**Main.py**

import tkinter as tk

from tkinter import \*

import cv2

import csv

import os

import numpy as np

from PIL import Image, ImageTk

import pandas as pd

import datetime

import time

# Window is our Main frame of system

window = tk.Tk()

window.title("FaceReqG-Face Recognition Based Attendance Management System")

window.geometry('1280x720')

window.configure(background='grey80')

# GUI for manually fill attendance

def manually\_fill():

global sb

sb = tk.Tk()

# sb.iconbitmap('AMS.ico')

sb.title("Enter subject name...")

sb.geometry('580x320')

sb.configure(background='grey80')

def err\_screen\_for\_subject():

def ec\_delete():

ec.destroy()

global ec

ec = tk.Tk()

ec.geometry('300x100')

# ec.iconbitmap('AMS.ico')

ec.title('Warning!!')

ec.configure(background='snow')

Label(ec, text='Please enter your subject name!!!', fg='red',

bg='white', font=('times', 16, ' bold ')).pack()

Button(ec, text='OK', command=ec\_delete, fg="black", bg="lawn green", width=9, height=1, activebackground="Red",

font=('times', 15, ' bold ')).place(x=90, y=50)

def fill\_attendance():

ts = time.time()

Date = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

Hour, Minute, Second = timeStamp.split(":")

# Creatting csv of attendance

# Create table for Attendance

date\_for\_DB = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')

global subb

subb = SUB\_ENTRY.get()

DB\_table\_name = str(subb + "\_" + Date + "\_Time\_" +

Hour + "\_" + Minute + "\_" + Second)

import pymysql.connections

# Connect to the database

try:

global cursor

connection = pymysql.connect(

host='localhost', user='root', password='', db='manually\_fill\_attendance')

cursor = connection.cursor()

except Exception as e:

print(e)

sql = "CREATE TABLE " + DB\_table\_name + """

(ID INT NOT NULL AUTO\_INCREMENT,

ENROLLMENT varchar(100) NOT NULL,

NAME VARCHAR(50) NOT NULL,

DATE VARCHAR(20) NOT NULL,

TIME VARCHAR(20) NOT NULL,

PRIMARY KEY (ID)

);

"""

try:

cursor.execute(sql) # for create a table

except Exception as ex:

print(ex) #

if subb == '':

err\_screen\_for\_subject()

else:

sb.destroy()

MFW = tk.Tk()

# MFW.iconbitmap('AMS.ico')

MFW.title("Manually attendance of " + str(subb))

MFW.geometry('880x470')

MFW.configure(background='grey80')

def del\_errsc2():

errsc2.destroy()

def err\_screen1():

global errsc2

errsc2 = tk.Tk()

errsc2.geometry('330x100')

# errsc2.iconbitmap('AMS.ico')

errsc2.title('Warning!!')

errsc2.configure(background='grey80')

Label(errsc2, text='Please enter Student & Enrollment!!!', fg='black', bg='white',

font=('times', 16, ' bold ')).pack()

Button(errsc2, text='OK', command=del\_errsc2, fg="black", bg="lawn green", width=9, height=1,

activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

def testVal(inStr, acttyp):

if acttyp == '1': # insert

if not inStr.isdigit():

return False

return True

ENR = tk.Label(MFW, text="Enter Enrollment", width=15, height=2, fg="black", bg="grey",

font=('times', 15))

ENR.place(x=30, y=100)

STU\_NAME = tk.Label(MFW, text="Enter Student name", width=15, height=2, fg="black", bg="grey",

font=('times', 15))

STU\_NAME.place(x=30, y=200)

global ENR\_ENTRY

ENR\_ENTRY = tk.Entry(MFW, width=20, validate='key',

bg="white", fg="black", font=('times', 23))

ENR\_ENTRY['validatecommand'] = (

ENR\_ENTRY.register(testVal), '%P', '%d')

ENR\_ENTRY.place(x=290, y=105)

def remove\_enr():

ENR\_ENTRY.delete(first=0, last=22)

STUDENT\_ENTRY = tk.Entry(

MFW, width=20, bg="white", fg="black", font=('times', 23))

STUDENT\_ENTRY.place(x=290, y=205)

def remove\_student():

STUDENT\_ENTRY.delete(first=0, last=22)

# get important variable

def enter\_data\_DB():

ENROLLMENT = ENR\_ENTRY.get()

STUDENT = STUDENT\_ENTRY.get()

if ENROLLMENT == '':

err\_screen1()

elif STUDENT == '':

err\_screen1()

else:

time = datetime.datetime.fromtimestamp(

ts).strftime('%H:%M:%S')

Hour, Minute, Second = time.split(":")

Insert\_data = "INSERT INTO " + DB\_table\_name + \

" (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"

VALUES = (str(ENROLLMENT), str(

STUDENT), str(Date), str(time))

try:

cursor.execute(Insert\_data, VALUES)

except Exception as e:

print(e)

ENR\_ENTRY.delete(first=0, last=22)

STUDENT\_ENTRY.delete(first=0, last=22)

def create\_csv():

import csv

cursor.execute("select \* from " + DB\_table\_name + ";")

csv\_name = 'Attendance/Manually Attendance/'+DB\_table\_name+'.csv'

with open(csv\_name, "w") as csv\_file:

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow(

[i[0] for i in cursor.description]) # write headers

csv\_writer.writerows(cursor)

O = "CSV created Successfully"

Notifi.configure(text=O, bg="Green", fg="white",

width=33, font=('times', 19, 'bold'))

Notifi.place(x=180, y=380)

import csv

import tkinter

root = tkinter.Tk()

root.title("Attendance of " + subb)

root.configure(background='grey80')

with open(csv\_name, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=18, height=1, fg="black", font=('times', 13, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

Notifi = tk.Label(MFW, text="CSV created Successfully", bg="Green", fg="white", width=33,

height=2, font=('times', 19, 'bold'))

c1ear\_enroll = tk.Button(MFW, text="Clear", command=remove\_enr, fg="white", bg="black", width=10,

height=1,

activebackground="white", font=('times', 15, ' bold '))

c1ear\_enroll.place(x=690, y=100)

c1ear\_student = tk.Button(MFW, text="Clear", command=remove\_student, fg="white", bg="black", width=10,

height=1,

activebackground="white", font=('times', 15, ' bold '))

c1ear\_student.place(x=690, y=200)

DATA\_SUB = tk.Button(MFW, text="Enter Data", command=enter\_data\_DB, fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="white", font=('times', 15, ' bold '))

DATA\_SUB.place(x=170, y=300)

MAKE\_CSV = tk.Button(MFW, text="Convert to CSV", command=create\_csv, fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="white", font=('times', 15, ' bold '))

MAKE\_CSV.place(x=570, y=300)

def attf():

import subprocess

subprocess.Popen(

r'explorer /select,"Attendance\Manually Attendance\"')

attf = tk.Button(MFW, text="Check Sheets", command=attf, fg="white", bg="black",

width=12, height=1, activebackground="white", font=('times', 14, ' bold '))

attf.place(x=730, y=410)

MFW.mainloop()

SUB = tk.Label(sb, text="Enter Subject : ", width=15, height=2,

fg="black", bg="grey80", font=('times', 15, ' bold '))

SUB.place(x=30, y=100)

global SUB\_ENTRY

SUB\_ENTRY = tk.Entry(sb, width=20, bg="white",

fg="black", font=('times', 23))

SUB\_ENTRY.place(x=250, y=105)

fill\_manual\_attendance = tk.Button(sb, text="Fill Attendance", command=fill\_attendance, fg="black", bg="SkyBlue1", width=20, height=2,

activebackground="white", font=('times', 15, ' bold '))

fill\_manual\_attendance.place(x=250, y=160)

sb.mainloop()

# For clear textbox

def clear():

txt.delete(first=0, last=22)

def clear1():

txt2.delete(first=0, last=22)

def del\_sc1():

sc1.destroy()

def err\_screen():

global sc1

sc1 = tk.Tk()

sc1.geometry('300x100')

# sc1.iconbitmap('AMS.ico')

sc1.title('Warning!!')

sc1.configure(background='grey80')

Label(sc1, text='Enrollment & Name required!!!', fg='black',

bg='white', font=('times', 16)).pack()

Button(sc1, text='OK', command=del\_sc1, fg="black", bg="lawn green", width=9,

height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

# Error screen2

def del\_sc2():

sc2.destroy()

def err\_screen1():

global sc2

sc2 = tk.Tk()

sc2.geometry('300x100')

# sc2.iconbitmap('AMS.ico')

sc2.title('Warning!!')

sc2.configure(background='grey80')

Label(sc2, text='Please enter your subject name!!!', fg='black',

bg='white', font=('times', 16)).pack()

Button(sc2, text='OK', command=del\_sc2, fg="black", bg="lawn green", width=9,

height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

# For take images for datasets

def take\_img():

l1 = txt.get()

l2 = txt2.get()

if l1 == '':

err\_screen()

elif l2 == '':

err\_screen()

else:

try:

cam = cv2.VideoCapture(0)

detector = cv2.CascadeClassifier(

'haarcascade\_frontalface\_default.xml')

Enrollment = txt.get()

Name = txt2.get()

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

cv2.imwrite("TrainingImage/ " + Name + "." + Enrollment + '.' + str(sampleNum) + ".jpg",

gray)

print("Images Saved for Enrollment :"+Enrollment)

cv2.imshow('Frame', img)

# wait for 100 miliseconds

if cv2.waitKey(1) & 0xFF == ord('q'):

break

#

# # break if the sample number is morethan 100

elif sampleNum > 70:

break

cam.release()

cv2.destroyAllWindows()

ts = time.time()

Date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

row = [Enrollment, Name, Date, Time]

with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:

writer = csv.writer(csvFile, delimiter=',')

writer.writerow(row)

csvFile.close()

res = "Images Saved for Enrollment : " + Enrollment + " Name : " + Name

Notification.configure(

text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

except FileExistsError as F:

f = 'Student Data already exists'

Notification.configure(text=f, bg="Red", width=21)

Notification.place(x=450, y=400)

# for choose subject and fill attendance

def subjectchoose():

def Fillattendances():

def subjectchoose():

def Fillattendances():

sub = tx.get()

now = time.time() # For calculate seconds of video

future = now + 20

if time.time() < future:

if sub == '':

err\_screen1()

else:

recognizer = cv2.face.LBPHFaceRecognizer\_create() # cv2.createLBPHFaceRecognizer()

try:

recognizer.read("TrainingImageLabel\Trainner.yml")

except:

e = 'Model not found,Please train model'

Notifica.configure(

text=e, bg="red", fg="black", width=33, font=('times', 15, 'bold'))

Notifica.place(x=20, y=250)

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 = ['Enrollment','Name','Date','Time']

attendance = pd.DataFrame(columns = col\_names)

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 < 70):

global Subject

Subject =tx.get()

ts = time.time()

date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

aa=df.loc[df['Enrollment'] == Id]['Name'].values

tt=str(Id)+"-"+aa

attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]

else:

Id = 'Unknown'

tt = str(Id)

cv2.rectangle(

im, (x, y), (x + w, y + h), (0, 25, 255), 7)

cv2.putText(im, str(tt), (x + h, y),

font, 1, (0, 25, 255), 4)

if time.time() > future:

break

attendance = attendance.drop\_duplicates(

['Enrollment'], keep='first')

cv2.imshow('Filling attedance..', im)

key = cv2.waitKey(30) & 0xff

if key == 27:

break

ts = time.time()

date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(

ts).strftime('%H:%M:%S')

Hour, Minute, Second = timeStamp.split(":")

fileName = "Attendance/" + Subject + "\_" + date + \

"\_" + Hour + "-" + Minute + "-" + Second + ".csv"

attendance = attendance.drop\_duplicates(

['Enrollment'], keep='first')

print(attendance)

attendance.to\_csv(fileName, index=False)

# Create table for Attendance

date\_for\_DB = datetime.datetime.fromtimestamp(

ts).strftime('%Y\_%m\_%d')

DB\_Table\_name = str(

Subject + "\_" + date\_for\_DB + "\_Time\_" + Hour + "\_" + Minute + "\_" + Second)

import pymysql.connections

# Connect to the database

try:

global cursor

connection = pymysql.connect(

host='localhost', user='root', password='', db='Face\_reco\_fill')

cursor = connection.cursor()

except Exception as e:

print(e)

sql = "CREATE TABLE " + DB\_Table\_name + """

(ID INT NOT NULL AUTO\_INCREMENT,

ENROLLMENT varchar(100) NOT NULL,

NAME VARCHAR(50) NOT NULL,

DATE VARCHAR(20) NOT NULL,

TIME VARCHAR(20) NOT NULL,

PRIMARY KEY (ID)

);

"""

# Now enter attendance in Database

insert\_data = "INSERT INTO " + DB\_Table\_name + \

" (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"

VALUES = (str(Id), str(aa), str(date), str(timeStamp))

try:

cursor.execute(sql) # for create a table

# For insert data into table

cursor.execute(insert\_data, VALUES)

except Exception as ex:

print(ex) #

M = 'Attendance filled Successfully'

Notifica.configure(text=M, bg="Green", fg="white",

width=33, font=('times', 15, 'bold'))

Notifica.place(x=20, y=250)

cam.release()

cv2.destroyAllWindows()

import csv

import tkinter

root = tkinter.Tk()

root.title("Attendance of " + Subject)

root.configure(background='grey80')

cs = 'FaceReqG' + fileName

with open(cs, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

print(attendance)

# windo is frame for subject chooser

windo = tk.Tk()

# windo.iconbitmap('AMS.ico')

windo.title("Enter subject name...")

windo.geometry('580x320')

windo.configure(background='grey80')

Notifica = tk.Label(windo, text="Attendance filled Successfully", bg="Green", fg="white", width=33,

height=2, font=('times', 15, 'bold'))

def Attf():

import subprocess

subprocess.Popen(

r'explorer /select,"Attendance\"')

attf = tk.Button(windo, text="Check Sheets", command=Attf, fg="white", bg="black",

width=12, height=1, activebackground="white", font=('times', 14, ' bold '))

attf.place(x=430, y=255)

sub = tk.Label(windo, text="Enter Subject : ", width=15, height=2,

fg="black", bg="grey", font=('times', 15, ' bold '))

sub.place(x=30, y=100)

tx = tk.Entry(windo, width=20, bg="white",

fg="black", font=('times', 23))

tx.place(x=250, y=105)

fill\_a = tk.Button(windo, text="Fill Attendance", fg="white", command=Fillattendances, bg="SkyBlue1", width=20, height=2,

activebackground="white", font=('times', 15, ' bold '))

fill\_a.place(x=250, y=160)

windo.mainloop()

def admin\_panel():

win = tk.Tk()

# win.iconbitmap('AMS.ico')

win.title("LogIn")

win.geometry('880x420')

win.configure(background='grey80')

def log\_in():

username = un\_entr.get()

password = pw\_entr.get()

if username == 'FaceReqG':

if password == 'FaceReqG':

win.destroy()

import csv

import tkinter

root = tkinter.Tk()

root.title("Student Details")

root.configure(background='grey80')

cs = 'StudentDetails/StudentDetails.csv'

with open(cs, newline="") as file:

reader = csv.reader(file)

r = 0

for col in reader:

c = 0

for row in col:

# i've added some styling

label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),

bg="white", text=row, relief=tkinter.RIDGE)

label.grid(row=r, column=c)

c += 1

r += 1

root.mainloop()

else:

valid = 'Incorrect ID or Password'

Nt.configure(text=valid, bg="red", fg="white",

width=38, font=('times', 19, 'bold'))

Nt.place(x=120, y=350)

else:

valid = 'Incorrect ID or Password'

Nt.configure(text=valid, bg="red", fg="white",

width=38, font=('times', 19, 'bold'))

Nt.place(x=120, y=350)

Nt = tk.Label(win, text="Attendance filled Successfully", bg="Green", fg="white", width=40,

height=2, font=('times', 19, 'bold'))

# Nt.place(x=120, y=350)

un = tk.Label(win, text="Enter username : ", width=15, height=2, fg="black", bg="grey",

font=('times', 15, ' bold '))

un.place(x=30, y=50)

pw = tk.Label(win, text="Enter password : ", width=15, height=2, fg="black", bg="grey",

font=('times', 15, ' bold '))

pw.place(x=30, y=150)

def c00():

un\_entr.delete(first=0, last=22)

un\_entr = tk.Entry(win, width=20, bg="white", fg="black",

font=('times', 23))

un\_entr.place(x=290, y=55)

def c11():

pw\_entr.delete(first=0, last=22)

pw\_entr = tk.Entry(win, width=20, show="\*", bg="white",

fg="black", font=('times', 23))

pw\_entr.place(x=290, y=155)

c0 = tk.Button(win, text="Clear", command=c00, fg="white", bg="black", width=10, height=1,

activebackground="white", font=('times', 15, ' bold '))

c0.place(x=690, y=55)

c1 = tk.Button(win, text="Clear", command=c11, fg="white", bg="black", width=10, height=1,

activebackground="white", font=('times', 15, ' bold '))

c1.place(x=690, y=155)

Login = tk.Button(win, text="LogIn", fg="black", bg="SkyBlue1", width=20,

height=2,

activebackground="Red", command=log\_in, font=('times', 15, ' bold '))

Login.place(x=290, y=250)

win.mainloop()

# For train the model

def trainimg():

recognizer = cv2.face.LBPHFaceRecognizer\_create()

global detector

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

try:

global faces, Id

faces, Id = getImagesAndLabels("TrainingImage")

except Exception as e:

l = 'please make "TrainingImage" folder & put Images'

Notification.configure(text=l, bg="SpringGreen3",

width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

recognizer.train(faces, np.array(Id))

try:

recognizer.save("TrainingImageLabel\Trainner.yml")

except Exception as e:

q = 'Please make "TrainingImageLabel" folder'

Notification.configure(text=q, bg="SpringGreen3",

width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

res = "Model Trained" # +",".join(str(f) for f in Id)

Notification.configure(text=res, bg="olive drab",

width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

def getImagesAndLabels(path):

imagePaths = [os.path.join(path, f) for f in os.listdir(path)]

# create empth face list

faceSamples = []

# 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 = detector.detectMultiScale(imageNp)

# If a face is there then append that in the list as well as Id of it

for (x, y, w, h) in faces:

faceSamples.append(imageNp[y:y + h, x:x + w])

Ids.append(Id)

return faceSamples, Ids

window.grid\_rowconfigure(0, weight=1)

window.grid\_columnconfigure(0, weight=1)

# window.iconbitmap('AMS.ico')

def on\_closing():

from tkinter import messagebox

if messagebox.askokcancel("Quit", "Do you want to quit?"):

window.destroy()

window.protocol("WM\_DELETE\_WINDOW", on\_closing)

message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System", bg="black", fg="white", width=50,

height=3, font=('times', 30, ' bold '))

message.place(x=80, y=20)

Notification = tk.Label(window, text="All things good", bg="Green", fg="white", width=15,

height=3, font=('times', 17))

lbl = tk.Label(window, text="Enter Enrollment : ", width=20, height=2,

fg="black", bg="grey", font=('times', 15, 'bold'))

lbl.place(x=200, y=200)

def testVal(inStr, acttyp):

if acttyp == '1': # insert

if not inStr.isdigit():

return False

return True

txt = tk.Entry(window, validate="key", width=20, bg="white",

fg="black", font=('times', 25))

txt['validatecommand'] = (txt.register(testVal), '%P', '%d')

txt.place(x=550, y=210)

lbl2 = tk.Label(window, text="Enter Name : ", width=20, fg="black",

bg="grey", height=2, font=('times', 15, ' bold '))

lbl2.place(x=200, y=300)

txt2 = tk.Entry(window, width=20, bg="white",

fg="black", font=('times', 25))

txt2.place(x=550, y=310)

clearButton = tk.Button(window, text="Clear", command=clear, fg="white", bg="black",

width=10, height=1, activebackground="white", font=('times', 15, ' bold '))

clearButton.place(x=950, y=210)

clearButton1 = tk.Button(window, text="Clear", command=clear1, fg="white", bg="black",

width=10, height=1, activebackground="white", font=('times', 15, ' bold '))

clearButton1.place(x=950, y=310)

AP = tk.Button(window, text="Check Registered students", command=admin\_panel, fg="black",

bg="SkyBlue1", width=19, height=1, activebackground="white", font=('times', 15, ' bold '))

AP.place(x=990, y=410)

takeImg = tk.Button(window, text="Take Images", command=take\_img, fg="black", bg="SkyBlue1",

width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

takeImg.place(x=90, y=500)

trainImg = tk.Button(window, text="Train Images", fg="black", command=trainimg, bg="SkyBlue1",

width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

trainImg.place(x=390, y=500)

FA = tk.Button(window, text="Automatic Attendance", fg="black", command=subjectchoose,

bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

FA.place(x=690, y=500)

quitWindow = tk.Button(window, text="Manually Fill Attendance", command=manually\_fill, fg="black",

bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))

quitWindow.place(x=990, y=500)

window.mainloop()

**testing.py**

import cv2

import numpy as np

recognizer = cv2.createLBPHFaceRecognizer()

recognizer.read('TrainingImageLabel/trainner.yml')

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

font = cv2.FONT\_HERSHEY\_SIMPLEX

cam = cv2.VideoCapture(0)

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:

Id, conf = recognizer.predict(gray[y:y+h, x:x+w])

# # else:

# # Id="Unknown"

# cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)

cv2.rectangle(im, (x, y), (x + w, y + h), (0, 260, 0), 7)

cv2.putText(im, str(Id), (x, y-40), font, 2, (255, 255, 255), 3)

# cv2.putText(im, str(Id), (x + h, y), font, 1, (0, 260, 0), 2)

cv2.imshow('im', im)

if cv2.waitKey(10) & 0xFF == ord('q'):

break

cam.release()

cv2.destroyAllWindows()

**training.py**

import cv2

import os

import numpy as np

from PIL import Image

#

# recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer = cv2.face.LBPHFaceRecognizer\_create()

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

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)]

# create empth face list

faceSamples = []

# 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 = detector.detectMultiScale(imageNp)

# If a face is there then append that in the list as well as Id of it

for (x, y, w, h) in faces:

faceSamples.append(imageNp[y:y+h, x:x+w])

Ids.append(Id)

return faceSamples, Ids

faces, Ids = getImagesAndLabels('TrainingImage')

recognizer.train(faces, np.array(Ids))

recognizer.save('TrainingImageLabel/trainner.yml')

