

SWE4010 – ARTIFICIAL INTELLIGENCE

PROJECT REPORT

PROJECT TITLE:

**FACE RECOGNITION BASED ATTENDENCE MONITORING SYSTEM**

TEAM MEMBERS:

**Saheli Das - 19MIS1164   
Niranjana O -19MIS1156   
 Akash R - 19MIS1084   
Khushi Kantula -19MIS1186   
Chandini -19MIS1002**

ABSTRACT

From now on, a computer-based student attendance checking system is required that supports the faculty to keep records of attendance. We have used an intelligent attendance system based on face recognition in this project. We have proposed to implement a "Smart Attendance System for Face Recognition" through this large applications are incorporated. The present implementation includes facial identification that is time saving and eradicates the possibilities of proxy attendance due to the facial authorization.

Attendance marking is a most time consuming task in every institutes. Current attendance system is mostly based on RFID, Fingerprint and even Notebook.

Those systems require physical interaction of human with the devices, everyone should wait until previous user mark the attendance and queue is essential to control the crowd.

Objective of our project is to develop a Face recognition based attendance system by using AI. It captures the faces with cameras and processes every image into datasets.

With the help of datasets system identifies which face belongs to which name then mark the attendance in the datasheet. Then export the attendance as excel sheet then save it to specific location. All captured images and datasets are saved to the servers.

The system implemented uses LBPH face recognizer to identify the face of the person in real time. Eigen faces and Fisher faces are affected both by light and we cannot ensure perfect light conditions in real life. An improvement in the LBPH faces recognizer to overcome this problem. This system compares the image of the test and the training image and determines who is and is not present. The attendance data is stored in an excel sheet that is automatically updated in the system.

LITERATURE SURVEY

When the Student images have been stored in the database. The raspberry pi camera module is placing the student entering the classroom. The USB camera module captures the student image. The system will automatically update the student presence in the class to the student's database.

The laptop of the web camera has captured the image. Each student's face is stored in the database. The image of the students for further process. Then detected face images are compared with the image of the student's database. And the recognition process. If the image is matched with the database. The student is marked as present. The student database is collected and stored to the pi. The camera will be connected to the raspberry pi module camera will be placed in front of the class. It captures the image of the student. Who are present in the class is used for face detection. Then the detected face is compared with the stored data of every student. Then recognize and mark the attendance. The camera will now capture your image. If the image detected matches the sample image for the database. The attendance is marked as present on the LCD.

The system employed the LBPH algorithm to extract the characteristics and the SVM classifier for classification purposes this document used an 80- person database (NITW database) with approximately 20 images of each individual collected for the project. This document sets out some performance evaluation conditions when combining LBPH and distance classifier, the false positive rate is 25 %, the object distance for correct recognition must be 4 feet, the training time being 563 milliseconds, 95 %of recognition percentage for static images, the recognition percentage (real-time video) was 78 %, the occluded faces 2.3% In Microsoft Visual C #and the EmguCV container the GUI is developed using the WinForms application.

PROPOSED WORK

A python GUI integrated attendance system using face recognition to take attendance.

In this python project, we have made an attendance system which takes attendance by using face recognition technique. We have also integrated it with GUI (Graphical user interface) so it can be easy to use by anyone. GUI for this project is also made on python using tkinter.

Face recognition can be achieved with the help of a learning concept of training and then testing the model with a given set of images.

Training rules are used to ensure the output decisions criteria and training algorithm can be used to get some input from the data to match the appropriate output type. So, the algorithm and rules are used to simplify the process of learning. The system uses the information gathered from the data to get results.

The precision and accuracy of the algorithm are verified by using a test set of images.

The LBPH algorithm typically makes use of 4 parameters:

* Radius: The distance of the circular local binary pattern from the center pixel to its circumference and usually takes a value of 1.
* Neighbors: The number of data points within a circular local binary pattern. Usually, the value of 8.
* Grid X: The number of cells in the horizontal plane is usually a value of 8.
* Grid Y: The number of cells in the vertical plane is usually a value of 8.

Given the above-mentioned parameters, LBPH works as follows;

A data set is created by taking images with a camera or taking images that are saved, and then provisioning a unique identifier or name of the person in the image and then adding the images to a database. It is recommended to take many samples from a single individual. A portion of the data set is used for the training of the algorithm, while the rest is used for testing.

Using a circular neighborhood concept (which takes non-integer pixel points around a selected area), the number of appearances of LBP codes in the image is put together to form a histogram. The classification is then carried out through the calculation of the basic similarities of the histograms under comparison.

This histogram contains a description of an individual at three different levels: at a pixel-level, labels are combined in a small area to create a regional level; the regional histograms in combination build a general description of the person.

The face recognition algorithm generally operates in one of two modes:

* Authentication of a facial image: This mode does facial recognition by a 1x1 comparison. The comparison is done between an input image and a specific image within the database. In many cases, this is the face that requires authentication at the time of this mode of facial recognition.
* Face recognition: in this mode, it is a 1xN, a comparison of the input face image with all the pictures that have been saved in the database to output the images of the user which conforms to the input face image.

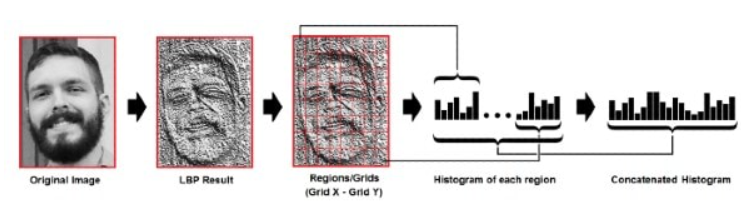


FIGURE (The above image illustrates the extraction)

RESULTS & DISCUSSIONS

User Interface:

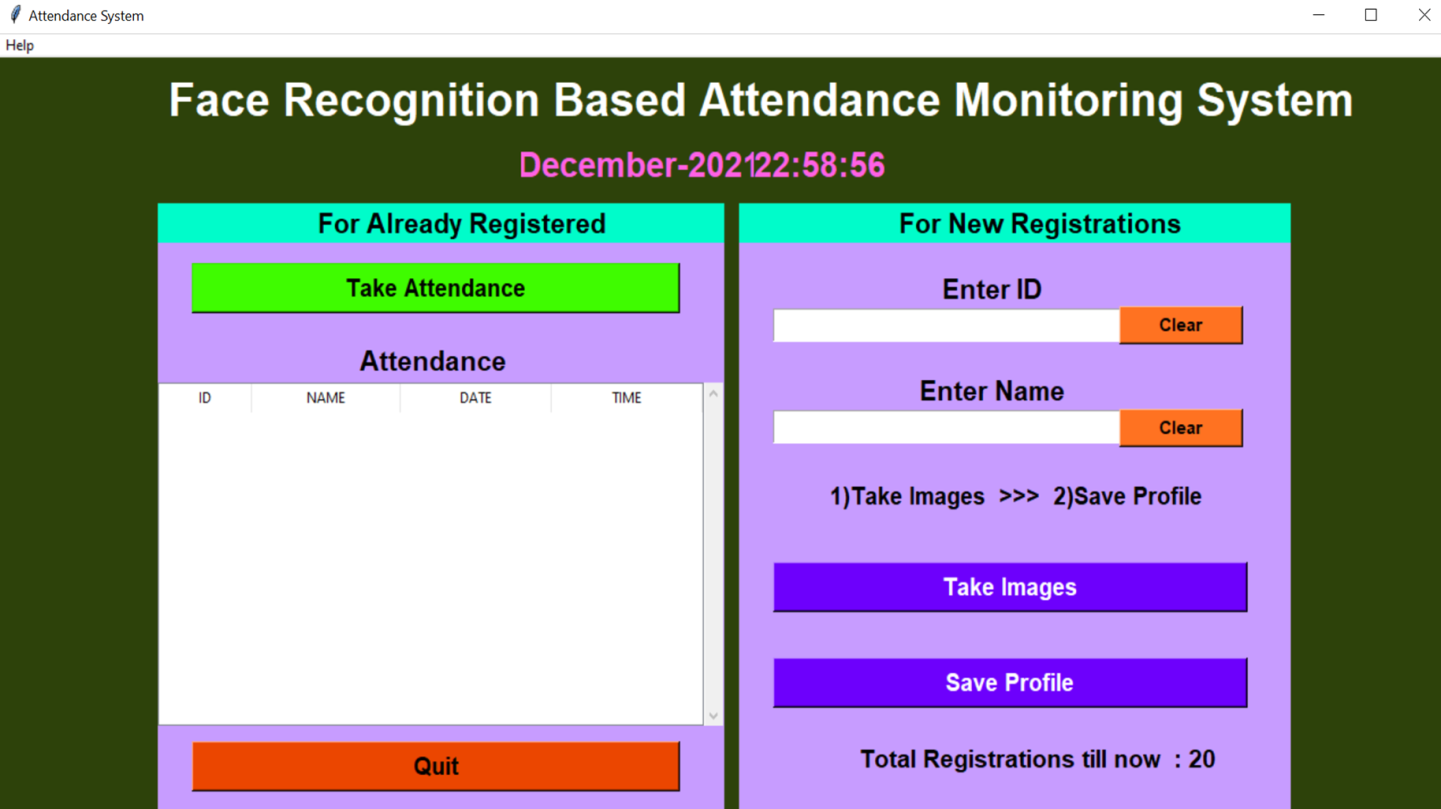
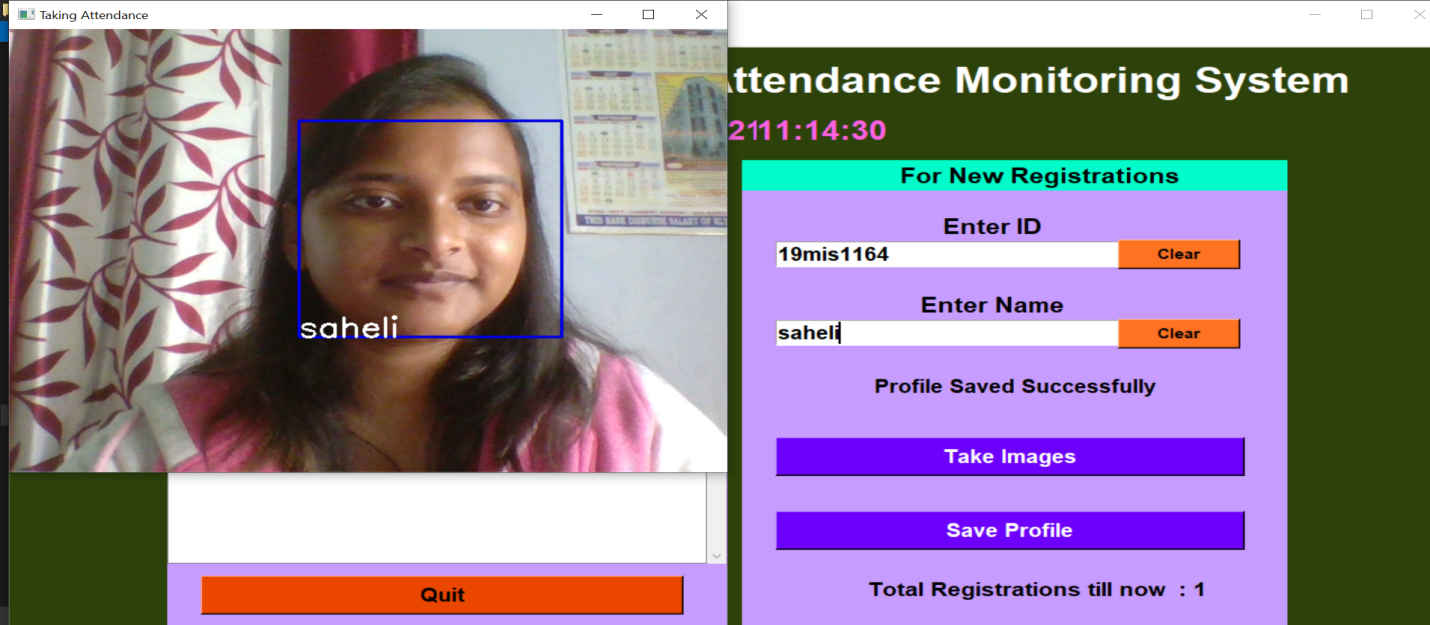


Figure (GUI of the application)

Multiple images of the students are captured and the images are pre-processed for detecting only the faces of the students.



The captured images of the students are stored in a local database. The stored images are trained and are assigned corresponding labels such as Id and name.



The corresponding attendance of the students is stored in an excel sheet.

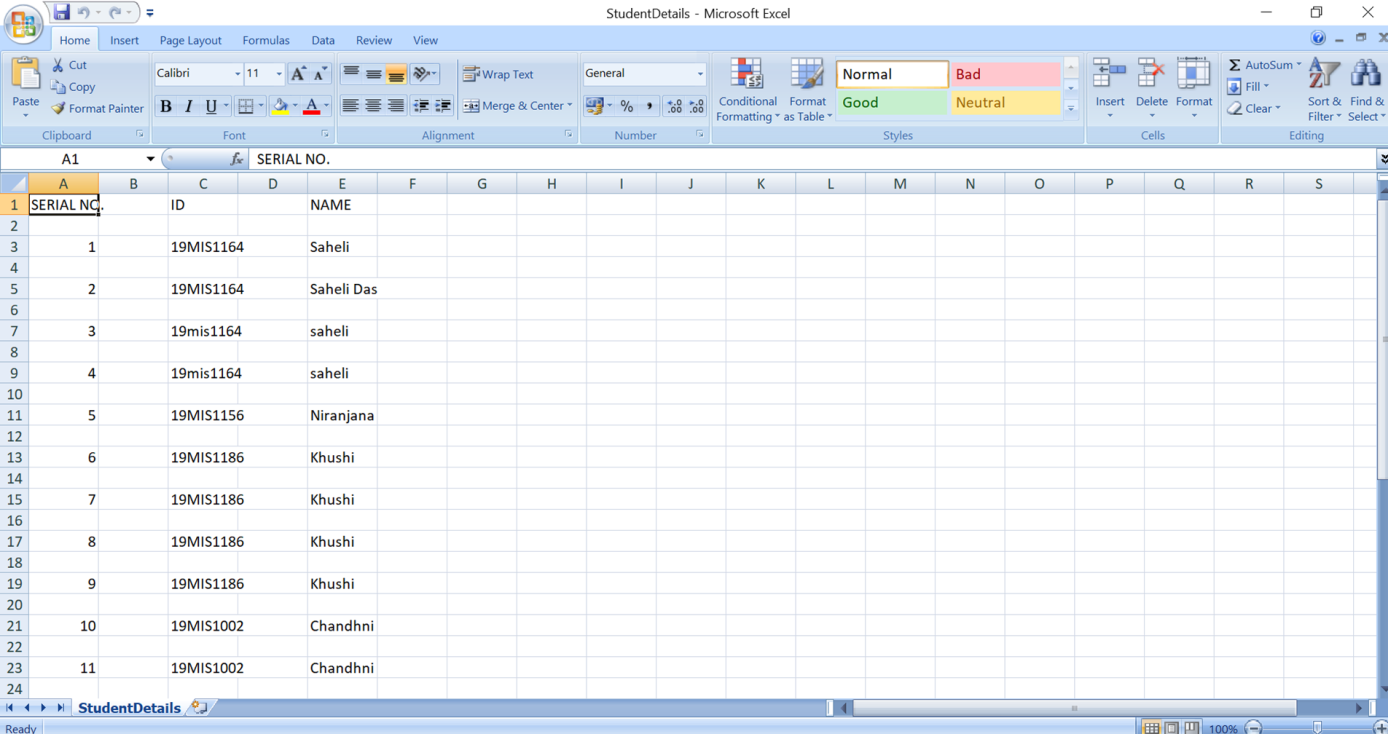


Figure (Student details stored in the data sheet)

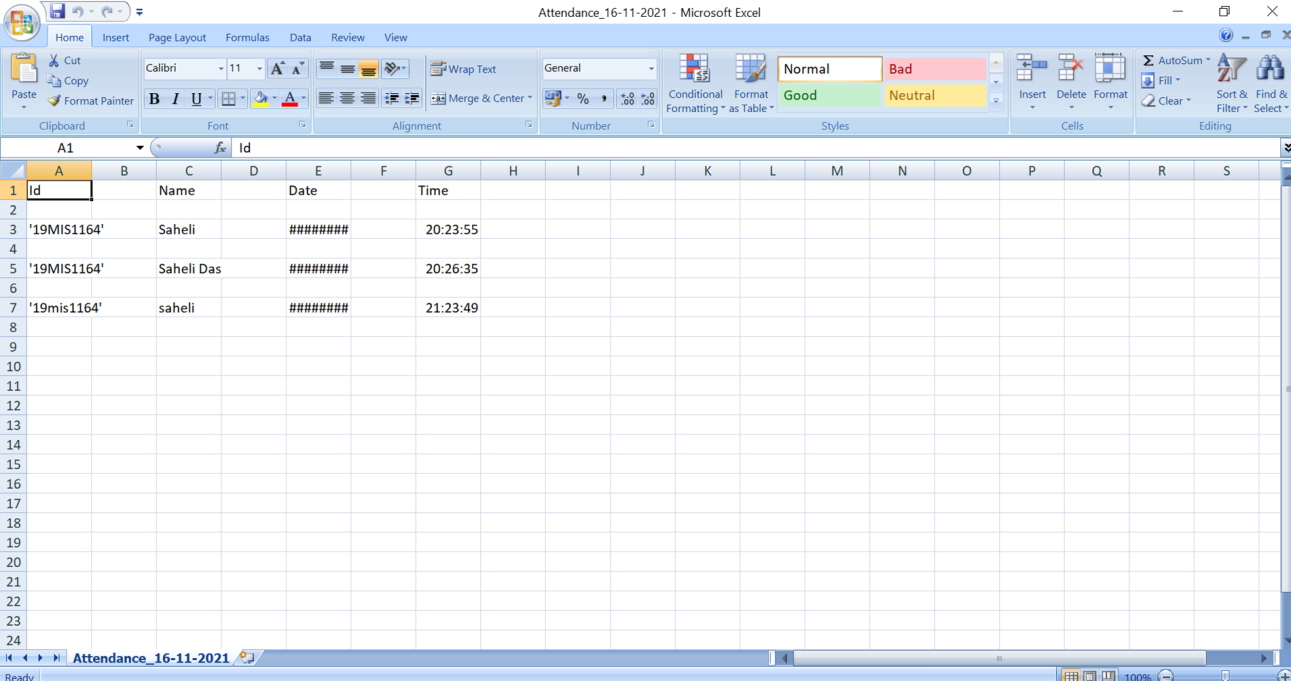


Figure (Attendance sheet)

CONCLUSION

Before the development of this project. There are many loopholes in the process of taking attendance using the old methods which caused many troubles to most of the institutions. Therefore, the facial recognition feature embedded in the attendance monitoring system can not only ensure attendance to be taken accurately and also eliminated the flaws in the previous system.

By using technology to conquer the defects cannot merely save resources but also reduces human intervention in the whole process by handling the entire complicated task to the machine. The only cost to this solution is to have sufficient space in to store all the faces into the database storage. Fortunately, there is such existence of micro SD that can compensate with the volume of the data. In this project, the face database is successfully built. Apart from that, the face recognizing system is also working well.

At the end, the system not only resolve troubles that exist in the old model but also provide convenience to the user to access the information collected by saving the attendance sheet automatically.

APPENDIX

# IMPORTING #

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mess

import tkinter.simpledialog as tsd

import cv2,os

import csv

import numpy as np

from PIL import Image

import pandas as pd

import datetime

import time

# FUNCTIONS #

def assure\_path\_exists(path):

dir = os.path.dirname(path)

if not os.path.exists(dir):

os.makedirs(dir)

#

def tick():

time\_string = time.strftime('%H:%M:%S')

clock.config(text=time\_string)

clock.after(200,tick)

#

def check\_haarcascadefile():

exists = os.path.isfile("haarcascade\_frontalface\_default.xml")

if exists:

pass

else:

mess.\_show(title='Some file missing', message='Please contact us for help')

window.destroy()

#

def save\_pass():

assure\_path\_exists("TrainingImageLabel/")

exists1 = os.path.isfile("TrainingImageLabel\psd.txt")

if exists1:

tf = open("TrainingImageLabel\psd.txt", "r")

key = tf.read()

else:

master.destroy()

new\_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show='\*')

if new\_pas == None:

mess.\_show(title='No Password Entered', message='Password not set!! Please try again')

else:

tf = open("TrainingImageLabel\psd.txt", "w")

tf.write(new\_pas)

mess.\_show(title='Password Registered', message='New password was registered successfully!!')

return

op = (old.get())

newp= (new.get())

nnewp = (nnew.get())

if (op == key):

if(newp == nnewp):

txf = open("TrainingImageLabel\psd.txt", "w")

txf.write(newp)

else:

mess.\_show(title='Error', message='Confirm new password again!!!')

return

else:

mess.\_show(title='Wrong Password', message='Please enter correct old password.')

return

mess.\_show(title='Password Changed', message='Password changed successfully!!')

master.destroy()

#

def change\_pass():

global master

master = tk.Tk()

master.geometry("400x160")

master.resizable(False,False)

master.title("Change Password")

master.configure(background="white")

lbl4 = tk.Label(master,text=' Enter Old Password',bg='white',font=('comic', 12, ' bold '))

lbl4.place(x=10,y=10)

global old

old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('comic', 12, ' bold '),show='\*')

old.place(x=180,y=10)

lbl5 = tk.Label(master, text=' Enter New Password', bg='white', font=('comic', 12, ' bold '))

lbl5.place(x=10, y=45)

global new

new = tk.Entry(master, width=25, fg="black",relief='solid', font=('comic', 12, ' bold '),show='\*')

new.place(x=180, y=45)

lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('comic', 12, ' bold '))

lbl6.place(x=10, y=80)

global nnew

nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('comic', 12, ' bold '),show='\*')

nnew.place(x=180, y=80)

cancel=tk.Button(master,text="Cancel", command=master.destroy ,fg="black" ,bg="red" ,height=1,width=25 , activebackground = "white" ,font=('comic', 10, ' bold '))

cancel.place(x=200, y=120)

save1 = tk.Button(master, text="Save", command=save\_pass, fg="black", bg="#00fcca", height = 1,width=25, activebackground="white", font=('comic', 10, ' bold '))

save1.place(x=10, y=120)

master.mainloop()

#

def psw():

assure\_path\_exists("TrainingImageLabel/")

exists1 = os.path.isfile("TrainingImageLabel\psd.txt")

if exists1:

tf = open("TrainingImageLabel\psd.txt", "r")

key = tf.read()

else:

new\_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show='\*')

if new\_pas == None:

mess.\_show(title='No Password Entered', message='Password not set!! Please try again')

else:

tf = open("TrainingImageLabel\psd.txt", "w")

tf.write(new\_pas)

mess.\_show(title='Password Registered', message='New password was registered successfully!!')

return

password = tsd.askstring('Password', 'Enter Password', show='\*')

if (password == key):

TrainImages()

elif (password == None):

pass

else:

mess.\_show(title='Wrong Password', message='You have entered wrong password')

#

def clear():

txt.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

def clear2():

txt2.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

#

def TakeImages():

check\_haarcascadefile()

columns = ['SERIAL NO.', '', 'ID', '', 'NAME']

assure\_path\_exists("StudentDetails/")

assure\_path\_exists("TrainingImage/")

serial = 0

exists = os.path.isfile("StudentDetails\StudentDetails.csv")

if exists:

with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:

reader1 = csv.reader(csvFile1)

for l in reader1:

serial = serial + 1

serial = (serial // 2)

csvFile1.close()

else:

with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile1:

writer = csv.writer(csvFile1)

writer.writerow(columns)

serial = 1

csvFile1.close()

Id = (txt.get())

name = (txt2.get())

if ((name.isalpha()) or (' ' in name)):

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 + "." + str(serial) + "." + Id + '.' + str(sampleNum) + ".jpg",

gray[y:y + h, x:x + w])

# display the frame

cv2.imshow('Taking Images', img)

# wait for 100 miliseconds

if cv2.waitKey(100) & 0xFF == ord('q'):

break

# break if the sample number is morethan 100

elif sampleNum > 100:

break

cam.release()

cv2.destroyAllWindows()

res = "Images Taken for ID : " + Id

row = [serial, '', Id, '', name]

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

writer = csv.writer(csvFile)

writer.writerow(row)

csvFile.close()

message1.configure(text=res)

else:

if (name.isalpha() == False):

res = "Enter Correct name"

message.configure(text=res)

#

def TrainImages():

check\_haarcascadefile()

assure\_path\_exists("TrainingImageLabel/")

recognizer = cv2.face\_LBPHFaceRecognizer.create()

harcascadePath = "haarcascade\_frontalface\_default.xml"

detector = cv2.CascadeClassifier(harcascadePath)

faces, ID = getImagesAndLabels("TrainingImage")

try:

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

except:

mess.\_show(title='No Registrations', message='Please Register someone first!!!')

return

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

res = "Profile Saved Successfully"

message1.configure(text=res)

message.configure(text='Total Registrations till now : ' + str(ID[0]))

# 3

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

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 TrackImages():

check\_haarcascadefile()

assure\_path\_exists("Attendance/")

assure\_path\_exists("StudentDetails/")

for k in tv.get\_children():

tv.delete(k)

msg = ''

i = 0

j = 0

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

exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")

if exists3:

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

else:

mess.\_show(title='Data Missing', message='Please click on Save Profile to reset data!!')

return

harcascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(harcascadePath);

cam = cv2.VideoCapture(0)

font = cv2.FONT\_HERSHEY\_SIMPLEX

col\_names = ['Id', '', 'Name', '', 'Date', '', 'Time']

exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")

if exists1:

df = pd.read\_csv("StudentDetails\StudentDetails.csv")

else:

mess.\_show(title='Details Missing', message='Students details are missing, please check!')

cam.release()

cv2.destroyAllWindows()

window.destroy()

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)

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

if (conf < 50):

ts = time.time()

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

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

aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values

ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values

ID = str(ID)

ID = ID[1:-1]

bb = str(aa)

bb = bb[2:-2]

attendance = [str(ID), '', bb, '', str(date), '', str(timeStamp)]

else:

Id = 'Unknown'

bb = str(Id)

cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)

cv2.imshow('Taking Attendance', im)

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

break

ts = time.time()

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

exists = os.path.isfile("Attendance\Attendance\_" + date + ".csv")

if exists:

with open("Attendance\Attendance\_" + date + ".csv", 'a+') as csvFile1:

writer = csv.writer(csvFile1)

writer.writerow(attendance)

csvFile1.close()

else:

with open("Attendance\Attendance\_" + date + ".csv", 'a+') as csvFile1:

writer = csv.writer(csvFile1)

writer.writerow(col\_names)

writer.writerow(attendance)

csvFile1.close()

with open("Attendance\Attendance\_" + date + ".csv", 'r') as csvFile1:

reader1 = csv.reader(csvFile1)

for lines in reader1:

i = i + 1

if (i > 1):

if (i % 2 != 0):

iidd = str(lines[0]) + ' '

tv.insert('', 0, text=iidd, values=(str(lines[2]), str(lines[4]), str(lines[6])))

csvFile1.close()

cam.release()

cv2.destroyAllWindows()

# USED STUFFS #

global key

key = ''

ts = time.time()

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

day,month,year=date.split("-")

mont={'01':'January',

'02':'February',

'03':'March',

'04':'April',

'05':'May',

'06':'June',

'07':'July',

'08':'August',

'09':'September',

'10':'October',

'11':'November',

'12':'December'

}

# GUI FRONT-END #

window = tk.Tk()

window.geometry("1280x720")

window.resizable(True,False)

window.title("Attendance System")

window.configure(background='#2d420a')

frame1 = tk.Frame(window, bg="#c79cff")

frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#c79cff")

frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="Face Recognition Based Attendance Monitoring System" ,fg="white",bg="#2d420a" ,width=55 ,height=1,font=('comic', 29, ' bold '))

message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="#c4c6ce")

frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="#c4c6ce")

frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+" | ", fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 22, ' bold '))

datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 22, ' bold '))

clock.pack(fill='both',expand=1)

tick()

head2 = tk.Label(frame2, text=" For New Registrations ", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )

head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text=" For Already Registered ", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )

head1.place(x=0,y=0)

lbl = tk.Label(frame2, text="Enter ID",width=20 ,height=1 ,fg="black" ,bg="#c79cff" ,font=('comic', 17, ' bold ') )

lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))

txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter Name",width=20 ,fg="black" ,bg="#c79cff" ,font=('comic', 17, ' bold '))

lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold ') )

txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile" ,bg="#c79cff" ,fg="black" ,width=39 ,height=1, activebackground = "#3ffc00" ,font=('comic', 15, ' bold '))

message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#c79cff" ,fg="black" ,width=39,height=1, activebackground = "#3ffc00" ,font=('comic', 16, ' bold '))

message.place(x=7, y=450)

lbl3 = tk.Label(frame1, text="Attendance",width=20 ,fg="black" ,bg="#c79cff" ,height=1 ,font=('comic', 17, ' bold '))

lbl3.place(x=100, y=115)

res=0

exists = os.path.isfile("StudentDetails\StudentDetails.csv")

if exists:

with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:

reader1 = csv.reader(csvFile1)

for l in reader1:

res = res + 1

res = (res // 2) - 1

csvFile1.close()

else:

res = 0

message.configure(text='Total Registrations till now : '+str(res))

# MENUBAR #

menubar = tk.Menu(window,relief='ridge')

filemenu = tk.Menu(menubar,tearoff=0)

filemenu.add\_command(label='Change Password', command = change\_pass)

filemenu.add\_command(label='Contact Us', command = contact)

filemenu.add\_command(label='Exit',command = window.destroy)

menubar.add\_cascade(label='Help',font=('comic', 29, ' bold '),menu=filemenu)

# TREEVIEW ATTENDANCE TABLE #

tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))

tv.column('#0',width=82)

tv.column('name',width=130)

tv.column('date',width=133)

tv.column('time',width=133)

tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)

tv.heading('#0',text ='ID')

tv.heading('name',text ='NAME')

tv.heading('date',text ='DATE')

tv.heading('time',text ='TIME')

# SCROLLBAR #

scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)

scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')

tv.configure(yscrollcommand=scroll.set)

# BUTTONS #

clearButton = tk.Button(frame2, text="Clear", command=clear ,fg="black" ,bg="#ff7221" ,width=11 ,activebackground = "white" ,font=('comic', 11, ' bold '))

clearButton.place(x=335, y=86)

clearButton2 = tk.Button(frame2, text="Clear", command=clear2 ,fg="black" ,bg="#ff7221" ,width=11 , activebackground = "white" ,font=('comic', 11, ' bold '))

clearButton2.place(x=335, y=172)

takeImg = tk.Button(frame2, text="Take Images", command=TakeImages ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

takeImg.place(x=30, y=300)

trainImg = tk.Button(frame2, text="Save Profile", command=psw ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

trainImg.place(x=30, y=380)

trackImg = tk.Button(frame1, text="Take Attendance", command=TrackImages ,fg="black" ,bg="#3ffc00" ,width=35 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

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

quitWindow = tk.Button(frame1, text="Quit", command=window.destroy ,fg="black" ,bg="#eb4600" ,width=35 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

quitWindow.place(x=30, y=450)

# END #

window.configure(menu=menubar)

window.mainloop()