TARP

DIGITAL ASSIGNMENT – 4

TEAM MEMBERS:

- ADRIJA MUKHOPADHYAY 19BDS0159
- MANDIRA HAWALDAR 19BCT0052
- TARANG GARG 19BCE0053
- VISHAKHA KUMARESAN 19BCE2678

IMPLEMENTATION

OF FACE DETCTION –

main

jupyter main Last Checkpoint: Yesterday at 8:56 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [ ]:

```
import os
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from person_info import Person
from train import Train
from facedetector import FaceDetector
import tkinter

class Face_recognition_System:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1530x790+0+0")
        self.root.title("face recognition system")

#first image
        img = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\college
        img=img.resize((500,130),Image.ANTIALIAS)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=500,height=130)


#second image
        img1 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img1=img1.resize((500,130),Image.ANTIALIAS)
        self.photoimg1=ImageTk.PhotoImage(img1)
        f_lbl=Label(self.root,image=self.photoimg1)
        f_lbl.place(x=500,y=0,width=500,height=130)

#third image
        img2 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img2=img2.resize((500,130),Image.ANTIALIAS)
        self.photoimg2=ImageTk.PhotoImage(img2)
        f_lbl=Label(self.root,image=self.photoimg2)
        f_lbl.place(x=1000,y=0,width=500,height=130)
```

```
#bg image

        img3 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img3=img3.resize((1530,710),Image.ANTIALIAS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        bg_img=Label(self.root,image=self.photoimg3)
        bg_img.place(x=0,y=130,width=1530,height=710)


        title_lbl=Label(bg_img,text="Face Recognition Sytem ", font=("times new roman",30,"bold"),bg="white",fg="red")
        title_lbl.place(x=0,y=0,width=1300,height=45)

#student button
        img4 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img4=img4.resize((150,150),Image.ANTIALIAS)
        self.photoimg4=ImageTk.PhotoImage(img4)

        b1=Button(bg_img,image=self.photoimg4,cursor="hand2",command=self.person_details)
        b1.place(x=100,y=50,width=150,height=150)

        b1_1=Button(bg_img,text="Person Details",cursor="hand2",command=self.person_details,font=("times new roman",15,"bold"
        b1_1.place(x=100,y=180,width=150,height=40)

 #Detect face button
        img5 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img5=img5.resize((150,150),Image.ANTIALIAS)
        self.photoimg5=ImageTk.PhotoImage(img5)

        b1=Button(bg_img,image=self.photoimg5,cursor="hand2",command=self.face_data)
        b1.place(x=500,y=50,width=150,height=150)

        b1_1=Button(bg_img,text="Face detector",cursor="hand2",command=self.face_data,font=("times new roman",15,"bold"),bg="
        b1_1.place(x=500,y=180,width=150,height=40)


#Attendance face button

        img6 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img6=img6.resize((150,150),Image.ANTIALIAS)
        self.photoimg6=ImageTk.PhotoImage(img6)

        b1=Button(bg_img,image=self.photoimg6,cursor="hand2",command=self.open_excel)
        b1.place(x=900,y=50,width=150,height=150)

        b1_1=Button(bg_img,text="Attendance",cursor="hand2",command=self.open_excel,font=("times new roman",15,"bold"),bg="da
        b1_1.place(x=900,y=180,width=150,height=40)
```

```
#Train face button
        img8 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img8=img8.resize((150,150),Image.ANTIALIAS)
        self.photoimg8=ImageTk.PhotoImage(img8)

        b1=Button(bg_img,image=self.photoimg8,cursor="hand2",command=self.train_data)
        b1.place(x=100,y=300,width=150,height=150)

        b1_1=Button(bg_img,text="Train face",cursor="hand2",command=self.train_data,font=("times new roman",15,"bold"),bg="da
        b1_1.place(x=100,y=450,width=150,height=40)

#Photos button
        img9 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img9=img9.resize((150,150),Image.ANTIALIAS)
        self.photoimg9=ImageTk.PhotoImage(img9)

        b1=Button(bg_img,image=self.photoimg9,cursor="hand2",command=self.open_img)
        b1.place(x=500,y=300,width=150,height=150)

        b1_1=Button(bg_img,text="Photos",cursor="hand2",command=self.open_img,font=("times new roman",15,"bold"),bg="darkblue
        b1_1.place(x=500,y=450,width=150,height=40)



#Exit button
        img11 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
        img11=img11.resize((150,150),Image.ANTIALIAS)
        self.photoimg11=ImageTk.PhotoImage(img11)

        b1=Button(bg_img,image=self.photoimg11,cursor="hand2",command=self.iexit)
        b1.place(x=900,y=300,width=150,height=150)

        b1_1=Button(bg_img,text="Exit",cursor="hand2",command=self.iexit,font=("times new roman",15,"bold"),bg="darkblue",fg=
        b1_1.place(x=900,y=450,width=150,height=40)
```

```
    def open_img(self):
        os.startfile("data")

    def open_excel(self):
        os.startfile("attendance.csv")

    def iexit(self):
        self.iexit=tkinter.messagebox.askyesno("Face Recognition","Are you sure you wana exit it?",parent=self.root)
        if self.iexit > 0:
            self.root.destroy()
        else:
            return

    #function buttons
    def person_details(self):
        self.new_window=Toplevel(self.root)
        self.app=Person(self.new_window)

    def train_data(self):
        self.new_window=Toplevel(self.root)
        self.app=Train(self.new_window)

    def face_data(self):
        self.new_window=Toplevel(self.root)
        self.app=FaceDetector(self.new_window)


if __name__=="__main__":
    root= Tk()
    obj=Face_recognition_System(root)


    root.mainloop()
```

personal_info

```
In [ ]:  from tkinter import*
         from tkinter import ttk
         from PIL import Image,ImageTk
         from tkinter import messagebox
         import mysql.connector
         import cv2
         import os

         class Person:
             def __init__(self,root):
                 self.root=root
                 self.root.geometry("1530x790+0+0")
                 self.root.title("face recognition system")

                 ##+++======variables
                 self.var_Department=StringVar()
                 self.var_Course=StringVar()
                 self.var_Year=StringVar()
                 self.var_Age=StringVar()
                 self.var_Person_ID=StringVar()
                 self.var_Name=StringVar()
                 self.var_Gender=StringVar()
                 self.var_Phone_no=StringVar()
                 self.var_Address=StringVar()


         #first image
                 img = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\college
                 img=img.resize((500,130),Image.ANTIALIAS)
                 self.photoimg=ImageTk.PhotoImage(img)
                 f_lbl=Label(self.root,image=self.photoimg)
                 f_lbl.place(x=0,y=0,width=500,height=130)


         #second image
                 img1 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
                 img1=img1.resize((500,130),Image.ANTIALIAS)
                 self.photoimg1=ImageTk.PhotoImage(img1)
                 f_lbl=Label(self.root,image=self.photoimg1)
                 f_lbl.place(x=500,y=0,width=500,height=130)

         #third image
                 img2 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
                 img2=img2.resize((500,130),Image.ANTIALIAS)
                 self.photoimg2=ImageTk.PhotoImage(img2)
                 f_lbl=Label(self.root,image=self.photoimg2)
                 f_lbl.place(x=1000,y=0,width=500,height=130)
```

```
#bg image

        img3 = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colleg
        img3=img3.resize((1530,710),Image.ANTIALIAS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        bg_img=Label(self.root,image=self.photoimg3)
        bg_img.place(x=0,y=130,width=1530,height=710)


        title_lbl=Label(bg_img,text="Missing Management System ", font=("times new roman",30,"bold"),bg="white",fg="red")
        title_lbl.place(x=0,y=0,width=1350,height=45)


        main_frame=Frame(bg_img,bd=2,bg="white")
        main_frame.place(x=11,y=55,width=1250,height=455)

#left label frame

        Left_frame= LabelFrame(main_frame,bd=2,bg="white",relief=RIDGE,text=" Missing Details",font=("times new roman",12,"bo
        Left_frame.place(x=21,y=10, width =650 , height=430)

        img_L = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
        img_L=img_L.resize((720,130),Image.ANTIALIAS)
        self.photoimg_L=ImageTk.PhotoImage(img_L)
        f_lbl=Label(Left_frame,image=self.photoimg_L)
        f_lbl.place(x=0,y=0,width=600,height=80)
#current course
        cc_frame= LabelFrame(Left_frame,bd=2,bg="white",relief=RIDGE,text="Missing Person Info",font=("times new roman",12,"b
        cc_frame.place(x=5,y=80, width =580 , height=120)

        #department label
        dep_label = Label(cc_frame, text="Department",font=("times new roman",12,"bold"),bg="white")
        dep_label.grid(row=0,column=0,padx=2,pady=10,sticky=W)


        dep_combo=ttk.Combobox(cc_frame,textvariable=self.var_Department,font=("times new roman",12,"bold"),width=17,state="r
        dep_combo["values"]=("Select Department","A","B","C")
        dep_combo.current(0)
        dep_combo.grid(row=0,column=1,padx=2,pady=10,sticky=W)
```

```
    #Course label
        c_label = Label(cc_frame, text="Course",font=("times new roman",12,"bold"),bg="white")
        c_label.grid(row=0,column=2,padx=2,pady=10,sticky=W)


        c_combo=ttk.Combobox(cc_frame,textvariable=self.var_Course,font=("times new roman",12,"bold"),width=17,state="readonl
        c_combo["values"]=("Select Course","A","B","C")
        c_combo.current(0)
        c_combo.grid(row=0,column=3,padx=2,pady=10,sticky=W)

    #Year
        y_label = Label(cc_frame, text="Year",font=("times new roman",12,"bold"),bg="white")
        y_label.grid(row=1,column=0,padx=2,pady=10,sticky=W)


        y_combo=ttk.Combobox(cc_frame,textvariable=self.var_Year,font=("times new roman",12,"bold"),width=17,state="readonly"
        y_combo["values"]=("Select Year","20-21","21-22","22-23")
        y_combo.current(0)
        y_combo.grid(row=1,column=1,padx=2,pady=10,sticky=W)

    #Age
        a_label = Label(cc_frame, text="Age",font=("times new roman",12,"bold"),bg="white")
        a_label.grid(row=1,column=2,padx=2,pady=10,sticky=W)


        a_combo=ttk.Combobox(cc_frame,textvariable=self.var_Age,font=("times new roman",12,"bold"),width=17,state="readonly")
        a_combo["values"]=("Select Year","10-20","20-30","30-40","40-50","50-60","60-70")
        a_combo.current(0)
        a_combo.grid(row=1,column=3,padx=2,pady=10,sticky=W)

#Personal Information
        p_frame= LabelFrame(Left_frame,bd=2,bg="white",relief=RIDGE,text="Missing Person Personal Info",font=("times new roma
        p_frame.place(x=5,y=200, width =580 , height=200)

        #Missing Person Id
        mi_label = Label(p_frame, text="Person Id",font=("times new roman",12,"bold"),bg="white")
        mi_label.grid(row=0,column=0,padx=1,pady=5,sticky=W)

        mi_entry= ttk.Entry(p_frame,textvariable= self.var_Person_ID,width=20,font=("times new roman",12,"bold"))
        mi_entry.grid(row=0,column=1,padx=1,pady=5,sticky=W)
```

```
 #Missing Person Name
mn_label = Label(p_frame, text="Name",font=("times new roman",12,"bold"),bg="white")
mn_label.grid(row=0,column=2,padx=1,pady=5,sticky=W)

mn_entry= ttk.Entry(p_frame,textvariable=self.var_Name,width=20,font=("times new roman",12,"bold"))
mn_entry.grid(row=0,column=3,padx=1,pady=5,sticky=W)


#Missing Person gender
g_label = Label(p_frame, text="Gender",font=("times new roman",12,"bold"),bg="white")
g_label.grid(row=1,column=0,padx=1,pady=5,sticky=W)


g_combo=ttk.Combobox(p_frame,textvariable=self.var_Gender,font=("times new roman",12,"bold"),width=17,state="readonly
g_combo["values"]=("Select Gender","Male","Female","Others")
g_combo.current(0)
g_combo.grid(row=1,column=1,padx=2,pady=10,sticky=W)

#Missing Person Phone no
mp_label = Label(p_frame, text="Phone No",font=("times new roman",12,"bold"),bg="white")
mp_label.grid(row=1,column=2,padx=1,pady=5,sticky=W)

mp_entry= ttk.Entry(p_frame,textvariable=self.var_Phone_no,width=20,font=("times new roman",12,"bold"))
mp_entry.grid(row=1,column=3,padx=1,pady=5,sticky=W)

#Missing Person Address
ma_label = Label(p_frame, text="Address",font=("times new roman",12,"bold"),bg="white")
ma_label.grid(row=2,column=0,padx=1,pady=5,sticky=W)

ma_entry= ttk.Entry(p_frame,textvariable=self.var_Address,width=20,font=("times new roman",12,"bold"))
ma_entry.grid(row=2,column=1,padx=1,pady=5,sticky=W)
```

```
    #radio Buttons
    self.var_radio1=StringVar()
    radio1=ttk.Radiobutton(p_frame,variable=self.var_radio1,text="take a photo Sample",value="Yes")
    radio1.grid(row=5,column=0)


    radio2=ttk.Radiobutton(p_frame,variable=self.var_radio1,text=" donot take a photo Sample",value="No")
    radio2.grid(row=5,column=1)


    #savebutton
    save_btn=Button(p_frame,text="Save",command=self.add_data,width=10,font=("times new roman",12,"bold"),bg="blue",fg="w
    save_btn.grid(row=6, column=0)


    #take a photo sample
    tps_btn=Button(p_frame,command=self.generate_dataset,text="take photo",width=10,font=("times new roman",12,"bold"),bg
    tps_btn.grid(row=6, column=1)
    #update a photo sample
    ups_btn=Button(p_frame,text="update photo",width=10,font=("times new roman",12,"bold"),bg="blue",fg="white")
    ups_btn.grid(row=6, column=2)
```

```
#Right label frame

        Right_frame= LabelFrame(main_frame,bd=2,bg="white",relief=RIDGE,text="Person Details",font=("times new roman",12,"bol
        Right_frame.place(x=630,y=10, width =600 , height=430)



        #image
        img_R = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
        img_R=img_R.resize((720,130),Image.ANTIALIAS)
        self.photoimg_R=ImageTk.PhotoImage(img_R)
        f_lbl=Label(Right_frame,image=self.photoimg_R)
        f_lbl.place(x=0,y=0,width=600,height=80)
```

```
    #image
    img_R = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
    img_R=img_R.resize((720,130),Image.ANTIALIAS)
    self.photoimg_R=ImageTk.PhotoImage(img_R)
    f_lbl=Label(Right_frame,image=self.photoimg_R)
    f_lbl.place(x=0,y=0,width=600,height=80)

    #Searching Sytem

    s_frame= LabelFrame(Right_frame,bd=2,bg="white",relief=RIDGE,text="Searching System",font=("times new roman",12,"bolc
    s_frame.place(x=5,y=80, width =585, height=80)

    s_label = Label(s_frame, text="Search BY:",font=("times new roman",12,"bold"),bg="red",fg="white")
    s_label.grid(row=0,column=0,padx=1,pady=5,sticky=W)

    s_combo=ttk.Combobox(s_frame,font=("times new roman",12,"bold"),width=15,state="readonly")
    s_combo["values"]=("Select","Person ID ","Name ")
    s_combo.current(0)
    s_combo.grid(row=0,column=1,padx=2,pady=10,sticky=W)

    s_entry= ttk.Entry(s_frame,width=15,font=("times new roman",12,"bold"))
    s_entry.grid(row=0,column=2,padx=1,pady=5,sticky=W)


    s_btn=Button(s_frame,text="Search",width=10,font=("times new roman",12,"bold"),bg="blue",fg="white")
    s_btn.grid(row=0, column=3 ,padx=4)

    show_btn=Button(s_frame,text="Show",width=10,font=("times new roman",12,"bold"),bg="blue",fg="white")
    show_btn.grid(row=0, column=4)


    #table
    t_frame= Frame(Right_frame,bd=2,bg="white",relief=RIDGE)
    t_frame.place(x=5,y=150, width =585, height=250)

    scroll_x=ttk.Scrollbar(t_frame,orient=HORIZONTAL)
    scroll_y=ttk.Scrollbar(t_frame,orient=VERTICAL)
    self.person_table=ttk.Treeview(t_frame,column=("Department","Course","Year","Age","Person ID","Name","Gender","Phone

    scroll_x.pack(side=BOTTOM,fill=X)
    scroll_y.pack(side=RIGHT,fill=Y)
    scroll_x.config(command=self.person_table.xview)
    scroll_y.config(command=self.person_table.yview)
```

```
    self.person_table.heading("Department",text="Department")
    self.person_table.heading("Course",text="Course")
    self.person_table.heading("Year",text="Year")
    self.person_table.heading("Age",text="Age")
    self.person_table.heading("Person ID",text="Person ID")
    self.person_table.heading("Name",text="Name")
    self.person_table.heading("Gender",text="Gender")
    self.person_table.heading("Phone No",text="Phone No")
    self.person_table.heading("Address",text="Address")
    self.person_table.heading("photo",text="PhotoSample Status")

    self.person_table["show"]="headings"


    self.person_table.column("Department",width=100)
    self.person_table.column("Course",width=100)
    self.person_table.column("Year",width=100)
    self.person_table.column("Age",width=100)
    self.person_table.column("Person ID",width=100)
    self.person_table.column("Name",width=100)
    self.person_table.column("Gender",width=100)
    self.person_table.column("Phone No",width=100)
    self.person_table.column("Address",width=100)
    self.person_table.column("photo",width=100)


    self.person_table.pack(fill=BOTH,expand=1)
    self.person_table.bind("<ButtonRelease>",self.get_cursor)
    self.fetch_data()

###function declaration

def add_data(self):
    if self.var_Department.get()=="Select Department" or self.var_Name.get()=="" or self.var_Person_ID.get()==="":
        messagebox.showerror("Error","All Fields are required",parent=self.root)
    else:
        try:
            conn=mysql.connector.connect(host="localhost",username="root",password="Dimpy#1609*",database="face_recognize
            my_cursor=conn.cursor()
            my_cursor.execute("insert into person values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(
                                                                                self.var_Department.get(),
                                                                                self.var_Course.get(),
                                                                                self.var_Year.get(),
                                                                                self.var_Age.get(),
                                                                                self.var_Person_ID.get()
```

```
                                                                                self.var_Name.get(),
                                                                                self.var_Gender.get(),
                                                                                self.var_Phone_no.get(),
                                                                                self.var_Address.get(),
                                                                                self.var_radio1.get()

                                                                               ))
                conn.commit()
                self.fetch_data()
                conn.close()
                messagebox.showinfo("Success","Person details has been added",parent=self.root)

            except Exception as es:
                messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


 #### fetch data
    def fetch_data(self):
        conn=mysql.connector.connect(host="localhost",username="root",password="Dimpy#1609*",database="face_recognizer")
        my_cursor=conn.cursor()
        my_cursor.execute("select * from person")
        data=my_cursor.fetchall()

        if len(data)!=0:
            self.person_table.delete(*self.person_table.get_children())
            for i in data:
                self.person_table.insert("", END,values=i)
            conn.commit()
        conn.close()
```

```
#####get_cursor
    def get_cursor(self,event=""):
        cursor_focus=self.person_table.focus()
        content=self.person_table.item(cursor_focus)
        data=content["values"]

        self.var_Department.set(data[0]),
        self.var_Course.set(data[1]),
        self.var_Year.set(data[2]),
        self.var_Age.set(data[3]),
        self.var_Person_ID.set(data[4]),
        self.var_Name.set(data[5]),
        self.var_Gender.set(data[6]),
        self.var_Phone_no.set(data[7]),
        self.var_Address.set(data[8]),
        self.var_radio1.set(data[9])



####Generate data set and take a photo sample #####
    def generate_dataset(self):
        if self.var_Department.get()=="Select Department" or self.var_Name.get()=="" or self.var_Person_ID.get()==="":
            messagebox.showerror("Error","All Fields are required",parent=self.root)
        else:
            try:
                conn=mysql.connector.connect(host="localhost",username="root",password="Dimpy#1609*",database="face_recognize
                my_cursor=conn.cursor()
                my_cursor.execute("Select * from person")
                myresult=my_cursor.fetchall()
                id=0
                for x in myresult:
                    id+=1
                my_cursor.execute("update person set Department=%s,Course=%s,Year=%s,Age=%s,Name=%s,Gender=%s,Phone_no=%s,Adc
```

```
        conn.commit()
        self.fetch_data()
        conn.close()

        ######Load predefined data
        face_classifier=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

        def face_cropped(img):
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            faces=face_classifier.detectMultiScale(gray,1.3,5)
            #scalinf factor=1.3
            #Minimum Neighbor=5

            for(x,y,w,h) in faces:
                face_cropped=img[y:y+h,x:x+w]
                return face_cropped

        cap=cv2.VideoCapture(0)
        img_id=0
        while True:
            ret,my_frame=cap.read()
            if face_cropped(my_frame)is not None:
```

```
                if face_cropped(my_frame)is not None:
                    img_id+=1
                    face=cv2.resize(face_cropped(my_frame),(450,450))
                    face=cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)
                    file_name_path="data/user."+str(id)+"."+str(img_id)+".jpg"
                    cv2.imwrite(file_name_path,face)
                    cv2.putText(face,str(img_id),(50,50),cv2.FONT_HERSHEY_COMPLEX,2,(0,255,0),2)
                    cv2.imshow("Crooped face",face)

                if cv2.waitKey(1)==13 or int(img_id)==100:
                    break
            cap.release()
            cv2.destroyAllWindows()
            messagebox.showinfo("Result","Generating data sets completed!!")
        except Exception as es:
            messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


if __name__=="__main__":
    root= Tk()
    obj=Person(root)
    root.mainloop()
```

train

```
In [1]:
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import mysql.connector
import cv2
import os
import numpy as np


class Train:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1530x790+0+0")
        self.root.title("face recognition system")

        title_lbl=Label(self.root,text="TRAIN DATA SET ", font=("times new roman",30,"bold"),bg="white",fg="red")
        title_lbl.place(x=0,y=0,width=1350,height=45)

        img_t = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
        img_t=img_t.resize((1330,265),Image.ANTIALIAS)
        self.photoimg_t=ImageTk.PhotoImage(img_t)

        f_lbl=Label(self.root,image=self.photoimg_t)
        f_lbl.place(x=0,y=43,width=1300,height=265)

        b1_1=Button(self.root,text="Train Data ",command=self.train_classifier,cursor="hand2",font=("times new roman",20,"bol
        b1_1.place(x=0,y=310,width=1300,height=40)

        img_b = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
        img_b=img_b.resize((1300,350),Image.ANTIALIAS)
        self.photoimg_b=ImageTk.PhotoImage(img_b)

        f_lbl=Label(self.root,image=self.photoimg_b)
        f_lbl.place(x=0,y=350,width=1300,height=350)
```

```
    def train_classifier(self):
        data_dir=("data")
        path=[os.path.join(data_dir,file) for file in os.listdir(data_dir)]


        faces=[]
        ids=[]
        for image in path:
            img=Image.open(image).convert('L') # gray scale image
            imageNp=np.array(img,'uint8')
            id=int(os.path.split(image)[1].split('.')[1])

            faces.append(imageNp)
            ids.append(id)
            cv2.imshow("Training",imageNp)
            cv2.waitKey(1)==13
        ids=np.array(ids)


        ### train the classifier

        clf=cv2.face.LBPHFaceRecognizer_create()
        clf.train(faces,ids)
        clf.write("classifier.xml")
        cv2.destroyAllWindows()
        messagebox.showinfo("Result","Training datasets completed!!")



if __name__=="__main__":
    root= Tk()
    obj=Train(root)
    root.mainloop()
```

Face_detector

```
In [2]:  from tkinter import*
         from tkinter import ttk
         from PIL import Image,ImageTk
         from tkinter import messagebox
         import mysql.connector
         from time import strftime
         from datetime import datetime
         import cv2
         import os
         import numpy as np


         class FaceDetector:
             def __init__(self,root):
                 self.root=root
                 self.root.geometry("1530x790+0+0")
                 self.root.title("face recognition system")

                 title_lbl=Label(self.root,text="FACE DETECTOR ", font=("times new roman",30,"bold"),bg="white",fg="red")
                 title_lbl.place(x=0,y=0,width=1350,height=45)

                 img_t = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
                 img_t=img_t.resize((700,610),Image.ANTIALIAS)
                 self.photoimg_t=ImageTk.PhotoImage(img_t)

                 f_lbl=Label(self.root,image=self.photoimg_t)
                 f_lbl.place(x=0,y=43,width=650,height=610)

                 img_b = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
                 img_b=img_b.resize((950,610),Image.ANTIALIAS)
                 self.photoimg_b=ImageTk.PhotoImage(img_b)

                 f_lbl=Label(self.root,image=self.photoimg_b)
                 f_lbl.place(x=500,y=43,width=950,height=610)

                 b1_1=Button(f_lbl,text="FACE DETECTOR ",command=self.face_recog,cursor="hand2",font=("times new roman",18,"bold"),bg=
                 b1_1.place(x=350,y=540,width=240,height=40)
```

```
    ###attendance####
 def mark_attendance(self,i,n,r,d):
     with open("attendance.csv","r+",newline="\n") as f:
         myDataList=f.readlines()
         name_list=[]
         for line in myDataList:
             entry=line.split((","))
             name_list.append(entry[0])
         if((i not in name_list) and (n not in name_list) and(r not in name_list) and(d not in name_list)):
             now=datetime.now()
             d1=now.strftime("%d/%m/%Y")
             dtString=now.strftime("%H:%M:%S")
             f.writelines(f"\n{i},{n},{r},{d},{dtString},{d1},Present")
```

```
    ###face detector####
def face_recog(self):
    def draw_boundray(img,classifier,scaleFactor,minNeighbors,color,text,clf):
        gray_image=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        features=classifier.detectMultiScale(gray_image,scaleFactor,minNeighbors)
        coord=[]
        for (x,y,w,h) in features:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
            id,predict=clf.predict(gray_image[y:y+h,x:x+w ])
            confidence=int((100*(1-predict/300)))

            conn=mysql.connector.connect(host="localhost",username="root",password="Dimpy#1609*",database="face_recognize
            my_cursor=conn.cursor()

            my_cursor.execute("select Name from person where Person_ID="+str(id))
            n=my_cursor.fetchone()
            n="+".join(n)

            my_cursor.execute("select Gender from person where Person_ID="+str(id))
            r=my_cursor.fetchone()
            r="+".join(r)

            my_cursor.execute("select Department from person where Person_ID="+str(id))
            d=my_cursor.fetchone()
            d="+".join(d)

            my_cursor.execute("select Person_ID from person where Person_ID="+str(id))
            i=my_cursor.fetchone()
            i="+".join(i)
```

```
                cv2.putText(img,f"Person_ID:{i}",(x,y-75),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                cv2.putText(img,f"Name:{n}",(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                cv2.putText(img,f"Gender:{r}",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                cv2.putText(img,f"Department:{d}",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                self.mark_attendance(i,n,r,d)
            else:
                cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
                cv2.putText(img,"Unknown Face",(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)

            coord=[x,y,w,y]

        return coord

    def recognize(img,clf,faceCascade):
        coord=draw_boundray(img,faceCascade,1.1,10,(255,25,255),"Face",clf)
        return img

    faceCascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    clf=cv2.face.LBPHFaceRecognizer_create()
    clf.read("classifier.xml")

    video_cap=cv2.VideoCapture(0)

    while True :
        ret,img=video_cap.read()
        img=recognize(img,clf,faceCascade)
        cv2.imshow("Welcome to face Recognition",img)
        if cv2.waitKey(1)==13:
            break
    video_cap.release()
    cv2.destroyAllWindows()



if __name__=="__main__":
    root= Tk()
    obj=FaceDetector(root)
    root.mainloop()
```

OF OBJECT DETECTION –

8/24/22, 8:31 PM Untitled0.ipynb - Colaboratory

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

```
YOLOv5 🚀 v6.2-51-ge6f54c5 Python-3.7.13 torch-1.12.1+cu113 CPU
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 37.3/107.7 GB disk)
```

```
!unzip -q ../train_data.zip -d ../
```

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 32 --epochs 140 --data coco128.yaml --weights yolov5s.p
```

```
train: weights=yolov5s.pt, cfg=, data=coco128.yaml, hyp=data/hyps/hyp.scratch-low
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v6.2-51-ge6f54c5 Python-3.7.13 torch-1.12.1+cu113 CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_
Weights & Biases: run 'pip install wandb' to automatically track and visualize YO
ClearML: run 'pip install clearml' to automatically track, visualize and remotely
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localho
Overriding model.yaml nc=80 with nc=1
```

```
                 from  n    params  module                                  argum
  0                -1  1      3520  models.common.Conv                      [3, 3
  1                -1  1     18560  models.common.Conv                      [32,
  2                -1  1     18816  models.common.C3                        [64,
  3                -1  1     73984  models.common.Conv                      [64,
  4                -1  2    115712  models.common.C3                        [128,
  5                -1  1    295424  models.common.Conv                      [128,
  6                -1  3    625152  models.common.C3                        [256,
  7                -1  1   1180672  models.common.Conv                      [256,
  8                -1  1   1182720  models.common.C3                        [512,
  9                -1  1    656896  models.common.SPPF                      [512,
 10                -1  1    131584  models.common.Conv                      [512,
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512,
 14                -1  1     33024  models.common.Conv                      [256,
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256,
 18                -1  1    147712  models.common.Conv                      [128,
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256,
 21                -1  1    590336  models.common.Conv                      [256,
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512,
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1, [
Model summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.9 GFLOPs

Transferred 343/349 items from yolov5s.pt
```

```
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 60 weight(dec
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3
train: Scanning '/content/yolov5/../train_data/labels/train.cache' images and lab
train: Caching images (0.0GB ram): 100% 15/15 [00:00<00:00, 316.69it/s]
val: Scanning '/content/yolov5/../train_data/labels/val.cache' images and labels.
val: Caching images (0.0GB ram): 100% 8/8 [00:00<00:00, 121.43it/s]

AutoAnchor: 2.22 anchors/target, 1.000 Best Possible Recall (BPR). Current anchor
Plotting labels to runs/train/exp8/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/train/exp8
Starting training for 140 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      0/139         0G     0.1164    0.03772          0         68        640: 10
                 Class     Images  Instances          P          R     mAP@.5 mAP(
```

EXPERIMENTAL COMPARISON

After comparing with the rest of the algorithms used in the papers we did a survey of, we can compare our model with the pre-existing ones.

	5 subjects			10 subjects (5 tested)			15 subjects (5 tested)		
	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40
Eigenfaces	60	60	60	40	40	40	20	20	20
Fisherfaces	70	50	50	20	50	30	20	10	10
Openface	30	40	40	0	10	20	10	30	30
LBPH	100	100	100	100	100	100	80	80	90
Random	20	20	20	10	10	10	6.7	6.7	6.7

Fig above: Face identification results for training and testing data in different environment 2 pictures were tested per person. Here only the data from 5 subjects, that were fixed, were used in testing phase (total 10 test images)

	5 subjects			10 subjects			15 subjects		
	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40
Eigenfaces	60	60	60	30	30	30	13	17	20
Fisherfaces	70	50	50	10	25	20	17	17	13
Openface	30	40	40	15	30	30	13	23	23
LBPH	100	100	100	100	100	100	93	93	97
Random	20	20	20	10	10	10	6.7	6.7	6.7

Fig above: Face identification results for training and testing data in different environment 2 pictures were tested per person

	5 subjects			10 subjects			15 subjects		
	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40	n = 10	n = 20	n = 40
Eigenfaces	92	100	100	96	100	100	96	99	100
Fisherfaces	92	100	100	96	98	100	93	95	100
Openface	96	100	100	98	100	100	99	99	100
LBPH	100	100	100	100	100	100	100	100	100
Random	20	20	20	10	10	10	6.7	6.7	6.7

Fig above: Face identification results for training and testing data in different environment 5 pictures were tested per person