Attendance System Prgms

1] Runfile.py

import tkinter as tk  
from tkinter import Message **,**Text  
import cv2**,**os  
import shutil  
import csv  
import numpy as np  
from PIL import Image**,** ImageTk  
import pandas as pd  
#import datetime  
import time  
from datetime import datetime  
  
window = tk.Tk()  
#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')  
window.title("Face\_Recogniser")  
  
dialog\_title = 'QUIT'  
dialog\_text = 'Are you sure?'  
#answer = messagebox.askquestion(dialog\_title, dialog\_text)  
   
#window.geometry('1280x720')  
window.configure(background='blue')  
  
#window.attributes('-fullscreen', True)  
  
window.grid\_rowconfigure(**0,** weight=**1**)  
window.grid\_columnconfigure(**0,** weight=**1**)  
  
message = tk.Label(window**,** text="Face-Recognition-Based-Attendance-Management-System" **,**bg="Green" **,**fg="white" **,**width=**50 ,**height=**3,**font=('times'**, 30,** 'italic bold underline'))   
  
message.place(x=**200,** y=**20**)  
  
lbl = tk.Label(window**,** text="Enter ID"**,**width=**20 ,**height=**2 ,**fg="red" **,**bg="yellow" **,**font=('times'**, 15,** ' bold ') )   
lbl.place(x=**400,** y=**200**)  
  
txt = tk.Entry(window**,**width=**20 ,**bg="yellow" **,**fg="red"**,**font=('times'**, 15,** ' bold '))  
txt.place(x=**700,** y=**215**)  
  
lbl2 = tk.Label(window**,** text="Enter Name"**,**width=**20 ,**fg="red" **,**bg="yellow" **,**height=**2 ,**font=('times'**, 15,** ' bold '))   
lbl2.place(x=**400,** y=**300**)  
  
txt2 = tk.Entry(window**,**width=**20 ,**bg="yellow" **,**fg="red"**,**font=('times'**, 15,** ' bold ') )  
txt2.place(x=**700,** y=**315**)  
  
lbl3 = tk.Label(window**,** text="Notification : "**,**width=**20 ,**fg="red" **,**bg="yellow" **,**height=**2 ,**font=('times'**, 15,** ' bold underline '))   
lbl3.place(x=**400,** y=**400**)  
  
message = tk.Label(window**,** text="" **,**bg="yellow" **,**fg="red" **,**width=**30 ,**height=**2,** activebackground = "yellow" **,**font=('times'**, 15,** ' bold '))   
message.place(x=**700,** y=**400**)  
  
lbl3 = tk.Label(window**,** text="Attendance : "**,**width=**20 ,**fg="red" **,**bg="yellow" **,**height=**2 ,**font=('times'**, 15,** ' bold underline'))   
lbl3.place(x=**400,** y=**650**)  
  
  
message2 = tk.Label(window**,** text="" **,**fg="red" **,**bg="yellow"**,**activeforeground = "green"**,**width=**30 ,**height=**2 ,**font=('times'**, 15,** ' bold '))   
message2.place(x=**700,** y=**650**)  
   
def clear():  
 txt.delete(**0,** 'end')   
 res = ""  
 message.configure(text= res)  
  
def clear2():  
 txt2.delete(**0,** 'end')   
 res = ""  
 message.configure(text= res)   
   
def is\_number(s):  
 try:  
 float(s)  
 return True  
 except ValueError:  
 pass  
   
 try:  
 import unicodedata  
 unicodedata.numeric(s)  
 return True  
 except (TypeError**,** ValueError):  
 pass  
   
 return False  
   
def TakeImages():  
 co=['Id']  
 df=pd.read\_csv("StudentDetails\StudentDetails.csv"**,**names=co)  
   
 namess = df['Id']  
 ides=[]  
  
   
 Id=(txt.get())  
   
 ides=Id  
   
 name=(txt2.get())  
 estest=**0** if ides in namess:  
 estest=**1** else:  
 estest=**0** #print estest  
 if (estest==**0**):  
 if(is\_number(Id) and name.isalpha()):  
 cam = cv2.VideoCapture(**0**)  
 harcascadePath = "haarcascade\_frontalface\_default.xml"  
 detector=cv2.CascadeClassifier(harcascadePath)  
 sampleNum=**0** while(True):  
 ret**,** img = cam.read()  
 gray = cv2.cvtColor(img**,** cv2.COLOR\_BGR2GRAY)  
 faces = detector.detectMultiScale(gray**, 1.3, 5**)  
 for (x**,**y**,**w**,**h) in faces:  
 cv2.rectangle(img**,**(x**,**y)**,**(x+w**,**y+h)**,**(**255,0,0**)**,2**)   
 #incrementing sample number   
 sampleNum=sampleNum+**1** #saving the captured face in the dataset folder TrainingImage  
 cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg"**,** gray[y:y+h**,**x:x+w])  
 #display the frame  
 cv2.imshow('frame'**,**img)  
 #wait for 100 miliseconds   
 if cv2.waitKey(**100**) & **0xFF** == ord('q'):  
 break  
 # break if the sample number is morethan 100  
 elif sampleNum>**200**:  
 break  
 cam.release()  
 cv2.destroyAllWindows()   
 res = "Images Saved for ID : " + Id +" Name : "+ name  
 row = [Id **,** name]  
 with open('StudentDetails\StudentDetails.csv'**,**'a+') as csvFile:  
 writer = csv.writer(csvFile)  
 writer.writerow(row)  
 csvFile.close()  
 message.configure(text= res)  
 else:  
 if(is\_number(Id)):  
 res = "Enter Alphabetical Name"  
 message.configure(text= res)  
 if(name.isalpha()):  
 res = "Enter Numeric Id"  
 message.configure(text= res)  
   
 else:  
 res = "Already Id Exist"  
 message.configure(text= res)  
  
def TakeImages1():  
 co=['Id']  
 df=pd.read\_csv("StudentDetails\StudentDetails.csv"**,**names=co)  
   
 namess = df['Id']  
 ides=[]  
  
   
 Id=(txt.get())  
   
 ides=Id  
 name=(txt2.get())  
 estest=**0** if ides in namess:  
 estest=**1** else:  
 estest=**0** #print estest  
 if (estest==**1**):  
 if(is\_number(Id) and name.isalpha()):  
 cam = cv2.VideoCapture(**0**)  
 harcascadePath = "haarcascade\_frontalface\_default.xml"  
 detector=cv2.CascadeClassifier(harcascadePath)  
 sampleNum=**0** while(True):  
 ret**,** img = cam.read()  
 gray = cv2.cvtColor(img**,** cv2.COLOR\_BGR2GRAY)  
 faces = detector.detectMultiScale(gray**, 1.3, 5**)  
 for (x**,**y**,**w**,**h) in faces:  
 cv2.rectangle(img**,**(x**,**y)**,**(x+w**,**y+h)**,**(**255,0,0**)**,2**)   
 #incrementing sample number   
 sampleNum=sampleNum+**1** #saving the captured face in the dataset folder TrainingImage  
 cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg"**,** gray[y:y+h**,**x:x+w])  
 #display the frame  
 cv2.imshow('frame'**,**img)  
 #wait for 100 miliseconds   
 if cv2.waitKey(**100**) & **0xFF** == ord('q'):  
 break  
 # break if the sample number is morethan 100  
 elif sampleNum>**200**:  
 break  
 cam.release()  
 cv2.destroyAllWindows()   
 res = "Images Saved for ID : " + Id +" Name : "+ name  
 message.configure(text= res)  
 else:  
 if(is\_number(Id)):  
 res = "Enter Alphabetical Name"  
 message.configure(text= res)  
 if(name.isalpha()):  
 res = "Enter Numeric Id"  
 message.configure(text= res)  
   
 else:  
 res = "Id is Not Trained Before"  
 message.configure(text= res)  
   
   
def TrainImages():  
 recognizer = cv2.face\_LBPHFaceRecognizer.create()

#recognizer = cv2.face.LBPHFaceRecognizer\_create()#$cv2.createLBPHFaceRecognizer()  
 harcascadePath = "haarcascade\_frontalface\_default.xml"  
 detector =cv2.CascadeClassifier(harcascadePath)  
 faces**,**Id = getImagesAndLabels("TrainingImage")  
 recognizer.train(faces**,** np.array(Id))  
 recognizer.save("TrainingImageLabel\Trainner.yml")  
 res = "Image Trained"#+",".join(str(f) for f in Id)  
 message.configure(text= res)  
  
def getImagesAndLabels(path):  
 #get the path of all the files in the folder  
 imagePaths=[os.path.join(path**,**f) for f in os.listdir(path)]   
 #print(imagePaths)  
   
 #create empth face list  
 faces=[]  
 #create empty ID list  
 Ids=[]  
 #now looping through all the image paths and loading the Ids and the images  
 for imagePath in imagePaths:  
 #loading the image and converting it to gray scale  
 pilImage=Image.open(imagePath).convert('L')  
 #Now we are converting the PIL image into numpy array  
 imageNp=np.array(pilImage**,**'uint8')  
 #getting the Id from the image  
 Id=int(os.path.split(imagePath)[-**1**].split(".")[**1**])  
 # extract the face from the training image sample  
 faces.append(imageNp)  
 Ids.append(Id)   
 return faces**,**Ids  
  
def calculate():  
 import pandas as pd  
 import glob  
 dfstud = pd.read\_csv('StudentDetails\StudentDetails.csv'**,** index\_col=None**,** header=**0**)  
   
 studname=dfstud['Name'].values  
   
 path = r'Attendance' # use your path  
 all\_files = glob.glob(path + "/\*.csv")  
 ts = time.time()   
 date = datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
  
 li = []  
  
 for filename in all\_files:  
 if date in filename:  
 df = pd.read\_csv(filename**,** index\_col=None**,** header=**0**)  
 li.append(df)  
 else:  
 print('inside else')  
  
 frame = pd.concat(li**,** axis=**0,** ignore\_index=True)  
   
 esname=frame['Name'].values  
 # print 'Esname:'  
 present=[]  
 coues=[]  
 espresent={}  
 #print esname  
 for st in studname:  
 cou=**0**;  
 name=st  
 present.append(name)  
 for es in esname:  
 siva=es  
 # print'List ==2 Name'  
 siva=siva.replace("'"**,**"")  
 siva2=siva.replace("["**,**"")  
 siva3=siva2.replace("]"**,**"")  
 # print siva3  
 # print name  
 if name==siva3:  
 cou=cou+**1** coues.append(cou)  
   
 #print'present list and count list=='  
 #print present  
 #print coues  
 lenList = len(present)  
 for elements in range(**0,**lenList) :  
 key = present[elements]  
 value = coues[elements]  
 espresent[key] = value  
 #print'Espresent==='  
 #print espresent.items()  
 ts = time.time()   
 date = datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
 timeStamp = datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 Hour**,**Minute**,**Second=timeStamp.split(":")  
 prelist=[]  
 abslist=[]  
 for st in studname:  
 stname=st  
 precou=espresent.get(stname**, 0**)  
 # print'precount & Student name'  
 # print precou  
 # print stname  
 if precou>**3**:  
 prelist.append(stname)  
 fileName1="Attendance\Final\_Present\_"+date+"\_"+Hour+".txt"  
 with open(fileName1**,** 'w') as f:  
 for item in prelist:  
 f.write("%s\n" % item)  
   
 else:  
 abslist.append(stname)  
 fileName1="Attendance\Final\_Absent\_"+date+"\_"+Hour+".txt"  
 with open(fileName1**,** 'w') as f:  
 for item in abslist:  
 f.write("%s\n" % item)  
   
   
   
   
   
   
   
def TrackImages():  
 recognizer = cv2.face.LBPHFaceRecognizer\_create()#cv2.createLBPHFaceRecognizer()  
 recognizer.read("TrainingImageLabel\Trainner.yml")  
 harcascadePath = "haarcascade\_frontalface\_default.xml"  
 faceCascade = cv2.CascadeClassifier(harcascadePath);   
 df=pd.read\_csv("StudentDetails\StudentDetails.csv")  
 cam = cv2.VideoCapture(**0**)  
 font = cv2.FONT\_HERSHEY\_SIMPLEX   
 col\_names = ['Id'**,**'Name'**,**'Date'**,**'Time']  
 co=['name']  
 attendance = pd.DataFrame(columns = col\_names)  
 namess=""  
  
 for index**,** row in df.iterrows():  
 namess+= row['Name']+" "  
 aa=""  
  
  
   
 attendance1 = pd.DataFrame(columns = co)  
 while True:  
 ret**,** im =cam.read()  
 gray=cv2.cvtColor(im**,**cv2.COLOR\_BGR2GRAY)  
 faces=faceCascade.detectMultiScale(gray**, 1.2,5**)   
 for(x**,**y**,**w**,**h) in faces:  
 cv2.rectangle(im**,**(x**,**y)**,**(x+w**,**y+h)**,**(**225,0,0**)**,2**)  
 Id**,** conf = recognizer.predict(gray[y:y+h**,**x:x+w])   
 if(conf < **50**):  
 ts = time.time()   
 date = datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
 timeStamp = datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 aa=df.loc[df['Id'] == Id]['Name'].values  
 #print str(aa)  
 aaa=''.join(e for e in aa if e.isalnum())  
 #print aaa  
 tt=str(Id)+"-"+aa  
 attendance.loc[len(attendance)] = [Id**,**aa**,**date**,**timeStamp]  
 attendance1.loc[len(attendance)] = [aa]  
   
 namess=namess.replace(aaa**,** " ")  
   
   
   
   
 else:  
 Id='Unknown'   
 tt=str(Id)   
 if(conf > **75**):  
 import os  
 noOfFile=len(os.listdir("ImagesUnknown"))+**1** #cv2.imwrite("ImagesUnknown\Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])   
 cv2.putText(im**,**str(tt)**,**(x**,**y+h)**,** font**, 1,**(**255,255,255**)**,2**)   
 attendance=attendance.drop\_duplicates(subset=['Id']**,**keep='first')   
 cv2.imshow('im'**,**im)  
 #from datetime import datetime  
 local = datetime.now()  
 aa= local.strftime("%M")  
 #print aa  
 status=**0** if int(aa)%**2**==**0**:  
 status=**1** if status==**1**:  
 ts = time.time()   
 date = datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
 timeStamp = datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 Hour**,**Minute**,**Second=timeStamp.split(":")  
 fileName="Attendance\Attendance\_"+date+"\_"+Hour+"-"+Minute+".csv"  
 fileName1="Attendance\Absent\_"+date+"\_"+Hour+"-"+Minute+".txt"  
 import os  
 exists = os.path.isfile('/path/to/file')  
 if exists:  
 print ("")  
 # Store configuration file values  
 else:  
 attendance.to\_csv(fileName**,**index=False)  
 #gradeBool = (df != attendance1).stack() # Create Frame of comparison booleans  
 #gradediff = pd.concat([df['Name'].stack()[gradeBool],attendance1.stack()[gradeBool]], axis=1)  
  
 #df\_1notin2 = df[~(df['Name'].isin(attendance1['Name']) )].reset\_index(drop=True)  
   
 file = open(fileName1**,**"w")   
 file.write(namess)  
 file.close()   
 #namess.to\_csv(fileName,index=False)  
 # print 'Absent'  
 #print aa  
 #print namess  
  
   
## cam.release()  
## cv2.destroyAllWindows()  
 #print(attendance)  
 res=attendance  
 message2.configure(text= res)  
   
 #break  
 if (cv2.waitKey(**1**)==ord('q')):  
 cam.release()  
 cv2.destroyAllWindows()  
  
 break  
## ts = time.time()   
## date = datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
## timeStamp = datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
## Hour,Minute,Second=timeStamp.split(":")  
## fileName="Attendance\Attendance\_"+date+"\_"+Hour+"-"+Minute+"-"+Second+".csv"  
## attendance.to\_csv(fileName,index=False)  
## cam.release()  
## cv2.destroyAllWindows()  
## #print(attendance)  
## res=attendance  
## message2.configure(text= res)  
  
   
clearButton = tk.Button(window**,** text="Clear"**,** command=clear **,**fg="red" **,**bg="yellow" **,**width=**20 ,**height=**2 ,**activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
clearButton.place(x=**950,** y=**200**)  
clearButton2 = tk.Button(window**,** text="Clear"**,** command=clear2 **,**fg="red" **,**bg="yellow" **,**width=**20 ,**height=**2,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
clearButton2.place(x=**950,** y=**300**)   
takeImg = tk.Button(window**,** text="Take Images"**,** command=TakeImages **,**fg="red" **,**bg="yellow" **,**width=**10 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
takeImg.place(x=**350,** y=**500**)  
retakeImg = tk.Button(window**,** text="Retrain"**,** command=TakeImages1 **,**fg="red" **,**bg="yellow" **,**width=**10 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
retakeImg.place(x=**50,** y=**500**)  
trainImg = tk.Button(window**,** text="Train Images"**,** command=TrainImages **,**fg="red" **,**bg="yellow" **,**width=**10 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
trainImg.place(x=**500,** y=**500**)  
trackImg = tk.Button(window**,** text="Track Images"**,** command=TrackImages **,**fg="red" **,**bg="yellow" **,**width=**10 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
trackImg.place(x=**800,** y=**500**)  
Attendance = tk.Button(window**,** text="Calculate"**,** command=calculate **,**fg="red" **,**bg="yellow" **,**width=**10 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
Attendance.place(x=**200,** y=**500**)  
quitWindow = tk.Button(window**,** text="Quit"**,** command=window.destroy **,**fg="red" **,**bg="yellow" **,**width=**20 ,**height=**3,** activebackground = "Red" **,**font=('times'**, 15,** ' bold '))  
quitWindow.place(x=**1100,** y=**500**)  
copyWrite = tk.Text(window**,** background=window.cget("background")**,** borderwidth=**0,**font=('times'**, 30,** 'italic bold underline'))  
copyWrite.tag\_configure("superscript"**,** offset=**10**)  
copyWrite.insert("insert"**,** "Developed by Ashish"**,**""**,** "TEAM"**,** "superscript")  
copyWrite.configure(state="disabled"**,**fg="red" )  
copyWrite.pack(side="left")  
copyWrite.place(x=**800,** y=**750**)  
   
window.mainloop()