DAFTAR PUSTAKA

Florestyanto, M. Y., Yuwono, B., & P., D. B. (2022). Dasar Pengolahan Citra Digital. In *Nucl. Phys.* (Vol. 13, Issue 1). Lembaga Penelitian dan Pengabdian Kepada Masyarakat UPN Veteran Yogyakarta.

Hasman, E., Ihsan, I. P., Pailing, H. S., & Safaruddin. (2021). Haar Cascade dan Algoritma Eignface Untuk Sistem Pembuka Pintu Otomatis. *JSAI (Journal Scientific and Applied Informatics)*, *4*(2), 182–192. https://doi.org/10.36085/jsai.v4i2.1642

Marleny, F. D. (2021). *Pengolahan Citra Digital Menggunakan Python*. CV. Pena Persada.

Padilha, T. P. P., & Lucena, L. E. A. de. (2020). A Systematic Review About Use of TensorFlow for Image Classification and Word Embedding in the Brazilian Context. *Academic Journal on Computing, Engineering and Applied Mathematics*, *1*(2), 24–27. https://doi.org/10.20873/uft.2675-3588.2020.v1n2.p24-27

Padmavathi, K., & Thangadurai, K. (2016). Implementation of RGB and grayscale images in plant leaves disease detection - Comparative study. *Indian Journal of Science and Technology*, *9*(6), 1–6. https://doi.org/10.17485/ijst/2016/v9i6/77739

Purwawijaya, E., Singarimbun, R. N., & Pasaribu, H. (2022). Implementasi Face Recognition Pada Absensi Karyawan Menggunakan Local Binary Pattern Histogram dan SHA 256 bit. *Jurnal Media Informatika Budidarma*, *6*(4), 2383. https://doi.org/10.30865/mib.v6i4.4923

Santoso, B., & Kristianto, R. P. (2020). Implementasi Penggunaan Opencv Pada Face Recognition Untuk Sistem Presensi Perkuliahan Mahasiswa. *Sistemasi*, *9*(2), 352. https://doi.org/10.32520/stmsi.v9i2.822

Sumarsono, I., & Harefa, K. (2023). Perancangan Sistem Aplikasi Absensi Menggunakan Face Recognition Dan Lokasi Berbasis Android Pada Pt. Trans Corp Food and Beverage. *LOGIC: Jurnal Ilmu Komputer …*, *1*(3), 395–405.

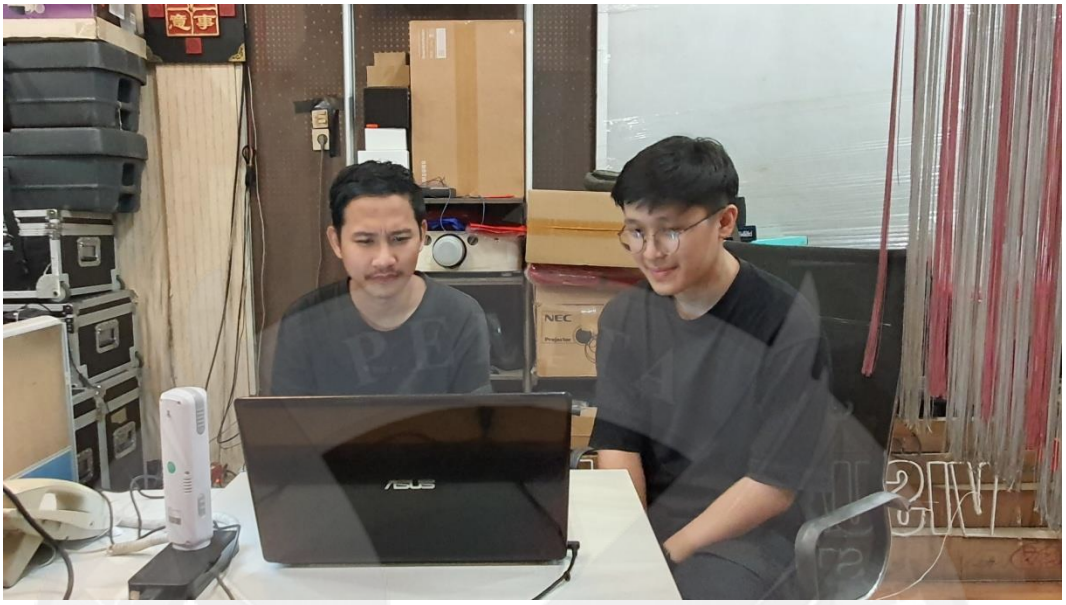
https://journal.mediapublikasi.id/index.php/logic/article/view/2648

Susilawati, H., Rukmana, A., & Nuraeni, F. (2023). Absensi Karyawan menggunakan Deteksi Wajah dan Gerakan Tangan Berbasis Raspberry Pi. *JATISI (Jurnal Teknik Informatika Dan Sistem …*, *10*(1). https://jurnal.mdp.ac.id/index.php/jatisi/article/view/3380%0Ahttps://jurnal.mdp.ac.id/index.php/jatisi/article/download/3380/1168

Tarigan, I. A., & Kurniawan, A. (2022). Prototipe Pendeteksi dan Pengenalan Wajah Berbasis Web Menggunakan Algoritma Local Binary Pattern Histogram untuk Absensi. *Jurnal MULTINETICS*, *8*(1), 77–86. Politeknik Negeri Jakarta

Wiranda, N., Purba, H. S., & Sukmawati, R. A. (2020). Survei Penggunaan Tensorflow pada Machine Learning untuk Identifikasi Ikan Kawasan Lahan Basah. *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, *10*(2), 179–188. https://doi.org/10.22146/ijeis.58315

LAMPIRAN A: BUKTI PENGUJIAN



LAMPIRAN B: SURAT IZIN PENELITIAN

Visual FX Studio
Jln. Mahayana No.30, Petisah Tengah, Kec. Medan Petisah,
Kota Medan, Sumatera Utara 20151
No. Telp : 0821-6034-3338

Kepada Yth,
Bapak/Ibu Pimpinan
Universitas Pelita Harapan
Medan

Dengan Hormat,
Dengan ini sebagai Pimpinan dari Visual FX Studio memberitahukan bahwa:

Nama	: Alessandro
NIM	: 03082180063
Prodi	: Teknik Informatika

Benar adanya telah mendapatkan ijin untuk melaksanakan penelitian di Visual FX Studio.
Demikianlah surat ini disampaikan, atas perhatiannya diucapkan terima kasih.

Medan, 02 November 2023
Pimpinan Visual FX Studio

Kosen
Pimpinan
082160343338

LAMPIRAN C: *LISTING* PROGRAM

developer.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import ttk
# from tkinter import messagebox
# import mysql.connector
# import cv2
# import os
# from turtle import left
#import numpy as np



class Developer:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1510x790+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')

    # #Tittle of MainFrame

title_lbl=Label(self.root,text="DEVELOPER" ,font=("times new
roman",25,"bold"),bg="white",fg="darkgreen")
        title_lbl.place(x=0,y=0,width=1410,height=50)

    #Top Left photo
        # img_top=Image.open(r"Project_Image\backand.jpg")
        #
img_top=img_top.resize((700,450),Image.Resampling.LANCZOS)
        # self.photoimg_top=ImageTk.PhotoImage(img_top)
        # f_lbl=Label(self.root,image=self.photoimg_top)
        # f_lbl.place(x=0,y=50,width=700,height=450)
    #Top Right photo
        img_top_left=Image.open(r"Project_Image\dftdev.jpg")

img_top_left=img_top_left.resize((1400,750),Image.Resampling
.LANCZOS)

self.photoimg_top_left=ImageTk.PhotoImage(img_top_left)
        f_lbl=Label(self.root,image=self.photoimg_top_left)
        f_lbl.place(x=0,y=50,width=1400,height=750)
    #Down photo
        # img_down=Image.open(r"Project_Image\dwnpic.jpg")
        #
img_down=img_down.resize((1400,310),Image.Resampling.LANCZOS
)
        # self.photoimg_down=ImageTk.PhotoImage(img_down)
```

```
        # f_lbl=Label(self.root,image=self.photoimg_down)
        # f_lbl.place(x=0,y=450,width=1400,height=310)


    #CDeveloper about labelFrame (170 positon of frame, 280
is height of frame)

developer_frame=LabelFrame(f_lbl,bd=3,relief=RIDGE,text="Abo
ut Developer",font=("times new roman",15,"bold"),bg="gold")

developer_frame.place(x=500,y=5,width=400,height=480)

    #My_image
        img_myImage=Image.open(r"Project_Image\myface.jpg")

img_myImage=img_myImage.resize((300,200),Image.Resampling.LA
NCZOS)

self.photoimg_myImage=ImageTk.PhotoImage(img_myImage)
    # Label for myImage

f_lbl=Label(developer_frame,image=self.photoimg_myImage)
        f_lbl.grid(row=0,column=0,padx=25,pady=10,sticky=W)

    #Developer About using Label
        Developer_name=Label(developer_frame,text="My name
is Alessandro",font=("times new
roman",15,"bold"),fg="darkblue")

Developer_name.grid(row=1,column=0,padx=2,pady=1,sticky=W)

Developer_enroll=Label(developer_frame,text="Enrollment No.
03082180063",font=("times new
roman",15,"bold"),fg="darkblue")

Developer_enroll.grid(row=2,column=0,padx=2,pady=1,sticky=W)
        dev_edu=Label(developer_frame,text="I am Student of
B.Tech in UPH Medan ",font=("times new
roman",15,"bold"),fg="darkblue")
        dev_edu.grid(row=3,column=0,padx=2,pady=1,sticky=W)
        dev_branch=Label(developer_frame,text="Email Id :
alessandro@gmail.com",font=("times new
roman",15,"bold"),fg="darkblue")

dev_branch.grid(row=5,column=0,padx=2,pady=1,sticky=W)

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

face_recognition.py

```
import os
from datetime import datetime
from time import strftime
from tkinter import *
from tkinter import messagebox, ttk
from unicodedata import name
import cv2
import mysql.connector
from PIL import Image, ImageTk
# import numpy as np


class Face_Recoginition:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1510x790+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')

        title_lbl=Label(self.root,text="FACE RECOGNTION
",font=("times new
roman",25,"bold"),bg="white",fg="darkgreen")
        title_lbl.place(x=0,y=0,width=1410,height=50)
        #1stLeft sede image
        img_left=Image.open(r"Project_Image\biometrics.jpg")

img_left=img_left.resize((600,710),Image.Resampling.LANCZOS)
        self.photoimg_left=ImageTk.PhotoImage(img_left)

        f_lbl=Label(self.root,image=self.photoimg_left)
        f_lbl.place(x=0,y=50,width=600,height=710)


        #2ndRight side Image
        img_right=Image.open(r"Project_Image\thumbnail.jpg")

img_right=img_right.resize((800,710),Image.Resampling.LANCZO
S)
        self.photoimg_right=ImageTk.PhotoImage(img_right)

        f_lbl=Label(self.root,image=self.photoimg_right)
        f_lbl.place(x=600,y=50,width=800,height=710)

        # Button
        img5=Image.open(r"Project_Image\camera.jpg")
```

```
        img5=img5.resize((230,180),Image.Resampling.LANCZOS)
        self.photoimg5=ImageTk.PhotoImage(img5)

b1=Button(f_lbl,image=self.photoimg5,cursor="hand2",command=
self.face_recog)
        b1.place(x=0,y=420,width=230,height=180)
        b1=Button(f_lbl,text="Face
Recognition",cursor="hand2",command=self.face_recog,font=("t
imes new roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=0,y=600,width=230,height=50)

#Attendace maeked CSV FILE
    # def mark_attendance(self,s,n,r,d):
    def mark_attendance(self,n,r,d):
        with open("attendance_data.csv","r+",newline="\n")
as f : #fie.readline>R+ile csv
            myDataList=f.readlines()
            name_list=[]
            for line in myDataList:
                entry=line.split(",")
                name_list.append(entry[0])
            # if((s not in name_list) and (n not in
name_list) and (r not in name_list) and (d  not in
name_list)):
            if((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{s},{n},{r},{d},{dtstring},{d1},Present")

f.writelines(f"{n},{r},{d},{dtstring},{d1},Present\n")




    #Face Recogniton
    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))
#formulaa confidence


conn=mysql.connector.connect(host="localhost",user="root",pa
ssword="",database="face_recognizer")
                my_cursor=conn.cursor()

                # my_cursor.execute("SELECT studentSNo FROM
student where studentSNo="+str(id))
                # s=my_cursor.fetchone()
                # s=str(s)
                # s="+".join(s)
                # s="+".join(eval(str(s)))

                my_cursor.execute("SELECT studentName FROM
student where studentSNo="+str(id))
                n=my_cursor.fetchone()
                if n is not None:
                    n = "+".join(n)
                else:
                    # Handle the case where no results were
found
                    n = "No Data Found"


                my_cursor.execute("SELECT enrollNo FROM
student where studentSNo="+str(id))
                r=my_cursor.fetchone()

                if r is not None:
                    r = "+".join(r)
                else:
                    # Handle the case where no results were
found
                    r = "No Data Found"

                my_cursor.execute("SELECT dep FROM student
where studentSNo="+str(id))
                d=my_cursor.fetchone()
                if d is not None:
                    d = "+".join(d)
                else:
                    # Handle the case where no results were
found
                    d = "No Data Found"
```

```
                if confidence>80:
                    # cv2.putText(img,f"S.No.: {s}",(x,y-
80),cv2.FONT_HERSHEY_COMPLEX,0.7,(255),2)
                    cv2.putText(img,f"Name: {n}",(x,y-
55),cv2.FONT_HERSHEY_COMPLEX,0.9,255,2)
                    cv2.putText(img,f"Enrollment No:
{r}",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.7,255,2)
                    cv2.putText(img,f"Position: {d}",(x,y-
5),cv2.FONT_HERSHEY_COMPLEX,0.7,255,2)
                    # self.mark_attendance(s,n,r,d)
                    self.mark_attendance(n,r,d)

                else:


cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
                    cv2.putText(img,"Unknown Employee
",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,25,255),3)

                    coord=[x,y,w,h]

                return coord
            #DO not reapet your self
        def recognize(img,clf,faceCascade):

coord=draw_boundray(img,faceCascade,1.1,10,255,"Face",clf)
            return img
        # Recognition and Detection
        faceCascade =
cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
        clf=cv2.face.LBPHFaceRecognizer_create()
        # clf=cv2.face.LBPHFaceRecognizer_create()
        clf.read("classifier.xml")

        video_cap=cv2.VideoCapture(0)
        video_cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
        video_cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)

        while True:
            ret, img = video_cap.read()
            img = recognize(img, clf, faceCascade)
            cv2.imshow("Welcome To Face Recognition", img)

            # Tambahkan kode untuk menutup kamera dan
jendela jika tombol "q" ditekan
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        video_cap.release()
        cv2.destroyAllWindows()
```

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

main.py

```
from tkinter import*
import tkinter
from PIL import Image,ImageTk
import os
from student import Student
from train import Train_Data
from face_recogniton import Face_Recoginition
from student_attendance import Student_Attendance
from developer import Developer
from time import strftime
import datetime as dt
# import tkinter as tk
# from tkinter import ttk
# from std_try import Student
# from pyclbr import Function
# from datetime import datetime
# from logging import root



class Face_Recognition_System:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1510x700+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')
        # root.resizable(1,1)

    #First Image Upload
        img=Image.open(r"Project_Image\sisrb.jpg")
        img=img.resize((500,150),Image.Resampling.LANCZOS)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=500,height=150)

    #Second1 Image Upload
        img1=Image.open(r"Project_Image\recogniation.jpg")
        img1=img1.resize((500,150),Image.Resampling.LANCZOS)
        self.photoimg1=ImageTk.PhotoImage(img1)
```

```
        f_lbl=Label(self.root,image=self.photoimg1)
        f_lbl.place(x=490,y=0,width=500,height=150)


     #Third Image Upload
        img2=Image.open(r"Project_Image\rgpv.jpg")
        img2=img2.resize((500,170),Image.Resampling.LANCZOS)
        self.photoimg2=ImageTk.PhotoImage(img2)
        f_lbl=Label(self.root,image=self.photoimg2)
        f_lbl.place(x=950,y=0,width=500,height=150)


     #background Image Upload
        img3=Image.open(r"Project_Image\bg.webp")

img3=img3.resize((1530,710),Image.Resampling.LANCZOS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        bg_img=Label(self.root,image=self.photoimg3)
        bg_img.place(x=0,y=170,width=1530,height=710)

    #Automatic Attendance Management System Using Face
Detecton
        title_lbl=Label(bg_img,text="Automatic Attendance
Management System Using Face Recognition ",font=("times new
roman",18,"bold"),bg="white",fg="red")
        title_lbl.place(x=0,y=0,width=1300,height=50)

        date = dt.datetime.now()
        label = Label(root, text=f"{date:%A}", font=('times
new roman',18,'bold'), bg='white',fg='darkblue')
        label.place(x=1130,y=172,width=150,height=50)
# Real world time %d for 24H %I for 12H
        def time():
                string = strftime('%I:%M:%S %p  %d/%m/%Y')
                lbl.config(text = string)
                lbl.after(1000, time)

        lbl = Label(title_lbl, font=('times new
roman',18,'bold'), bg='white',fg='darkblue')
        lbl.place(x=5,y=0,width=270,height=50)
        time()



#Student button1
        img4=Image.open(r"Project_Image\studet.jpg")
        img4=img4.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg4=ImageTk.PhotoImage(img4)
```

```
b1=Button(bg_img,image=self.photoimg4,command=self.student_d
etails,cursor="hand2")
        b1.place(x=180,y=120,width=180,height=120)
        b1=Button(bg_img,text="Employee
Details",command=self.student_details,cursor="hand2",font=("
times new roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=180,y=240,width=180,height=40)

#Face Recognition button2
        img5=Image.open(r"Project_Image\face.jpeg")
        img5=img5.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg5=ImageTk.PhotoImage(img5)

b1=Button(bg_img,image=self.photoimg5,cursor="hand2",command
= self.Face_Data)
        b1.place(x=480,y=120,width=180,height=120)
        b1=Button(bg_img,text="Face
Recognition",cursor="hand2",command =
self.Face_Data,font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=480,y=240,width=180,height=40)

    #Attendance button3
        img6=Image.open(r"Project_Image\attendance.jpeg")
        img6=img6.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg6=ImageTk.PhotoImage(img6)

b1=Button(bg_img,image=self.photoimg6,cursor="hand2",command
=self.Attendace)
        b1.place(x=780,y=120,width=180,height=120)

b1=Button(bg_img,text="Attendance",cursor="hand2",command=se
lf.Attendace,font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=780,y=240,width=180,height=40)

    #Help Desk button4
        img7=Image.open(r"Project_Image\developer.jpeg")
        img7=img7.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg7=ImageTk.PhotoImage(img7)

b1=Button(bg_img,image=self.photoimg7,cursor="hand2",command
=self.Developer)
        b1.place(x=1080,y=120,width=180,height=120)

b1=Button(bg_img,text="Developer",cursor="hand2",command=sel
f.Developer,font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=1080,y=240,width=180,height=40)
```

```
    #Train Data/Face button5
        img8=Image.open(r"Project_Image\train.jpeg")
        img8=img8.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg8=ImageTk.PhotoImage(img8)

b1=Button(bg_img,image=self.photoimg8,cursor="hand2",command
=self.Train_Student_Data)
        b1.place(x=180,y=320,width=180,height=120)
        b1=Button(bg_img,text="Train
Data",cursor="hand2",command=self.Train_Student_Data,font=("
times new roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=180,y=440,width=180,height=40)

#Photos button6
        img9=Image.open(r"Project_Image\photo.jpeg")
        img9=img9.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg9=ImageTk.PhotoImage(img9)

b1=Button(bg_img,image=self.photoimg9,cursor="hand2",command
=self.open_img)
        b1.place(x=480,y=320,width=180,height=120)

b1=Button(bg_img,text="Photos",cursor="hand2",command=self.o
pen_img,font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=480,y=440,width=180,height=40)

#Developer button7
        img11=Image.open(r"Project_Image\exit.jpg")

img11=img11.resize((180,120),Image.Resampling.LANCZOS)
        self.photoimg11=ImageTk.PhotoImage(img11)

b1=Button(bg_img,image=self.photoimg11,cursor="hand2",comman
d=self.Win_Exit)
        b1.place(x=780,y=320,width=180,height=120)

b1=Button(bg_img,text="Exit",cursor="hand2",font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        b1.place(x=780,y=440,width=180,height=40)

# Exit button8
        #img11=Image.open(r"Project_Image\exit.jpg")

#img11=img11.resize((180,120),Image.Resampling.LANCZOS)
        #self.photoimg11=ImageTk.PhotoImage(img11)

#b1=Button(bg_img,image=self.photoimg11,cursor="hand2",comma
nd=self.Win_Exit)
```

```
        #b1.place(x=1080,y=320,width=180,height=120)

#b1=Button(bg_img,text="Exit",cursor="hand2",command=self.Wi
n_Exit,font=("times new
roman",15,"bold"),bg="darkblue",fg="white")
        #b1.place(x=1080,y=440,width=180,height=40)



    def open_img(sslef):
        os.startfile("Data")


        # FunctionBUTTONS for student imagge and tittle
    def student_details(self):
        self.new_window=Toplevel(self.root)
        self.rupesh=Student(self.new_window)

    def Train_Student_Data(self):
        self.new_window=Toplevel(self.root)
        self.rupesh=Train_Data(self.new_window)

    def Face_Data(self):
        self.new_window=Toplevel(self.root)
        self.rupesh=Face_Recoginition(self.new_window)

    def Attendace(self):
        self.new_window=Toplevel(self.root)
        self.rupesh=Student_Attendance(self.new_window)

    def Developer(self):
        self.new_window=Toplevel(self.root)
        self.rupesh=Developer(self.new_window)


    def Win_Exit(self):
        self.Win_Exit=tkinter.messagebox.askyesno("Face
Recognition", "Do you want to exit window
page",parent=self.root)
        if self.Win_Exit > 0:
                self.root.destroy()
        else:
                return

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

employee.py

```
from tkinter import ttk
import os
import shutil
from tkinter import*
from PIL import Image,ImageTk
from tkinter import messagebox
from cv2 import FONT_HERSHEY_COMPLEX, CascadeClassifier,
VideoCapture, destroyAllWindows, imshow, imwrite, waitKey
import mysql.connector
import cv2
from tkinter import filedialog
# from turtle import update
# from importlib.resources import patht
# from ast import Pass
#from numpy import delete
# from platform import release
# from sqlite3 import Cursor



class Student:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1530x790+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')
        self.student_table = ttk.Treeview(root)
# Variables
        self.var_dep=StringVar()
        self.var_course=StringVar()
        self.var_year=StringVar()
        self.var_semester=StringVar()
        self.var_studentSNo=StringVar()
        self.var_studentName=StringVar()
        self.var_enrollNo=StringVar()
        self.var_classSection=StringVar()
        self.var_gender=StringVar()
        self.var_studentDOB=StringVar()
        self.var_emailId=StringVar()
        self.var_address=StringVar()
        self.var_phoneNo=StringVar()
        self.var_tgName=StringVar()

    #First Image Upload
        img=Image.open(r"Project_Image\frms.jpg")
        img=img.resize((500,190),Image.Resampling.LANCZOS)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=500,height=150)
```

```
    #Second1 Image Upload
        img1=Image.open(r"Project_Image\recogniation.jpg")
        img1=img1.resize((500,150),Image.Resampling.LANCZOS)
        self.photoimg1=ImageTk.PhotoImage(img1)
        f_lbl=Label(self.root,image=self.photoimg1)
        f_lbl.place(x=490,y=0,width=500,height=150)


    #Third Image Upload
        img2=Image.open(r"Project_Image\images.jpg")
        img2=img2.resize((500,170),Image.Resampling.LANCZOS)
        self.photoimg2=ImageTk.PhotoImage(img2)
        f_lbl=Label(self.root,image=self.photoimg2)
        f_lbl.place(x=950,y=0,width=500,height=150)

    #background Image Upload
        img3=Image.open(r"Project_Image\unlock.webp")

img3=img3.resize((1530,710),Image.Resampling.LANCZOS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        bg_img=Label(self.root,image=self.photoimg3)
        bg_img.place(x=0,y=120,width=1530,height=710)

        title_lbl=Label(bg_img,text="EMPLOYEE MANAGEMENT
SYSTEM",font=("times new
roman",25,"bold"),bg="white",fg="darkgreen")
        title_lbl.place(x=0,y=0,width=1430,height=35)

    #main frame in student
        main_frame=Frame(bg_img,bd=2)
        main_frame.place(x=20,y=45,width=1320,height=520)
        #left label frame

Left_lframe=LabelFrame(main_frame,bd=2,relief=RIDGE,text="Em
ployee Details",font=("times new roman",18,"bold"))
        Left_lframe.place(x=10,y=10,width=650,height=550)


    #image add in leftside
        img_left=Image.open(r"Project_Image\hand.jpg")

img_left=img_left.resize((640,100),Image.Resampling.LANCZOS)
        self.photoimg_left=ImageTk.PhotoImage(img_left)
        f_lbl=Label(Left_lframe,image=self.photoimg_left)
        f_lbl.place(x=5,y=0,width=650,height=80)

    #current course information

current_course_frame=LabelFrame(Left_lframe ,bd=2,relief=RID
```

```
GE,text="Current Division Information",font=("times new
roman",15,"bold"))

current_course_frame.place(x=10,y=80 ,width=620,height=100)
    #Department

dep_label=Label(current_course_frame,text="Position",font=("
times new roman",15,"bold"),bg="white",fg="darkgreen")
        dep_label.grid(row=0,column=0,padx=2,sticky=W)

dep_combo=ttk.Combobox(current_course_frame,textvariable=sel
f.var_dep,font=("times new
roman",15,"bold"),state="readonly",width=17)
        dep_combo["values"]=("Select Position","Chief
Financial Officer","Chief Information Officer","Chief
Marketing Officer","Chief Operation Officer","Human
Resources Development","Marketing Manager","Product
Manager")
        dep_combo.current(0)
        dep_combo.grid(row=0,column=1,padx=10,pady=2)


    #Course

course_label=Label(current_course_frame,text="Religion",font
=("times new roman",15,"bold"),bg="white",fg="darkgreen")
        course_label.grid(row=0,column=2,padx=2,sticky=W)

course_combo=ttk.Combobox(current_course_frame,textvariable=
self.var_course,font=("times new
roman",15,"bold"),state="readonly",width=15)
        course_combo["values"]=("Select
Religion","Muslim","Protestant","Catholic","Hindu","Budhist"
,"Confucian")
        course_combo.current(0)
        course_combo.grid(row=0,column=3,padx=10,
pady=2,sticky=W)


    #Year
        year_label=Label(current_course_frame,text="Year
Joined",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")
        year_label.grid(row=1,column=0,padx=2,sticky=W)

year_combo=ttk.Combobox(current_course_frame,textvariable=se
lf.var_year,font=("times new
roman",15,"bold"),state="readonly",width=17)
        year_combo["values"]=("Select
Year","2000","2001","2002","2003","2004","2005","2006","2007
```

```
","2008","2009","2010","2011","2012","2013","2014","2015","2
016","2017","2018","2019","2020","2021","2022","2023")
        year_combo.current(0)
        year_combo.grid(row=1,column=1,padx=10,pady=2)


    #Semester

semester_label=Label(current_course_frame,text="Status",font
=("times new roman",15,"bold"),bg="white",fg="darkgreen")
        semester_label.grid(row=1,column=2,padx=2,sticky=W)

semester_combo=ttk.Combobox(current_course_frame,textvariabl
e=self.var_semester,font=("times new
roman",15,"bold"),state="readonly",width=15)
        semester_combo["values"]=("Select
Status","Single","Married","Divorced","Widower","Widow")
        semester_combo.current(0)
        semester_combo.grid(row=1,column=3,padx=10,
pady=2,sticky=W)


    #Class student information (170 positon of frame, 280 is
height of frame)

class_student_frame=LabelFrame(Left_lframe ,bd=2,relief=RIDG
E,text="Employee information",font=("times new
roman",15,"bold"))

class_student_frame.place(x=10,y=175 ,width=620,height=300)

    #student serial no

studentSNo_label=Label(class_student_frame,text="Employee
No. :",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

studentSNo_label.grid(row=0,column=0,padx=2,pady=5,sticky=W)
    #student s.no. entry

studentSNo_entry=ttk.Entry(class_student_frame,textvariable=
self.var_studentSNo,font=("times new
roman",15,"bold"),width=15)

studentSNo_entry.grid(row=0,column=1,padx=2,pady=5,sticky=W)


    #student Name

studentName_label=Label(class_student_frame,text="Name :",fo
nt=("times new roman",15,"bold"),bg="white",fg="darkgreen")
```

```
studentName_label.grid(row=0,column=2,padx=2,pady=5,sticky=W
)
    #student name entry

studentName_entry=ttk.Entry(class_student_frame,textvariable
=self.var_studentName,font=("times new
roman",15,"bold"),width=15)

studentName_entry.grid(row=0,column=3,padx=2,pady=5,sticky=W
)


    #Class section

classSection_label=Label(class_student_frame,text="Education
:",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

classSection_label.grid(row=1,column=0,padx=2,sticky=W)

    #student Class section entry


classSection_entry=ttk.Combobox(class_student_frame,textvari
able=self.var_classSection,font=("times new
roman",15,"bold"),state="readonly",width=13)
        classSection_entry["values"]=("Select
Tier","SD","SMP","SMA","S1","S2","S3")
        classSection_entry.current(0)

classSection_entry.grid(row=1,column=1,padx=2,pady=5,sticky=
W)



    #student Enrollment numbber

enrollNo_label=Label(class_student_frame,text="Enrollment
No. :",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

enrollNo_label.grid(row=1,column=2,padx=2,pady=5,sticky=W)
    #student Enrollment numbber entry

enrollNo_entry=ttk.Entry(class_student_frame,textvariable=se
lf.var_enrollNo,font=("times new roman",15,"bold"),width=15)

enrollNo_entry.grid(row=1,column=3,padx=2,pady=5,sticky=W)
```

```
    #Student Gender

gender_label=Label(class_student_frame,text="Gender :",font=
("times new roman",15,"bold"),bg="white",fg="darkgreen")
        gender_label.grid(row=2,column=0,padx=2,sticky=W)

gender_combo=ttk.Combobox(class_student_frame,textvariable=s
elf.var_gender,font=("times new
roman",15,"bold"),state="readonly",width=13)
        gender_combo["values"]=("Select
Gender","Male","Female","Other")
        gender_combo.current(0)

gender_combo.grid(row=2,column=1,padx=2,pady=5,sticky=W)
    #
gender_label=Label(class_student_frame,text="Gender :",font=
("times new roman",15,"bold"),bg="white",fg="darkgreen")
    #
gender_label.grid(row=2,column=0,padx=2,pady=5,sticky=W)
    # #student Gender entry
    #
gender_entry=ttk.Entry(class_student_frame,textvariable=self
.var_gender,font=("times new roman",15,"bold"),width=15)
    #
gender_entry.grid(row=2,column=1,padx=2,pady=5,sticky=W)


    #student Date Of Birth

studentDOB_label=Label(class_student_frame,text="Date Of
Birth :",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

studentDOB_label.grid(row=2,column=2,padx=2,pady=5,sticky=W)
    #student Date Of Birth entry

studentDOB_entry=ttk.Entry(class_student_frame,textvariable=
self.var_studentDOB,font=("times new
roman",15,"bold"),width=15)

studentDOB_entry.grid(row=2,column=3,padx=2,pady=5,sticky=W)


    #Student Email i
        emailId_label=Label(class_student_frame,text="Email
id :",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

emailId_label.grid(row=3,column=0,padx=2,pady=5,sticky=W)
    #student Email id entry
```

```
emailId_entry=ttk.Entry(class_student_frame,textvariable=sel
f.var_emailId,font=("times new roman",15,"bold"),width=15)

emailId_entry.grid(row=3,column=1,padx=2,pady=5,sticky=W)


    #student Phone number
        phoneNo_label=Label(class_student_frame,text="Phone
No. :",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

phoneNo_label.grid(row=3,column=2,padx=2,pady=5,sticky=W)
    #student phone number entry

phoneNo_entry=ttk.Entry(class_student_frame,textvariable=sel
f.var_phoneNo,font=("times new roman",15,"bold"),width=15)

phoneNo_entry.grid(row=3,column=3,padx=2,pady=5,sticky=W)


     #Student Address

address_label=Label(class_student_frame,text="Address :",fon
t=("times new roman",15,"bold"),bg="white",fg="darkgreen")

address_label.grid(row=4,column=0,padx=2,pady=5,sticky=W)
    #student Address entry

address_entry=ttk.Entry(class_student_frame,textvariable=sel
f.var_address,font=("times new roman",15,"bold"),width=15)

address_entry.grid(row=4,column=1,padx=2,pady=5,sticky=W)


    #student T.G. name
        tgName_label=Label(class_student_frame,text="T.G.
Name ",font=("times new
roman",15,"bold"),bg="white",fg="darkgreen")

tgName_label.grid(row=4,column=2,padx=2,pady=5,sticky=W)
    #student T.G. name entry

tgName_entry=ttk.Entry(class_student_frame,textvariable=self
.var_tgName,font=("times new roman",15,"bold"),width=15)

tgName_entry.grid(row=4,column=3,padx=2,pady=5,sticky=W)


    #Button frame
```

```
btn_frame=Frame(class_student_frame,bd=2,relief=RIDGE,bg="wh
ite")
        btn_frame.place(x=0,y=220,width=650,height=70)


save_btn=Button(btn_frame,text="SAVE",command=self.add_data,
width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        save_btn.grid(row=0,column=0)


upate_btn=Button(btn_frame,text="UPDATE",command=self.update
_data,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        upate_btn.grid(row=0,column=1)


delete_btn=Button(btn_frame,text="DELETE",command=self.delet
e_data,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        delete_btn.grid(row=0,column=2)


reset_btn=Button(btn_frame,text="RESET",command=self.reset_d
ata,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        reset_btn.grid(row=0,column=3)


btn_frame1=Frame(class_student_frame,bd=2,relief=RIDGE,bg="w
hite")
        btn_frame1.place(x=0,y=250,width=650,height=35)


take_photo_btn=Button(btn_frame1,command=self.generate_datas
et,text="Take Photo Sample",width=43,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        take_photo_btn.grid(row=1,column=0)

        update_photo_btn=Button(btn_frame1,text="Update
Photo
Sample",command=self.Upload_photo,width=43,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        update_photo_btn.grid(row=1,column=1)

    #Right label frame

Right_frame=LabelFrame(main_frame,bd=2,relief=RIDGE,text="Em
ployee Data",font=("times new roman",15,"bold"))
        Right_frame.place(x=640,y=10,width=600,height=500)
```

```
        img_right=Image.open(r"Project_Image\unlock.webp")

img_right=img_right.resize((600,100),Image.Resampling.LANCZO
S)
        self.photoimg_right=ImageTk.PhotoImage(img_right)
        f_lbl=Label(Right_frame,image=self.photoimg_right)
        f_lbl.place(x=5,y=0,width=600,height=100)

    #====================SEARCH SYSTEM=============

search_frame=LabelFrame(Right_frame,bd=2,bg="white",relief=R
IDGE,text="Search System",font=("times new
roman",15,"bold"))
        search_frame.place(x=3,y=100,width=580,height=70)
    #=============search by label============
        search_label=Label(search_frame,text="Search By:
",font=("times new roman",12,"bold"),bg="blue",fg="white")

search_label.grid(row=0,column=0,padx=5,pady=5,sticky=W)
    #============= combobox for select============
        search_combo=ttk.Combobox(search_frame,font=("times
new roman",12,"bold"),state="readonly",width=13)
        search_combo["values"]=("Select","Roll_No.
","Phone_No. ")
        search_combo.current(0)

search_combo.grid(row=0,column=1,padx=2,pady=5,sticky=W)
    #==============enter the data==============

search_entry=ttk.Entry(search_frame,width=13,font=("times
new roman",12,"bold"))

search_entry.grid(row=0,column=2,padx=5,pady=5,sticky=W)
    #==============Button for search ==============

search_btn=Button(search_frame,text="Search",width=10,font=(
"times new roman",12,"bold"),bg="blue",fg="white")
        search_btn.grid(row=0,column=3,padx=5)
    #==============Button for showall ==============
        showAll_btn=Button(search_frame,text="Show
All",width=10,font=("times new
roman",12,"bold"),bg="blue",fg="white")
        showAll_btn.grid(row=0,column=4,padx=5)

    #====================TABLE FRAME=============

table_frame=LabelFrame(Right_frame,bd=2,bg="white",relief=RI
DGE)
        table_frame.place(x=5,y=180,width=580,height=300)
```

```
    #==============ttk.scrollbar module for scolling HOLI
VERT ===============

scroll_x=ttk.Scrollbar(table_frame,orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar(table_frame,orient=VERTICAL)


self.student_table=ttk.Treeview(table_frame,column=("dep","c
ourse","year","semester","studentSNo","studentName","classSe
ction","enrollNo","gender","studentDOB","emailId","phoneNo",
"address","tgName","photo"),xscrollcommand=scroll_x.set,yscr
ollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)

        scroll_x.config(command=self.student_table.xview)
        scroll_y.config(command=self.student_table.yview)

        self.student_table.heading("dep",text="Position")
        self.student_table.heading("course",text="Religion")
        self.student_table.heading("year",text="Year
Joined")
        self.student_table.heading("semester",text="Status")

self.student_table.heading("studentSNo",text="Employee No.")

self.student_table.heading("studentName",text="Name")

self.student_table.heading("classSection",text="Education")

self.student_table.heading("enrollNo",text="Enrollment No.")
        self.student_table.heading("gender",text="Gender")
        self.student_table.heading("studentDOB",text="Date
Of Birth")
        self.student_table.heading("emailId",text="Email
Id")
        self.student_table.heading("phoneNo",text="Phone
No.")
        self.student_table.heading("address",text="Address")
        self.student_table.heading("tgName",text="T.G.
Name")
        self.student_table["show"]="headings"

        self.student_table.column("dep",width=100)
        self.student_table.column("course",width=100)
        self.student_table.column("year",width=100)
        self.student_table.column("semester",width=100)
        self.student_table.column("studentSNo",width=100)
        self.student_table.column("studentName",width=100)
        self.student_table.column("classSection",width=100)
        self.student_table.column("enrollNo",width=100)
```

```
        self.student_table.column("gender",width=100)
        self.student_table.column("studentDOB",width=100)
        self.student_table.column("emailId",width=100)
        self.student_table.column("phoneNo",width=100)
        self.student_table.column("address",width=100)
        self.student_table.column("tgName",width=100)

        self.student_table.pack(fill=BOTH,expand=1)

self.student_table.bind("<ButtonRelease>",self.get_cursor)
        self.fetch_data()

    #make function for save/data adding
    def add_data(self):
        if self.var_dep.get()=="Select Position" or
self.var_studentName.get()=="" or
self.var_studentSNo.get()==""  or
self.var_course.get()=="Select Religion" or
self.var_year.get()=="Select Year"  or
self.var_semester.get()=="Select Status" or
self.var_classSection.get() == "Select Tier" or
self.var_gender.get() == "Select Gender" or
self.var_enrollNo.get()=="" or self.var_phoneNo.get()=="":
        # if self.var_dep.get()=="Select Department" or
self.var_studentName.get()=="" or
self.var_studentSNo.get()=="":
            messagebox.showerror("Error!","All Field are
required",parent=self.root)
        else:
            try:
            #   messagebox.showinfo("Success","WelcomeTo
SN3R")

conn=mysql.connector.connect(host="localhost",user="root",pa
ssword="",database="face_recognizer")
                my_cursor=conn.cursor()
                my_cursor.execute("insert into student
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(


self.var_dep.get(),

self.var_course.get(),

self.var_year.get(),

self.var_semester.get(),

self.var_studentSNo.get(),

self.var_studentName.get(),
```

```
self.var_classSection.get(),

self.var_enrollNo.get(),

self.var_gender.get(),

self.var_studentDOB.get(),

self.var_emailId.get(),

self.var_phoneNo.get(),

self.var_address.get(),

self.var_tgName.get(),


 ))
                conn.commit()
                self.fetch_data()
                self.reset_data()
                conn.close()
                messagebox.showinfo("Success","Employee
details has been added successfully",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",user="root",pa
ssword="",database="face_recognizer")
        my_cursor=conn.cursor()
        my_cursor.execute("SELECT * FROM student")
        data=my_cursor.fetchall()

self.student_table.delete(*self.student_table.get_children()
)

        if len(data)!=0:

self.student_table.delete(*self.student_table.get_children()
)
            for i in data:
                self.student_table.insert("",END,values=i)
            conn.commit()
        conn.close()
    #GET CURSOR
```

```
    def get_cursor(self,event=""):
        Cursor_focus=self.student_table.focus()
        content=self.student_table.item(Cursor_focus)
#contain store in entry fill
        data=content["values"]
        self.var_dep.set(data[0]),
        self.var_course.set(data[1]),
        self.var_year.set(data[2]),
        self.var_semester.set(data[3]),
        self.var_studentSNo.set(data[4]),
        self.var_studentName.set(data[5]),
        self.var_classSection.set(data[6]),
        self.var_enrollNo.set(data[7]),
        self.var_gender.set(data[8]),
        self.var_studentDOB.set(data[9]),
        self.var_emailId.set(data[10]),
        self.var_phoneNo.set(data[11]),
        self.var_address.set(data[12]),
        self.var_tgName.set(data[13])

#     #UPDATE FUNCTION
#     #UPDATE FUNCTION
    def update_data(self):
        if self.var_dep.get()=="Select Position" or
self.var_studentName.get()=="" or
self.var_studentSNo.get()==""  or
self.var_course.get()=="Select Religion" or
self.var_year.get()=="Select Year"  or
self.var_semester.get()=="Select Status" or
self.var_classSection.get() == "Select Tier" or
self.var_gender.get() == "Select Gender" or
self.var_enrollNo.get()=="" or self.var_phoneNo.get()=="":
        # if self.var_dep.get()=="Select Department" or
self.var_studentName.get()=="" or
self.var_studentSNo.get()=="":
            messagebox.showerror("Error!","All Fields are
required",parent=self.root)
        else:
            try:
                Update=messagebox.askyesno("Update","Do you
want to update this employee details",parent=self.root)
                if Update >0:

conn=mysql.connector.connect(host="localhost",user="root",pa
ssword="",database="face_recognizer")
                    my_cursor=conn.cursor()
                    my_cursor.execute("Update student set
dep=%s,course=%s,year=%s,semester=%s,studentName=%s,classSec
tion=%s,enrollNo=%s,gender=%s,studentDOB=%s,emailId=%s,phone
No=%s,address=%s,tgName=%s where studentSNo=%s",(
```

```
self.var_dep.get(),

self.var_course.get(),

self.var_year.get(),

self.var_semester.get(),

self.var_studentName.get(),

self.var_classSection.get(),

self.var_enrollNo.get(),

self.var_gender.get(),

self.var_studentDOB.get(),

self.var_emailId.get(),

self.var_phoneNo.get(),

self.var_address.get(),

self.var_tgName.get(),

self.var_studentSNo.get()

))
                else:
                 if not Update:
                    return
                messagebox.showinfo("Sucsess", "Employee
Details Update has been completed
successfully",parent=self.root)
                conn.commit()
                self.fetch_data()
                conn.close()
            except Exception as es:
                messagebox.showerror("Error!",f"Due
to :{str(es)}",parent=self.root)

    #DELETE FUNCTION
    def delete_data(self):
        if self.var_studentSNo.get()=="":
            messagebox.showerror("Error!","Employee Serial
Number must be required",parent=self.root)
        else:
            try:
```

```
                delete=messagebox.askyesno("Delete","Do you
want to delete this employee details",parent=self.root)
                if delete >0:

conn=mysql.connector.connect(host="localhost",user="root",pa
ssword="",database="face_recognizer")
                    my_cursor=conn.cursor()
                    sql="delete from student where
studentSNo=%s"
                    val=(self.var_studentSNo.get(), )
                    my_cursor.execute(sql,val)
                else:
                    if not delete:
                        return
                conn.commit()
                self.fetch_data()
                self.reset_data()
                conn.close()
                messagebox.showinfo("Delete", "Employee
Details has been  deleted successfulluy ",parent=self.root)
            except Exception as es:
                messagebox.showerror("Error!",f"Due
to :{str(es)}",parent=self.root)

    #Reset Function
    def reset_data(self):
        self.var_dep.set("Select Position")
        self.var_course.set("Select Religion")
        self.var_year.set("Select Year")
        self.var_semester.set("Select Status")
        self.var_studentSNo.set("")
        self.var_studentName.set("")
        self.var_classSection.set("Select Tier")
        self.var_enrollNo.set("")
        self.var_gender.set("Select Gender")
        self.var_studentDOB.set("")
        self.var_emailId.set("")
        self.var_phoneNo.set("")
        self.var_address.set("")
        self.var_tgName.set("")


    # Generate data set or take photo samples
    def generate_dataset(self):
        if self.var_dep.get()=="Select Position" or
self.var_studentName.get()=="" or
self.var_studentSNo.get()==""  or
self.var_course.get()=="Select Religion" or
self.var_year.get()=="Select Year"  or
self.var_semester.get()=="Select Status" or
self.var_classSection.get() == "Select Tier" or
```

```
self.var_gender.get() == "Select Gender" or
self.var_enrollNo.get()=="" or self.var_phoneNo.get()=="":
            messagebox.showerror("Error!","All Fields are
required",parent=self.root)
        else:
            try:

conn=mysql.connector.connect(host="localhost",user="root",pa
ssword="",database="face_recognizer")
                my_cursor=conn.cursor()
                my_cursor.execute("select * from student")
                myresult=my_cursor.fetchall()
                id=0
                for x in myresult:
                    id+=1
                my_cursor.execute("Update student set
dep=%s,course=%s,year=%s,semester=%s,studentName=%s,classSec
tion=%s,enrollNo=%s,gender=%s,studentDOB=%s,emailId=%s,phone
No=%s,address=%s,tgName=%s where studentSNo=%s",(


self.var_dep.get(),

self.var_course.get(),

self.var_year.get(),

self.var_semester.get(),

self.var_studentName.get(),

self.var_classSection.get(),

self.var_enrollNo.get(),

self.var_gender.get(),

self.var_studentDOB.get(),

self.var_emailId.get(),

self.var_phoneNo.get(),

self.var_address.get(),

self.var_tgName.get(),

self.var_studentSNo.get()==id+1

        ))
                conn.commit()
```

```
                self.fetch_data()
                self.reset_data()
                conn.close()
                #Load predifiend data on face frontals from
opencv
                face_classifier =
cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
                def face_cropped(img):


gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces=face_classifier.detectMultiScale(gray,1.3,5)
                    #scaling factor -1.3
                    #Minimal 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)
                cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
                cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
                img_id=0
                while True:
                   ret,my_frame=cap.read()
                   if face_cropped(my_frame) is not None:
                       img_id+=1

face=cv2.resize(face_cropped(my_frame),(350,350))

face=cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)
#patha user name#

file_name_path="data/user."+str(id)+"."+str(img_id)+".jpg"
                        cv2.imwrite(file_name_path,face)
#Saves an image to a specified file or any storage device.

cv2.putText(face,str(img_id),(50,50),cv2.FONT_HERSHEY_COMPLE
X,2,(0,225,0),2)  #draw a text string on any image
                        cv2.imshow("Cropped Face",face)
# display an image in a window

                    if cv2.waitKey(1)==13 or
int(img_id)==100:
                        break
                cap.release()
                cv2.destroyAllWindows()
                messagebox.showinfo("Result","Generating
Data Sets have been completed successfully! ")
            except Exception as es:
```

```
                messagebox.showerror("Error!",f"Due
to :{str(es)}",parent=self.root)


    def Upload_photo(self):
        Uphoto = filedialog.askopenfilename(initialdir="/",
title="Select Photo", filetype=(("jpeg", "*.jpg"), ("png",
"*.png")))

        # Create a new Label widget to display the selected
photo path
        label = ttk.Label(self.root, text="")
        label.grid(column=1, row=2)
        label.configure(text=Uphoto)

        main_py_directory =
os.path.dirname(os.path.abspath(__file__))

        # Specify the target directory where you have write
permissions
        target_directory = main_py_directory + "/data"

        # Create the target directory if it doesn't exist
        os.makedirs(target_directory, exist_ok=True)

        # Get the filename from the full path
        filename = os.path.basename(Uphoto)

        # Construct the full destination path
        destination_path = os.path.join(target_directory,
filename)

        # Move the file to the target directory
        shutil.move(Uphoto, destination_path)

        # Update the label text with the new destination
path
        label.configure(text=destination_path)




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

```
    # pady mean social distancing like diff b/w to button
    #pillow  python image in library
    #ANTIALIAS IS DESRIPTEF DUE TO USE  Resampling.LANCZOS
    # opencv module import cv2
```

employee_attendance.py

```
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
from time import strftime
from datetime import datetime
import os
import csv
from tkinter import filedialog
# from unicodedata import name
# import mysql.connector
# import numpy as np
# import cv2


myData=[]
class Student_Attendance:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1510x790+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')

    #Variables
        # self.var_attendace_studentSNo=StringVar()
        self.var_attendace_studentName=StringVar()
        self.var_attendace_enrollNo=StringVar()
        self.var_attendace_dep=StringVar()
        self.var_attendace_date=StringVar()
        self.var_attendace_time=StringVar()
        self.var_attendace_attendance=StringVar()



    #First Image Upload
        img=Image.open(r"Project_Image\hand.jpg")
        img=img.resize((500,190),Image.Resampling.LANCZOS)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=500,height=150)
```

```
    #Second1 Image Upload
        img1=Image.open(r"Project_Image\attendance.jpeg")
        img1=img1.resize((500,150),Image.Resampling.LANCZOS)
        self.photoimg1=ImageTk.PhotoImage(img1)
        f_lbl=Label(self.root,image=self.photoimg1)
        f_lbl.place(x=490,y=0,width=500,height=150)

    #Third Image Upload
        img2=Image.open(r"Project_Image\attendace.jpg")
        img2=img2.resize((500,170),Image.Resampling.LANCZOS)
        self.photoimg2=ImageTk.PhotoImage(img2)
        f_lbl=Label(self.root,image=self.photoimg2)
        f_lbl.place(x=950,y=0,width=500,height=150)

    #background Image Upload
        img3=Image.open(r"Project_Image\bg.webp")

img3=img3.resize((1530,710),Image.Resampling.LANCZOS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        bg_img=Label(self.root,image=self.photoimg3)
        bg_img.place(x=0,y=120,width=1530,height=710)

        title_lbl=Label(bg_img,text="ATTENDANCE MANAGEMENT
STSTEM",font=("times new
roman",25,"bold"),bg="white",fg="darkgreen")
        title_lbl.place(x=0,y=0,width=1430,height=50)

        # Real world time
        def time():
                string = strftime('%I:%M:%S %p  %d/%m/%Y')
                lbl.config(text = string)
                lbl.after(1000, time)

        lbl = Label(title_lbl, font=('times new
roman',20,'bold'), bg='white',fg='darkblue')
        lbl.place(x=5,y=0,width=350,height=50)
        time()

    #main frame in student
        main_frame=Frame(bg_img,bd=2)
        main_frame.place(x=20,y=45,width=1320,height=520)

    #left label frame

Left_lframe=LabelFrame(main_frame,bd=2,relief=RIDGE,text="Em
ployee Attendance Details",font=("times new
roman",18,"bold"))
        Left_lframe.place(x=10,y=10,width=650,height=550)

        img_left=Image.open(r"Project_Image\rightpic.jpg")
```

```
img_left=img_left.resize((640,100),Image.Resampling.LANCZOS)
        self.photoimg_left=ImageTk.PhotoImage(img_left)

        f_lbl=Label(Left_lframe,image=self.photoimg_left)
        f_lbl.place(x=0,y=0,width=650,height=100)

    #current attendace information

current_attendace_frame=LabelFrame(Left_lframe ,bd=2,relief=
RIDGE,text="Current Course information :",font=("times new
roman",15,"bold"))

current_attendace_frame.place(x=10,y=100 ,width=620,height=3
50)
    #Label & Entry
    #student serial no
        #
studentSNo_label=Label(current_attendace_frame,text="Student
S.No. :",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")
        #
studentSNo_label.grid(row=0,column=0,padx=2,pady=2,sticky=W)
    # student s.no. entry
        #
studentSNo_entry=ttk.Entry(current_attendace_frame,textvaria
ble=self.var_attendace_studentSNo,font=("times new
roman",14,"bold"),width=13)
        #
studentSNo_entry.grid(row=0,column=1,padx=2,pady=0,sticky=W)

#   # student Name label uncomment from here
    #
studentName_label=Label(current_attendace_frame,text="Studen
t Name :",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")
    #
studentName_label.grid(row=0,column=2,padx=2,pady=2,sticky=W
)
    # #student name entry
    #
studentName_entry=ttk.Entry(current_attendace_frame,textvari
able=self.var_attendace_studentName,font=("times new
roman",14,"bold"),width=13)
    #
studentName_entry.grid(row=0,column=3,padx=2,pady=2,sticky=W
)


    # #student Enrollment numbber label
```

```
    #
enrollNo_label=Label(current_attendace_frame,text="Enrollmen
t No. :",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")
    #
enrollNo_label.grid(row=1,column=0,padx=2,pady=5,sticky=W)
    # #student Enrollment numbber entry
    #
enrollNo_entry=ttk.Entry(current_attendace_frame,textvariabl
e=self.var_attendace_enrollNo,font=("times new
roman",14,"bold"),width=13)
    #
enrollNo_entry.grid(row=1,column=1,padx=2,pady=5,sticky=W)


    # #Department
    #
dep_label=Label(current_attendace_frame,text="Department :",
font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")
    #     dep_label.grid(row=1,column=2,padx=2,sticky=W)
    # #student department name entry
    #
dep_label_entry=ttk.Entry(current_attendace_frame,textvariab
le=self.var_attendace_dep,font=("times new
roman",14,"bold"),width=13)
    #
dep_label_entry.grid(row=1,column=3,padx=2,pady=5,sticky=W)


    # #Attendace Date label
    #
date_label=Label(current_attendace_frame,text="Date :",font=
("times new roman",14,"bold"),bg="white",fg="darkgreen")
    #     date_label.grid(row=2,column=0,padx=2,sticky=W)
    # #Attendance date entry
    #
date_label_entry=ttk.Entry(current_attendace_frame,textvaria
ble=self.var_attendace_date,font=("times new
roman",14,"bold"),width=13)
    #
date_label_entry.grid(row=2,column=1,padx=2,pady=5,sticky=W)

    # #Attendace Time label
    #
date_label=Label(current_attendace_frame,text="Time :",font=
("times new roman",14,"bold"),bg="white",fg="darkgreen")
    #     date_label.grid(row=2,column=2,padx=2,sticky=W)
    # #Attendance time entry
    #
date_label_entry=ttk.Entry(current_attendace_frame,textvaria
```

```
ble=self.var_attendace_time,font=("times new
roman",14,"bold"),width=13)
    #
date_label_entry.grid(row=2,column=3,padx=2,pady=5,sticky=W)


    # #Attendace status
    #
attendace_label=Label(current_attendace_frame,text="Attendan
ce Status",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")
    #
attendace_label.grid(row=3,column=0,padx=2,pady=5,sticky=W)
    # #attendace status combobox
    #
self.attendance_status=ttk.Combobox(current_attendace_frame,
textvariable=self.var_attendace_attendance,font=("times new
roman",14,"bold"),state="readonly",width=11)
    #     self.attendance_status["values"]=("Select Status
","Present", "Absent")
    #     self.attendance_status.current(0)
    #     self.attendance_status.grid(row=3,column=1,padx=2,
pady=5,sticky=W)









        #student Name label

studentName_label=Label(current_attendace_frame,text="Employ
ee Name :",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")

studentName_label.grid(row=0,column=0,padx=2,pady=2,sticky=W
)
    #student name entry
```

```
studentName_entry=ttk.Entry(current_attendace_frame,textvari
able=self.var_attendace_studentName,font=("times new
roman",14,"bold"),width=13)

studentName_entry.grid(row=0,column=1,padx=2,pady=2,sticky=W
)


    #student Enrollment numbber label

enrollNo_label=Label(current_attendace_frame,text="Enrollmen
t No. :",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")

enrollNo_label.grid(row=0,column=2,padx=2,pady=5,sticky=W)
    #student Enrollment numbber entry

enrollNo_entry=ttk.Entry(current_attendace_frame,textvariabl
e=self.var_attendace_enrollNo,font=("times new
roman",14,"bold"),width=13)

enrollNo_entry.grid(row=0,column=3,padx=2,pady=5,sticky=W)


    #Department

dep_label=Label(current_attendace_frame,text="Position :",fo
nt=("times new roman",14,"bold"),bg="white",fg="darkgreen")
        dep_label.grid(row=1,column=0,padx=2,sticky=W)
    #student department name entry

dep_label_entry=ttk.Entry(current_attendace_frame,textvariab
le=self.var_attendace_dep,font=("times new
roman",14,"bold"),width=13)

dep_label_entry.grid(row=1,column=1,padx=2,pady=5,sticky=W)


    #Attendace Date label

time_label=Label(current_attendace_frame,text="Time :",font=
("times new roman",14,"bold"),bg="white",fg="darkgreen")
        time_label.grid(row=1,column=2,padx=2,sticky=W)
    #Attendance date entry

time_label_entry=ttk.Entry(current_attendace_frame,textvaria
ble=self.var_attendace_date,font=("times new
roman",14,"bold"),width=13)

time_label_entry.grid(row=1,column=3,padx=2,pady=5,sticky=W)
```

```
    #Attendace Time label

date_label=Label(current_attendace_frame,text="Date :",font=
("times new roman",14,"bold"),bg="white",fg="darkgreen")
        date_label.grid(row=2,column=0,padx=2,sticky=W)
    #Attendance time entry

date_label_entry=ttk.Entry(current_attendace_frame,textvaria
ble=self.var_attendace_time,font=("times new
roman",14,"bold"),width=13)

date_label_entry.grid(row=2,column=1,padx=2,pady=5,sticky=W)


    #Attendace status

attendace_label=Label(current_attendace_frame,text="Attendan
ce Status",font=("times new
roman",14,"bold"),bg="white",fg="darkgreen")

attendace_label.grid(row=2,column=2,padx=2,pady=5,sticky=W)
    #attendace status combobox

self.attendance_status=ttk.Combobox(current_attendace_frame,
textvariable=self.var_attendace_attendance,font=("times new
roman",14,"bold"),state="readonly",width=11)
        self.attendance_status["values"]=("Select Status
","Present", "Absent")
        self.attendance_status.current(0)
        self.attendance_status.grid(row=2,column=3,padx=2,
pady=5,sticky=W)


#   #Button frame

btn_frame=Frame(current_attendace_frame,bd=2,relief=RIDGE,bg
="white")
        btn_frame.place(x=0,y=300,width=620,height=80)

        import_csv_btn=Button(btn_frame,text="IMPORT
CSV",command=self.importCsv,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        import_csv_btn.grid(row=0,column=0)

        export_csv_btn=Button(btn_frame,text="EXPORT
CVS",command=self.exportCsv,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        export_csv_btn.grid(row=0,column=1)
```

```
update_btn=Button(btn_frame,text="UPDATE",width=21,font=("ti
mes new roman",10,"bold"),bg="blue",fg="white")
        update_btn.grid(row=0,column=2)


reset_btn=Button(btn_frame,text="RESET",command=self.reset_d
ate,width=21,font=("times new
roman",10,"bold"),bg="blue",fg="white")
        reset_btn.grid(row=0,column=3)



    #Right frame

Right_frame=LabelFrame(main_frame,bd=2,relief=RIDGE,text="At
tendance Details",font=("times new roman",15,"bold"))
        Right_frame.place(x=670,y=10,width=640,height=500)
    #table frame ==========

table_frame=LabelFrame(Right_frame,bd=2,bg="white",relief=RI
DGE)
        table_frame.place(x=5,y=5,width=630,height=470)
    #Scroll bar =======

scroll_x=ttk.Scrollbar(table_frame,orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar(table_frame,orient=VERTICAL)

        #
self.AttendanceReportTable=ttk.Treeview(table_frame,column=(
"studentSNo","studentName","enrollNo","dep","date","time","a
ttendance"),xscrollcommand=scroll_x.set,yscrollcommand=scrol
l_y.set)

self.AttendanceReportTable=ttk.Treeview(table_frame,column=(
"studentName","enrollNo","dep","time","date","attendance"),x
scrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)


scroll_x.config(command=self.AttendanceReportTable.xview)

scroll_y.config(command=self.AttendanceReportTable.yview)

        #
self.AttendanceReportTable.heading("studentSNo",text="Studen
t S.No.")

self.AttendanceReportTable.heading("studentName",text="Emplo
yee Name")
```

```
self.AttendanceReportTable.heading("enrollNo",text="Enrollme
nt No.")

self.AttendanceReportTable.heading("dep",text="Position")

self.AttendanceReportTable.heading("date",text="Date")

self.AttendanceReportTable.heading("time",text="Time")

self.AttendanceReportTable.heading("attendance",text="Attend
ance")
        self.AttendanceReportTable["show"]="headings"


        #
self.AttendanceReportTable.column("studentSNo",width=100)

self.AttendanceReportTable.column("studentName",width=100)

self.AttendanceReportTable.column("enrollNo",width=100)
        self.AttendanceReportTable.column("dep",width=150)
        self.AttendanceReportTable.column("date",width=100)
        self.AttendanceReportTable.column("time",width=100)

self.AttendanceReportTable.column("attendance",width=100)

        self.AttendanceReportTable.pack(fill=BOTH,expand=1)


self.AttendanceReportTable.bind("<ButtonRelease>",self.get_c
ursor)

        # ===========Fatch Data===============
    def fetchData(self, rows):

self.AttendanceReportTable.delete(*self.AttendanceReportTabl
e.get_children())
        for i in rows:

self.AttendanceReportTable.insert("",END,values=i)
    #Import CSV
    def importCsv(self):
        global myData
        myData.clear()

fln=filedialog.askopenfilename(initialdir=os.getcwd(),title=
"Open CSV",filetypes=(("CSV File","*.csv"),("ALL
File","*.*")),parent=self.root)
        with open(fln) as myfile:
            csvread=csv.reader(myfile,delimiter=",")
```

```
            for i in csvread:
                myData.append(i)
            self.fetchData(myData)

    def exportCsv(self):
        try:
            if len(myData)<1:
                messagebox.showerror("NO DATA","No data has
been found to Export",parent=self.root)
                return False

fln=filedialog.asksaveasfilename(initialdir=os.getcwd(),titl
e="Open CSV",filetypes=(("CSV File","*.csv"),("ALL
File","*.*")),parent=self.root)
            with open(fln,mode="w",newline="") as myfile:

exp_csv_write=csv.writer(myfile,delimiter=",")
                for i in myData:
                    exp_csv_write.writerow(i)
                messagebox.showinfo("DATA EXPORT","Your data
has been exported to" +os.path.basename(fln)+"
successfully")
        except Exception as es:
                messagebox.showerror("Error!",f"Due
to :{str(es)}",parent=self.root)


    def get_cursor(self,event=""):
        cursor_row=self.AttendanceReportTable.focus()
        content=self.AttendanceReportTable.item(cursor_row)
        rows=content["values"]
        # self.var_attendace_studentSNo.set(rows[0])
        # self.var_attendace_studentName.set(rows[1])
        # self.var_attendace_enrollNo.set(rows[2])
        # self.var_attendace_dep.set(rows[3])
        # self.var_attendace_date.set(rows[4])
        # self.var_attendace_time.set(rows[5])
        # self.var_attendace_attendance.set(rows[6])

        self.var_attendace_studentName.set(rows[0])
        self.var_attendace_enrollNo.set(rows[1])
        self.var_attendace_dep.set(rows[2])
        self.var_attendace_date.set(rows[3])
        self.var_attendace_time.set(rows[4])
        self.var_attendace_attendance.set(rows[5])


    def reset_date(self):
        # self.var_attendace_studentSNo.set("")
        self.var_attendace_studentName.set("")
        self.var_attendace_enrollNo.set("")
```

```
        self.var_attendace_dep.set("")
        self.var_attendace_date.set("")
        self.var_attendace_time.set("")
        self.var_attendace_attendance.set("")



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

train.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import messagebox
import cv2
import os
import numpy as np
# import mysql.connector
# from tkinter import ttk



class Train_Data:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1510x790+0+0")
        self.root.title("Automatic Attendance Management
System")
        root.iconbitmap('clgicon.ico')

        #Tittle of MainFrame
        title_lbl=Label(self.root,text='''TRAIN DATA
SET''',font=("times new
roman",25,"bold"),bg="white",fg="darkgreen")
        title_lbl.place(x=0,y=0,width=1410,height=50)

        #Top_Image
        img_top=Image.open(r"Project_Image\training
images.jpg")

img_top=img_top.resize((1410,240),Image.Resampling.LANCZOS)
        self.photoimg_top=ImageTk.PhotoImage(img_top)
        f_lbl=Label(self.root,image=self.photoimg_top)
        f_lbl.place(x=0,y=50,width=1410,height=240)
        #Button
```

```
        b1=Button(self.root,text="TRAIN DATA \n Click
",cursor="hand2",command=self.train_classifier,font=("times
new roman",30,"bold"),bg="darkblue",fg="white")
        b1.place(x=0,y=300,width=1410,height=100)
        #Down Image
        img_bottom=Image.open(r"Project_Image\train1.jpg")

img_bottom=img_bottom.resize((1510,375),Image.Resampling.LAN
CZOS)
        self.photoimg_bottom=ImageTk.PhotoImage(img_bottom)
        f_lbl=Label(self.root,image=self.photoimg_bottom)
        f_lbl.place(x=0,y=410,width=1510,height=300)

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

        faces=[]
        ids=[]

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

            faces.append(imageNp)
            ids.append(id)
            cv2.imshow("Training Photos Sample",imageNp)
            cv2.waitKey(1)==13
        ids=np.array(ids) # use for 90% performane increase
to covert RMA


        #Train the classifir and save
        clf=cv2.face.LBPHFaceRecognizer_create()
        # clf=cv2.face.createLBPHFaceRecognizer()
        clf.train(faces,ids)
        clf.write("classifier.xml")
        cv2.destroyAllWindows()
        messagebox.showinfo("Result","Training  Data Sets
have beeen completed!")

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

face_recognizer.sql

```
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Oct 31, 2023 at 08:15 AM
-- Server version: 10.4.28-MariaDB
-- PHP Version: 8.0.28

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `face_recognizer`
--

-- --------------------------------------------------------

--
-- Table structure for table `student`
--

CREATE TABLE `student` (
  `dep` text NOT NULL,
  `course` text NOT NULL,
  `year` text NOT NULL,
  `semester` text NOT NULL,
  `studentSNo` int(11) NOT NULL,
  `studentName` text NOT NULL,
  `classSection` text NOT NULL,
  `enrollNo` text NOT NULL,
  `gender` text NOT NULL,
  `studentDOB` text NOT NULL,
  `emailId` text NOT NULL,
  `phoneNo` text NOT NULL,
  `address` text NOT NULL,
  `tgName` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

--
```

```
-- Dumping data for table `student`
--

INSERT INTO `student` (`dep`, `course`, `year`, `semester`,
`studentSNo`, `studentName`, `classSection`, `enrollNo`,
`gender`, `studentDOB`, `emailId`, `phoneNo`, `address`,
`tgName`) VALUES
('Human Resources Development', 'Muslim', '2003', 'Single',
1, 'Dimas Agung Samudra', 'SMA', '1', 'Male', '18 November
1988', 'dimas72519123@gmail.com', '621374431922', 'Jalan
Raya No. 8', 'Dimas'),
('Chief Financial Officer', 'Catholic', '2006', 'Single', 2,
'Andry', 'S1', '2', 'Male', '18 Oktober 1998',
'andryandry18@gmail.com', '625821968462', 'Jln Sukaraya No.
18', 'Andry'),
('Marketing Manager', 'Muslim', '2005', 'Single', 3, 'Abi
Taufik', 'S1', '3', 'Male', '3 July 1994',
'abivabi81@gmail.com', '6281246462811', 'Jalan Bunga Wijaya
Kesuma No.31', 'Abi '),
('Chief Financial Officer', 'Muslim', '2000', 'Single', 4,
'Melissa Putri', 'SMA', '4', 'Female', '28 April 2000',
'melissaputri31@gmail.com', '6288707223505', 'Jalan
Sidomulio No. 5', 'Melissa'),
('Chief Marketing Officer', 'Budhist', '2020', 'Single', 5,
'Stephanie ', 'S1', '5', 'Female', '15 Nov 1999',
'Stephanie19@gmail.com', '628383728392', 'Jalan Multatuli
No.31', 'Stephanie'),
('Chief Operation Officer', 'Hindu', '2019', 'Single', 6,
'Eileen', 'S1', '6', 'Female', '12 May 1999',
'eileeneileen87@gmail.com', '6282737273823', 'Jln Dipenogoro
No. 13', 'Eileen'),
('Chief Financial Officer', 'Catholic', '2011', 'Single', 7,
'Arif', 'SMA', '7', 'Male', '8 Oktober 1992',
'arifmuhammad8@gmail.com', '62838232323', 'Jalan Multak
No.15', 'Arif'),
('Product Manager', 'Muslim', '2016', 'Single', 8, 'Nuri',
'S1', '8', 'Female', '3 July 1997', 'NNNuri9191@gmail.com',
'629273921731', 'Jalan Teladan No. 6', 'Nuri'),
('Human Resources Development', 'Muslim', '2004', 'Married',
9, 'Mayumi', 'S1', '9', 'Female', '15 November 1997',
'mayumide97@gmail.com', '62782193822', 'Jln Diponegoro No.
88', 'Mayumi'),
('Chief Operation Officer', 'Budhist', '2016', 'Married',
10, 'Andre Atmaja', 'S1', '10', 'Male', '7 January 1993',
'dremaja93@gmail.com', '6283628682362', 'Jalan Mulia Raya
No. 18', 'Andre'),
('Marketing Manager', 'Hindu', '2018', 'Single', 11,
'Alessandro', 'SMA', '11', 'Male', '28 Oktober 1998',
'sandroales6@gmail.com', '62838283823', 'Jalan Nusa Raya No.
32', 'Alessandro');
```

```
--
-- Indexes for dumped tables
--

--
-- Indexes for table `student`
--
ALTER TABLE `student`
  ADD PRIMARY KEY (`studentSNo`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
*/;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;
```