**Student Attendance System**

**Introduction:**

The world has become faster still we are doing a lot of things manually, such as student attendance so I thought to make it easier to save our precious time from spending on manual attendance.

It will allow the user to make attendance just by registering their face once.

**What is the Project about?**

The project is about registering attendance of students, teachers and office employee using face recognition technology.

**What tool was used?**

I used python as tool. As it was most convenient to apply the AI ideas so i chose this language.

**What is the platform of the project?**

I used anaconda as the platform of my project. Because it’s easy to for python language. And easy to work with opencv and **tkinter. As my project require them badly.**

Now I will be describing my project in details to let you know how I made it work.

First of all I imported all the necessary Python file or a Python module to access the script.

**import tkinter as tk** : **import Tkinter** imports the "namespace" **Tkinter** in the namespace and **import Tkinter as tk** does the same, but "renames" it locally to '**tk**' to save typing.

**import cv2,os:** OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Now, there in the latest releases, there is only the **cv2** module, and cv is a subclass inside **cv2** .

The OS module in python provides functions for interacting with the operating system. OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

**import csv**: The **csv** module gives the **Python**  the ability to parse **CSV** (Comma Separated Values) files. A **CSV** file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter.

**import numpy as np:** it creats a link that points to **numpy**, but **numpy** is not further linked to f2py. The link is established when we do **import numpy**.f2py. Calling the statement **import numpy as np**, shortening the phrase "**numpy**" to "**np**" to make the code easier to read

**from PIL import Image:** Python Imaging Library (abbreviated as **PIL**) is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different **image** file formats. It is available for Windows, Mac OS X and Linux.

**import pandas as pd**: **pandas** is a popular **Python**-based data analysis toolkit which can be **imported** using **import pandas as pd**. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array.

**import datetime**: A date in **Python** is not a data type of its own, but we **can import** a module named **datetime** to work with dates as date objects.

**import time**: **time**() The **time**() function returns the number of seconds passed since epoch. For UNIX system, January 1, 1970, 00:00:00 at UTC is epoch (the point where **time** begins). **Import time** seconds = **time**.**time**() print("Seconds since epoch =", seconds)

**from tkinter import messagebox:** The **tkinker** tkMessageBox has various methods to display a **message box**.There is a slight difference between [**Tkinter**](https://pythonspot.com/tkinter/) for Python 2.7 and Python 3. My python version is 3 that’s why I used ‘from tkinter import messagebox’ to display message.

**window = tk.Tk(): Tk** class is used to create a root window. Frame is a container for other widgets. Our example class inherits from the Frame container widget

**window.title("Attendance System"):** it is the title of the window. Where we kept the title name as “Attendance System”.

**window.configure(background='pink'):** to make the background pink color we used this.

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

**window.grid\_columnconfigure(0, weight=1):** Every column and row has a "weight" grid option associated with it, which tells it how much it should grow if there is extra room in the master to fill. By default, the weight of each column or row is 0, meaning don't expand to fill space.

**message = tk.Label(window, text="METROPOLITAN UNIVERSITY" ,bg="white" ,fg="black" ,width=20 ,height=2,font=('Times New Roman', 25, 'bold'))**

**message.place(x=1150, y=760)**

**message = tk.Label(window, text="ATTENDANCE MANAGEMENT PORTAL" ,bg="pink" ,fg="black" ,width=40 ,height=1,font=('Times New Roman', 35, 'bold underline'))**

**message.place(x=200, y=20)**

In this portion of code I placed two message using tk.label and fixed their position using message.place .

lbl = tk.Label(window, text="Enter Your ID",width=20 ,height=2 ,fg="black" ,bg="Pink" ,font=('Times New Roman', 25, ' bold ') )

lbl.place(x=200-x\_cord, y=200-y\_cord)

lbl2 = tk.Label(window, text="Enter Your Name",width=20 ,fg="black" ,bg="pink" ,height=2 ,font=('Times New Roman', 25, ' bold '))

lbl2.place(x=600-x\_cord, y=200-y\_cord)

txt2 = tk.Entry(window,width=30 ,bg="white" ,fg="blue",font=('Times New Roman', 15, ' bold ') )

txt2.place(x=650-x\_cord, y=300-y\_cord)

lbl3 = tk.Label(window, text="NOTIFICATION",width=20 ,fg="black" ,bg="pink" ,height=2 ,font=('Times New Roman', 25, ' bold '))

lbl3.place(x=1060-x\_cord, y=200-y\_cord)

message = tk.Label(window, text="" ,bg="white" ,fg="blue" ,width=30 ,height=1, activebackground = "white" ,font=('Times New Roman', 15, ' bold '))

message.place(x=1075-x\_cord, y=300-y\_cord)

lbl3 = tk.Label(window, text="ATTENDANCE",width=20 ,fg="white" ,bg="lightgreen" ,height=2 ,font=('Times New Roman', 30, ' bold '))

lbl3.place(x=120, y=570-y\_cord)

message2 = tk.Label(window, text="" ,fg="red" ,bg="yellow",activeforeground = "green",width=60 ,height=4 ,font=('times', 15, ' bold '))

message2.place(x=700, y=570-y\_cord)

lbl4 = tk.Label(window, text="STEP 1",width=20 ,fg="green" ,bg="pink" ,height=2 ,font=('Times New Roman', 20, ' bold '))

lbl4.place(x=240-x\_cord, y=375-y\_cord)

lbl5 = tk.Label(window, text="STEP 2",width=20 ,fg="green" ,bg="pink" ,height=2 ,font=('Times New Roman', 20, ' bold '))

lbl5.place(x=645-x\_cord, y=375-y\_cord)

lbl6 = tk.Label(window, text="STEP 3",width=20 ,fg="green" ,bg="pink" ,height=2 ,font=('Times New Roman', 20, ' bold '))

lbl6.place(x=1100-x\_cord, y=362-y\_cord)

In this portion I placed some text, message and label to let user enter data and label them. And had some button declaring secuence step1, step2,step3 to make the procedure more easier. Which I will link in the next portion of code.

Where,

First step “image capture” button will capture 60 best possible frames and save them for later works.

Second step model training button will train images with the given data sets of particular student.

Attendance marking button will mark attendance, and a pop up will show up where the camera would match the face in front of it with the given data sets. Pressing q will updated attendance. N background a csv file will automatically create.

def clear1():

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():

Id=(txt.get())

name=(txt2.get())

if not Id:

res="Please enter Id"

message.configure(text = res)

MsgBox = tk.messagebox.askquestion ("Warning","Please enter ID properly , press yes if you understood",icon = 'warning')

if MsgBox == 'no':

tk.messagebox.showinfo('Your need','Please go through the project brief file properly')

elif not name:

res="Please enter Name"

message.configure(text = res)

MsgBox = tk.messagebox.askquestion ("Warning","Please enter your name properly , press yes if you understood",icon = 'warning')

if MsgBox == 'no':

tk.messagebox.showinfo('Your need','Please go through the project brief file properly')

elif(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>60:

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)

def TrainImages():

recognizer = cv2.face\_LBPHFaceRecognizer.create()

faces,Id = getImagesAndLabels("TrainingImage")

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

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

res = "Image Trained"

clear1();

clear2();

message.configure(text= res)

tk.messagebox.showinfo('Completed','Your model has been trained successfully!!')

def getImagesAndLabels(path):

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

faces=[]

Ids=[]

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():

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']

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 < 50):

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['Id'] == Id]['Name'].values

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

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

else:

Id='Unknown'

tt=str(Id)

if(conf > 75):

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)

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

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

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

cam.release()

cv2.destroyAllWindows()

res=attendance

message2.configure(text= res)

res = "Attendance Taken"

message.configure(text= res)

tk.messagebox.showinfo('Completed','Congratulations ! Your attendance has been marked successfully for the day!!')

def quit\_window():

MsgBox = tk.messagebox.askquestion ('Exit Application','Are you sure you want to exit the application',icon = 'warning')

if MsgBox == 'yes':

tk.messagebox.showinfo("Greetings", "Thank You very much for using our software. Have a nice day ahead!!")

window.destroy()

takeImg = tk.Button(window, text="IMAGE CAPTURE BUTTON", command=TakeImages ,fg="white" ,bg="blue" ,width=25 ,height=2, activebackground = "pink" ,font=('Times New Roman', 15, ' bold '))

takeImg.place(x=245-x\_cord, y=425-y\_cord)

trainImg = tk.Button(window, text="MODEL TRAINING BUTTON", command=TrainImages ,fg="white" ,bg="blue" ,width=25 ,height=2, activebackground = "pink" ,font=('Times New Roman', 15, ' bold '))

trainImg.place(x=645-x\_cord, y=425-y\_cord)

trackImg = tk.Button(window, text="ATTENDANCE MARKING BUTTON", command=TrackImages ,fg="white" ,bg="red" ,width=30 ,height=3, activebackground = "pink" ,font=('Times New Roman', 15, ' bold '))

trackImg.place(x=1075-x\_cord, y=412-y\_cord)

In this final potion of my project I added some message box, label and linked the buttons of step1, step2, step3 with the buttons and added commands to them from OpenCv library. And formatted the csv file to save all the records.

To run this program successfully we need to just make a folder named TrainingImageLabel and

install opencv and tkinter in anaconda and then we can run attendance.py