblock %}

<!-- jQuery -->

<script src="{% static 'js/jquery-2.1.0.min.js' %}"></script>

<!-- Bootstrap -->

<script src="{% static 'js/popper.js' %}"></script>

<script src="{% static 'js/bootstrap.min.js' %}"></script>

<!-- Plugins -->

<script src="{% static 'js/scrollreveal.min.js' %}"></script>

<script src="{% static 'js/waypoints.min.js' %}"></script>

<script src="{% static 'js/jquery.counterup.min.js' %}"></script>

<script src="{% static 'js/imgfix.min.js' %}"></script>

<!-- Global Init -->

<script src="{% static 'js/custom.js' %}"></script>

</body>

</html>

**User Views:**

from django.shortcuts import render

# Create your views here.

from django.shortcuts import render, HttpResponse

from django.contrib import messages

from .forms import UserRegistrationForm

from .models import UserRegistrationModel, UserActionsModel

from django.core.files.storage import FileSystemStorage

from datetime import datetime

# Create your views here.

def UserRegisterActions(request):

if request.method == 'POST':

form = UserRegistrationForm(request.POST)

if form.is\_valid():

print('Data is Valid')

form.save()

messages.success(request, 'You have been successfully registered')

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

else:

messages.success(request, 'Email or Mobile Already Existed')

print("Invalid form")

else:

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

if request.method == "POST":

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

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

print("Login ID = ", loginid, ' Password = ', pswd)

try:

check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggeduser'] = check.name

request.session['loginid'] = loginid

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'users/UserHomePage.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'UserLogin.html')

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Login id and password')

return render(request, 'UserLogin.html', {})

def UserHome(request):

return render(request, 'users/UserHomePage.html', {})

def UserUploadPic(request):

return render(request, "users/UploadPicForm.html", {})

def UploadImageAction(request):

image\_file = request.FILES['file']

# let's check if it is a csv file

if not image\_file.name.endswith('.png'):

messages.error(request, 'THIS IS NOT A PNG FILE')

fs = FileSystemStorage(location="media/actions/")

filename = fs.save(image\_file.name, image\_file)

# detect\_filename = fs.save(image\_file.name, image\_file)

uploaded\_file\_url = "/media/actions/" + filename # fs.url(filename)

print("Image path ", uploaded\_file\_url)

from .utility.ClassifyIMages import ClassifyUserPics

obj = ClassifyUserPics()

result = obj.startProcess(filename)

print("Result=", result)

return render(request, "users/UploadPicForm.html", {'result': result})

def UserStreamAction(request):

from .utility.UserLiveStream import LiveStream

obj = LiveStream()

actions = obj.startProcess()

print('Result is:', actions)

name = request.session['loggeduser']

login\_user = request.session['loginid']

email = request.session['email']

c\_date = datetime.now()

UserActionsModel.objects.create(name=name, login\_user=login\_user, email=email, actions=str(actions), c\_date=c\_date)

return render(request, 'users/UserActions.html', {'actions': actions})

def UserActionsViews(request):

login\_user = request.session['loginid']

data = UserActionsModel.objects.filter(login\_user=login\_user)

return render(request, 'users/UserActionViews.html', {'data': data})

def UserVideoProcess(request):

if request.method == 'POST':

image\_file = request.FILES['file']

fs = FileSystemStorage(location="media/videos/")

filename = fs.save(image\_file.name, image\_file)

# detect\_filename = fs.save(image\_file.name, image\_file)

uploaded\_file\_url = "media/videos/" + filename # fs.url(filename)

print("Image path ", uploaded\_file\_url)

import subprocess

import os

path = os.path.join(uploaded\_file\_url)

harActivity = "human\_activity\_reco.py --model resnet-34\_kinetics.onnx --classes action\_recognition\_kinetics.txt --input "+path

print(harActivity)

subprocess.call("python " + harActivity)

return render(request, 'users/UserVideoProcess.html', {})

else:

return render(request, 'users/UserVideoProcess.html', {})