**VIEWS.PY**

from django.shortcuts import render

from django.template import RequestContext

import pymysql

from django.http import HttpResponse

from django.conf import settings

from django.core.files.storage import FileSystemStorage

import datetime

import cv2

from keras.models import load\_model

from keras.preprocessing.image import img\_to\_array

import numpy as np

def Index(request):

if request.method == 'GET':

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

def User(request):

if request.method == 'GET':

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

def Admin(request):

if request.method == 'GET':

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

def AdminLogin(request):

if request.method == 'POST':

username = request.POST.get('t1', False)

password = request.POST.get('t2', False)

if username == 'admin' and password == 'admin':

context= {'data':'welcome '+username}

return render(request, 'AdminScreen.html', context)

else:

context= {'data':'login failed'}

return render(request, 'Admin.html', context)

def ViewRating(request):

if request.method == 'GET':

strdata = '<table border=1 align=center width=100%><tr><th>Customer Name</th><th>Rating</th><th>Facial Expression</th><th>Photo</th> <th>Date & Time</th></tr><tr>'

con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root', database = 'facial',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select \* FROM rating")

rows = cur.fetchall()

for row in rows:

strdata+='<td>'+row[0]+'</td><td>'+str(row[1])+'</td><td>'+row[2]+'</td><td><img src=/static/photo/'+row[0]+'.png width=200 height=200></img></td><td>'+str(row[4])+'</td></tr>'

context= {'data':strdata}

return render(request, 'ViewRatings.html', context)

def Rating(request):

if request.method == 'POST' and request.FILES['t3']:

output = ''

myfile = request.FILES['t3']

name = request.POST.get('t1', False)

rating = request.POST.get('t2', False)

fs = FileSystemStorage()

filename = fs.save('C:/Python/Facial/Facial/FacialApp/static/photo/'+name+'.png', myfile)

now = datetime.datetime.now()

current\_time = now.strftime("%Y-%m-%d %H:%M:%S")

detection\_model\_path = 'C:/Python/Facial/Facial/FacialApp/haarcascade\_frontalface\_default.xml'

emotion\_model\_path = 'C:/Python/Facial/Facial/FacialApp/\_mini\_XCEPTION.106-0.65.hdf5'

face\_detection = cv2.CascadeClassifier(detection\_model\_path)

emotion\_classifier = load\_model(emotion\_model\_path, compile=False)

EMOTIONS = ["angry","disgust","scared", "happy", "sad", "surprised","neutral"]

orig\_frame = cv2.imread('C:/Python/Facial/Facial/FacialApp/static/photo/'+name+'.png')

orig\_frame = cv2.resize(orig\_frame, (48, 48))

frame = cv2.imread(filename,0)

faces = face\_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flags=cv2.CASCADE\_SCALE\_IMAGE)

print("==================="+str(len(faces)))

print(emotion\_classifier)

if len(faces) > 0:

faces = sorted(faces, reverse=True,key=lambda x: (x[2] - x[0]) \* (x[3] - x[1]))[0]

(fX, fY, fW, fH) = faces

roi = frame[fY:fY + fH, fX:fX + fW]

roi = cv2.resize(roi, (48, 48))

roi = roi.astype("float") / 255.0

roi = img\_to\_array(roi)

roi = np.expand\_dims(roi, axis=0)

preds = emotion\_classifier.predict(roi)[0]

emotion\_probability = np.max(preds)

label = EMOTIONS[preds.argmax()]

if label == 'happy':

output = 'Satisfied'

if label == 'neutral':

output = 'Neutral'

if label == 'angry' or label == 'sad' or label == 'disgust' or label == 'scared' or label == 'surprised':

output = 'Disappointed'

print("==================="+output)

db\_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root', database = 'facial',charset='utf8')

db\_cursor = db\_connection.cursor()

query = "INSERT INTO rating(customer\_name,rating,facial\_expression,photo\_path,rating\_date) VALUES('"+name+"','"+rating+"','"+output+"','"+name+'.png'+"','"+current\_time+"')"

db\_cursor.execute(query)

db\_connection.commit()

print(db\_cursor.rowcount, "Record Inserted")

if db\_cursor.rowcount == 1:

context= {'data':'Your Rating is : '+rating+' and Facial Expression : '+output}

return render(request, 'User.html', context)

else:

context= {'data':'Error in request process'}

return render(request, 'User.html', context)