FACE RECOGNITION USING OPENCV

A REPORT

Submitted by

DESHIREDDY KRISHNAREDDY (20MIS1069)

APPIREDDY GOWTHAM REDDY (20MIS1175)

In partial fulfilment for the award

Of

M.Tech. Integrated Software Engineering

School of Computer Science and Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

NOVEMBER 2022

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

We, DeshiReddy KrishnaReddy , Appireddy Gowthamreddy, hereby declare that the project entitled **“FACE RECOGNITION USING OPENCV”** submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements of **Project Component for the Course SWE1010- DIGITAL IMAGE PROCESSING** is a record of bona fide work carried out by me**.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, in any other institute or university.

ACKNOWLEDGEMENT

We would like to express my special thanks to Dr.GEETHA S, my teacher for this subject, who guided and helped me to make this project. At last we would like to thank my friends for giving me the support for completing.

CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Expansion
UAT	User Acceptance Testing
SRS	Software Requirement Specification
DDS	Data Design Specification
MIN	Minimum

ABSTRACT

Face recognition is the process of identifying or verifying an individual's identity by looking at their face. Face recognition for attendance systems is a procedure that uses bio-statistics and computer technology to recognize students' faces. As a result, attendance reports will be generated. The system is put through its paces under various scenarios before moving on to the following steps. The designed technology is cost-effective and requires minimal installation. The programme finds the face's distinct traits in the database and encodes them into a pattern image.

CHAPTER 1

Introduction

PURPOSE OF THE SYSTEM

- ✓ To save time of taking attendance and also reduces proxy attendance.
- ✓ To lower the burden of the staff.
- ✓ To make the work easy by using technology.

CHAPTER 2

2.3 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- **PROCESSOR:** Intel Core i5-8259U, or AMD Ryzen 7 2700X
- **GRAPHICS CARD:** NVIDIA GT 1030 2GB or Quadro P1000

Face recognition based attendance system

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

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

TECHNOLOGY USED:

1. tkinter for whole GUI
2. OpenCV for taking images and face recognition (cv2.face.LBPHFaceRecognizer_create())
3. CSV, Numpy, Pandas, datetime etc. for other purposes.

FEATURES:

1. Easy to use with interactive GUI support.
2. Password protection for new person registration.
3. Creates/Updates CSV file for deatils of students on registration.
4. Creates a new CSV file everyday for attendance and marks attendance with proper date and time.
5. Displays live attendance updates for the day on the main screen in tabular format with Id, name, date and time.

2.4 INPUT AND OUTPUT

The major inputs and outputs and major functions of the system are follows:

Input

- Faces of all the students should be scanned and uploaded to the database along with the student details (Name, ID, password etc).
- Then upload the necessary details to involve in the storing operation.

Output

- The pictures of the student faces which have been stored in the database will be recognized when they try scanning the face.
- The data will be seen directly in the database

2.5 INPUT DESIGN

- Input design is a part of overall system design. The main objective during the input design as given below.
- **Input States:** User sets the personal data to be stored in the database like name, reg.no etc.
- In the end when the face is recognized he can see his name also.
- **Input Media:** we can also review the details entered.

2.6 LIMITATIONS

- Massive data storage load: The ML technology used for facial recognition requires high-performance data storage that may not be available to all users.
- Detection is vulnerable.
- A potential breach of privacy.
- Difficult in maintaining the large amount of increased overhead.

2.7 DRAWBACKS IN EXISTING SYSTEM:

- Manual systems put pressure on the manual system put pressure on the people to be correct in all details of their work at all times the problem being that people are not perfect however each of us wishes we are
- These attendance systems are manual
- There is always a chance of forgery (one person say signing the presence of the other one) since these are manually so there is a great risk of error.
- More manpower is required
- Calculations related to attendance are done manually (total classes attended in a month) which is prone to error.
- It is difficult to maintain a database or register in manual system.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

Face recognition using OpenCV is about introducing Attendance system using face recognition is a procedure of recognizing students by using face biostatistics based on the high-definition monitoring and other computer technologies. ... The traditional process of making attendance and present biometric systems is vulnerable to proxies.

PURPOSE

The purpose of this Software Requirement Specification (SRS) is to help the project. Instead of using the conventional methods, this proposed system aims to develop an automated system that records the student's attendance by using facial recognition technology. The main objective of this work is to make the attendance marking and management system efficient, time saving, simple and easy.

3.2 FUNCTIONAL REQUIREMENTS

The system functional requirement describes activities and Services that must provide:

- Tracking and marking student attendance by Facial recognition in specific time.

- Allowing the faculty to modify the student Absent or late arrivals.

- Showing the names of students with the exact Time stamp i.e., exact time of entering the class.

3.3 NON-FUNCTIONAL REQUIREMENTS:

3.3.1 Usability

This system can be effectively used in taking attendance for Students, Teachers, Staffs in both schools and colleges, ATM's, identifying duplicate voters, passport and visa verification, driving license verification, in defense, competitive and other exams, in governments and private sectors.

3.3.2 Reliability

Face attendance is practically 100% accurate.

3.3.3 Performance

The system works on face recognition where each student in the class is photographed and their details are stored in a server. The teacher can then record the attendance by just clicking some pictures of the classroom. The system will recognize the faces and verify the presence or absence of each student.

3.3.4 Supportability

There are many benefits facial recognition can offer society, from preventing crimes and increasing safety and security to reducing unnecessary human interaction and labor. In some instances, it can even help support medical efforts.

3.3.5 Packaging

a. The system must be able to run on the Windows OS beginning with Windows 7, and must be able to run on future releases such as the upcoming Windows 10

b. The software must incorporate a license key authentication process.

c. The packaging must come with a manual that details the use of the system, and also the instructions on how to use the program. This manual may be included either in a booklet that comes with the software, or on the disc that the software itself is on.

3.2.6 Implementation -

The face recognition is implemented with the help of Principal Component Analysis (PCA) algorithm. The system will recognize the face of the student and saves the response in database automatically. The system also includes the feature of retrieving the list of students who are absent in a particular day.

3.2.7 Interfacing -

The system must offer an easy and simple way of viewing the attendance.

CHAPTER 4

SYSTEM DESIGN

4.3 Diagram

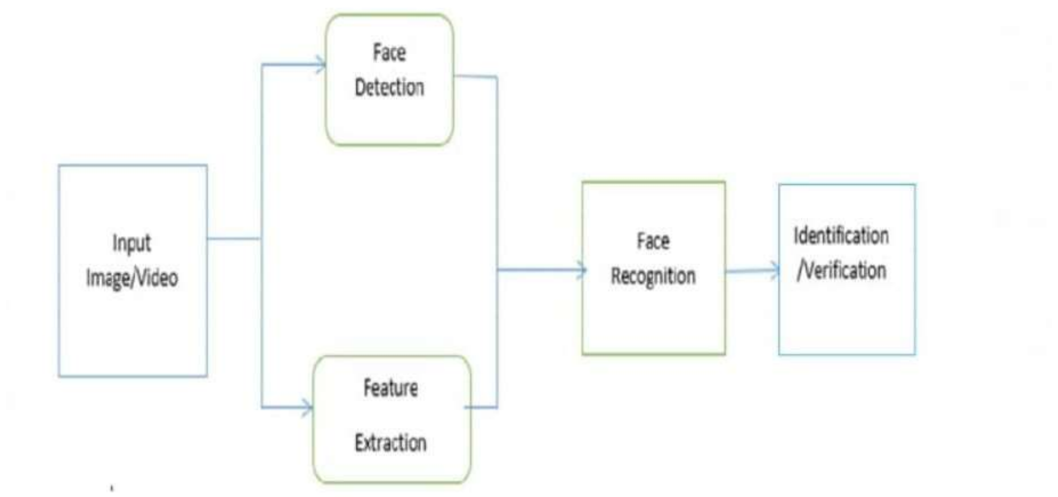


Figure 1: Block diagram of Face Detection

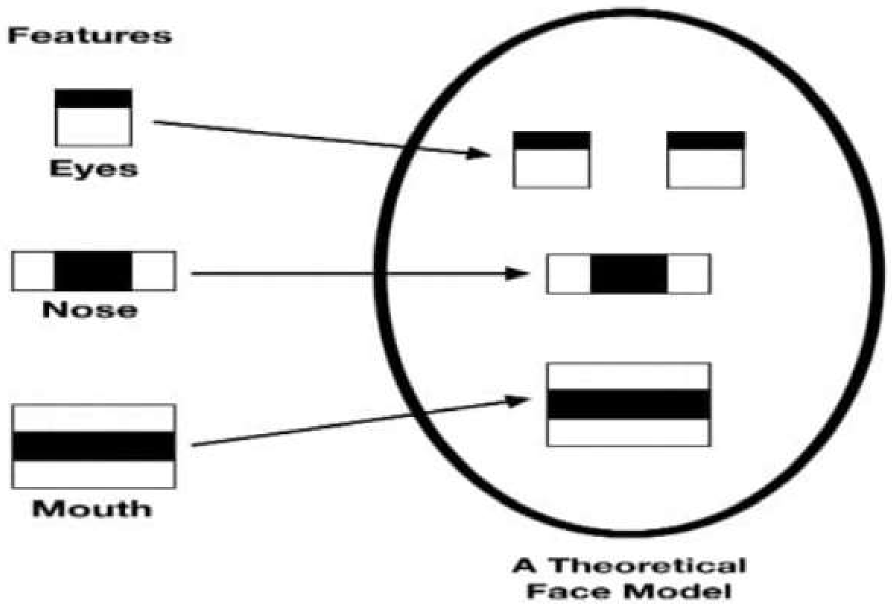
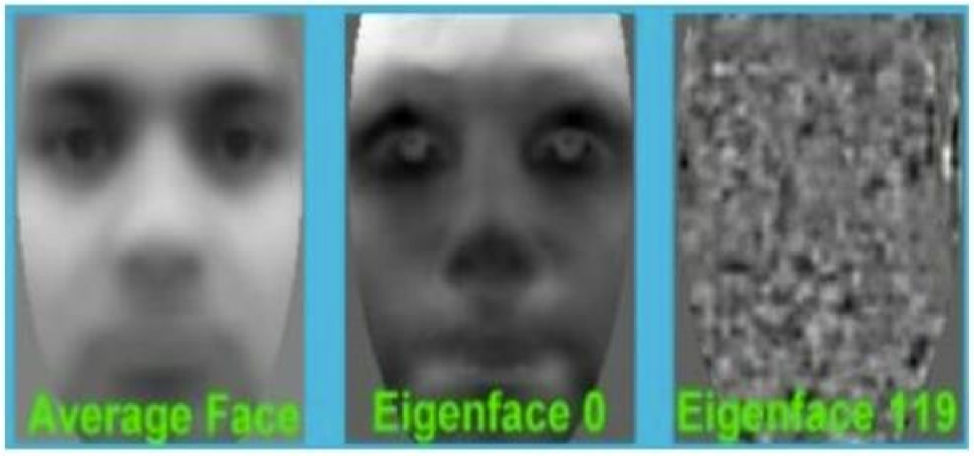
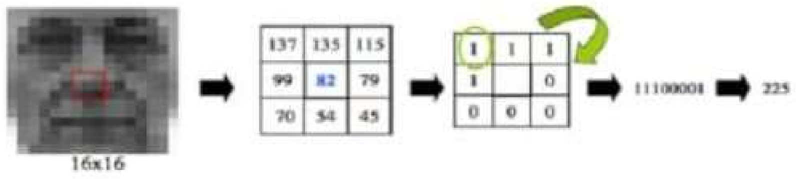


Figure 2: Detection of Theoretical Face model using Haar like features



Eigen Face representation for Face Recognition

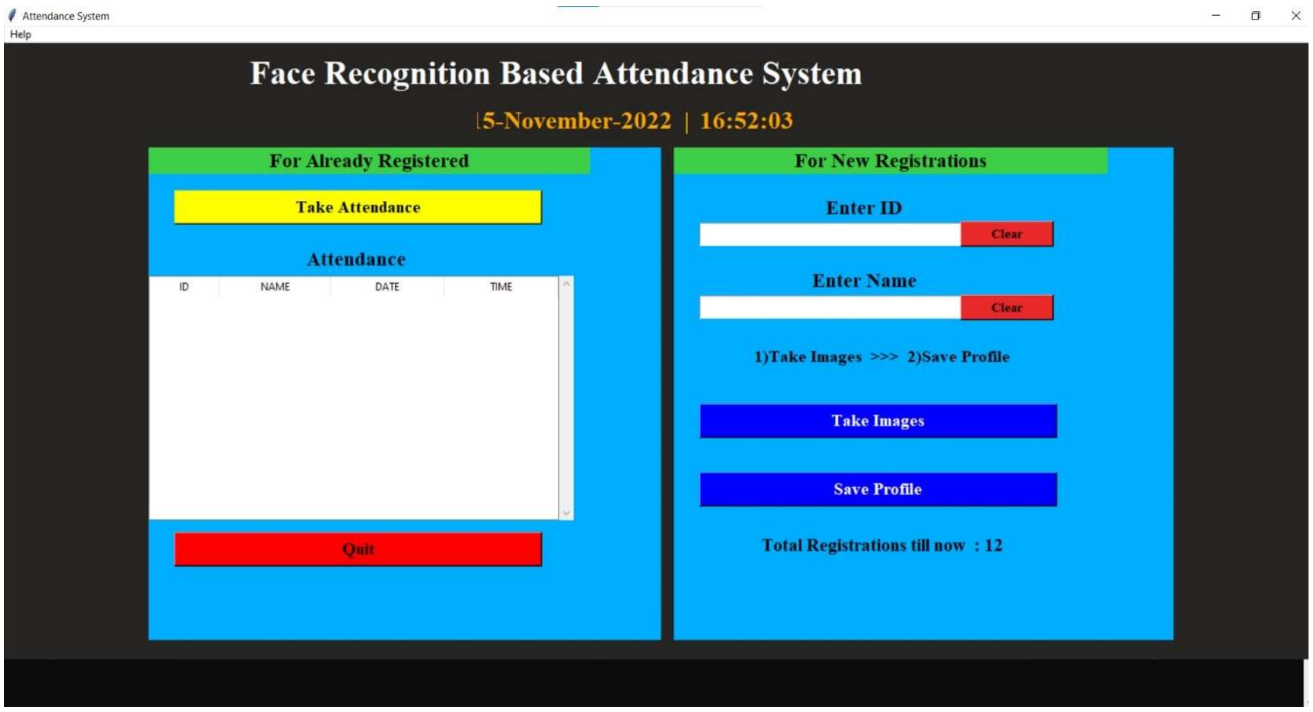


LBP Conversion to Binary

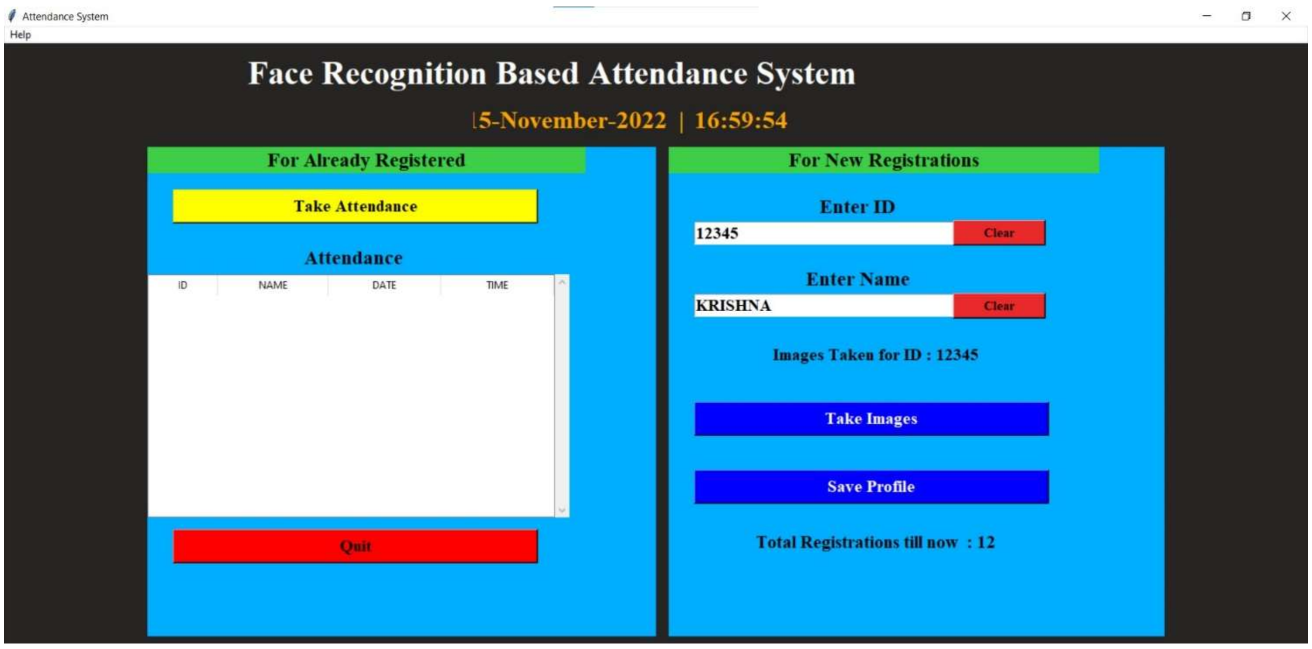
Chapter 5

SCREENSHOTS:

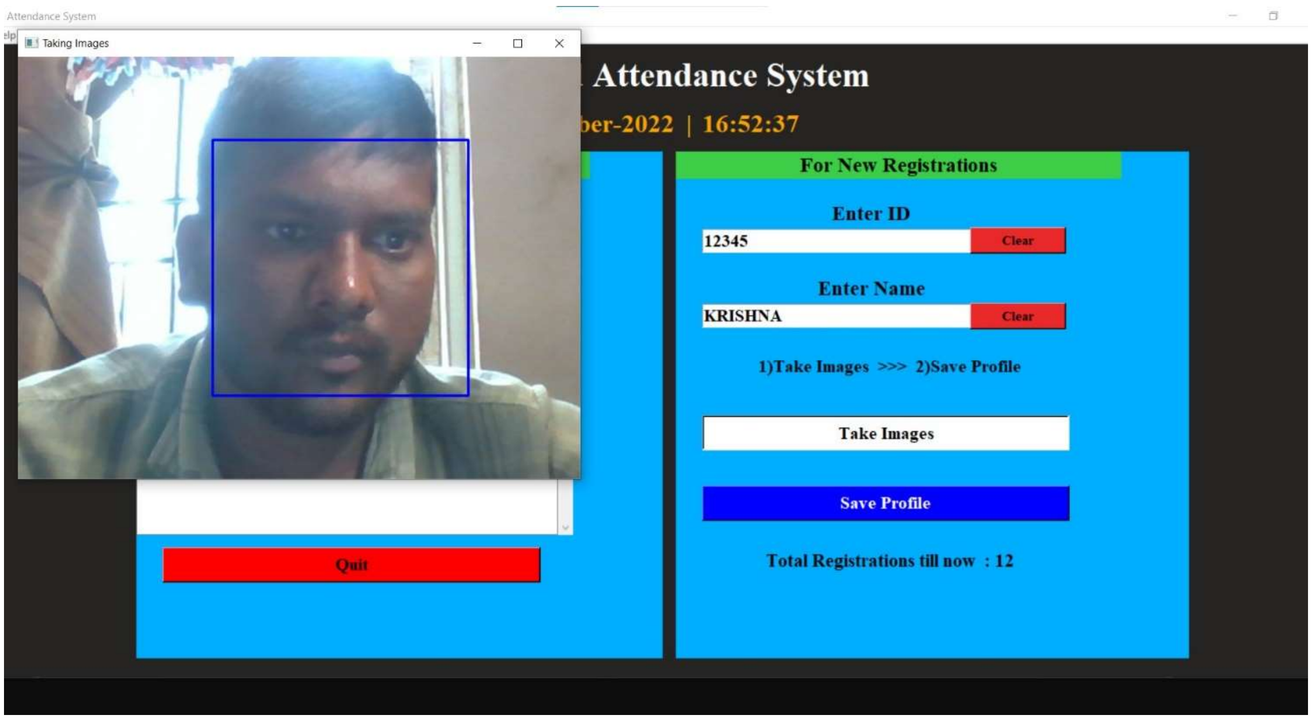
User Interface of Attendance System:



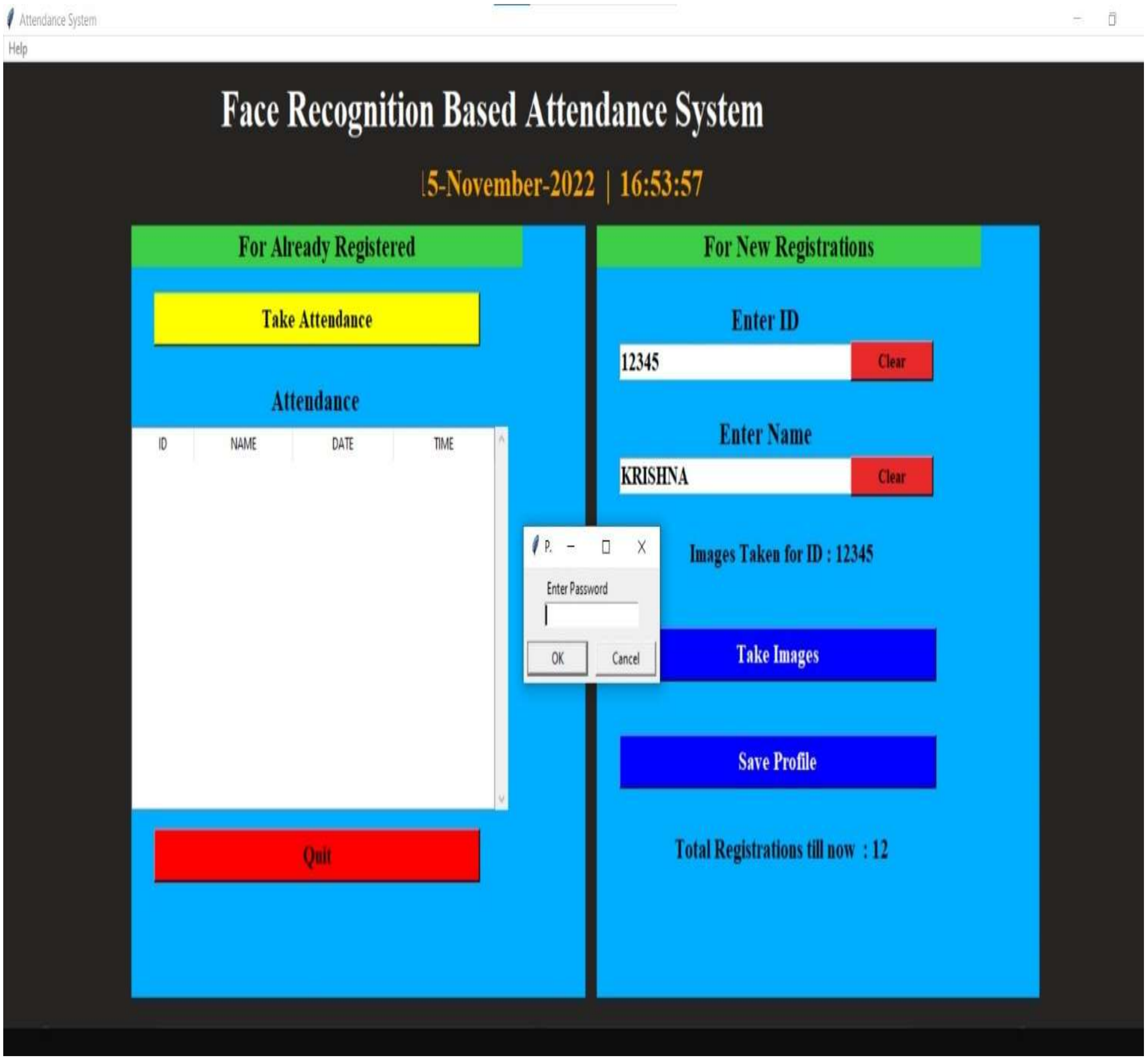
Enter ID, Enter NAME Of the person



Click on Take Images button , it will open camera and detect faces.



<u>Click Save Image button and Enter Password For privacy, it will save the face.</u>



CODES:

Face_reg.py

```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox as mess
import tkinter.simpledialog as tsd
import cv2,os
import csv
import numpy as np
from PIL import Image
import pandas as pd
import datetime
import time

############################################# FUNCTIONS
################################################

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

##################################################################################

def tick():
    time_string = time.strftime('%H:%M:%S')
    clock.config(text=time_string)
    clock.after(200,tick)

###################################################################################

def contact():
    mess._show(title='Contact us', message="Please contact us on :
'shubhamkumar8180323@gmail.com' ")

###################################################################################

def check_haarcascadefile():
    exists = os.path.isfile("haarcascade_frontalface_default.xml")
    if exists:
        pass
    else:
        mess._show(title='Some file missing', message='Please contact us for help')
        window.destroy()

###################################################################################

def save_pass():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!!
Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
    op = (old.get())
    newp= (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open("TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old
password.')
        return
    mess._show(title='Password Changed', message='Password changed successfully!!')
    master.destroy()

###################################################################################

def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
    master.configure(background="white")
    lbl4 = tk.Label(master,text='    Enter Old Password',bg='white',font=('times',
12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('times', 12, '
bold '),show='*')
    old.place(x=180,y=10)
    lbl5 = tk.Label(master, text='   Enter New Password', bg='white', font=('times',
12, ' bold '))
    lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('times', 12, '
bold '),show='*')
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('times',
12, ' bold '))
    lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('times', 12,
' bold '),show='*')
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy
,fg="black"  ,bg="red" ,height=1,width=25 , activebackground = "white"
,font=('times', 10, ' bold '))
    cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black",
bg="#3ece48", height = 1,width=25, activebackground="white", font=('times', 10, '
bold '))
    save1.place(x=10, y=120)
    master.mainloop()

####################################################################################
#

def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!!
Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
    password = tsd.askstring('Password', 'Enter Password', show='*')
    if (password == key):
        TrainImages()
    elif (password == None):
        pass
    else:
        mess._show(title='Wrong Password', message='You have entered wrong
password')

#####################################################################################
##

def clear():
    txt.delete(0, 'end')
    res = "1)Take Images  >>>  2)Save Profile"
    message1.configure(text=res)


def clear2():
    txt2.delete(0, 'end')
    res = "1)Take Images  >>>  2)Save Profile"
    message1.configure(text=res)

#####################################################################################
###

def TakeImages():
    check_haarcascadefile()
    columns = ['SERIAL NO.', '', 'ID', '', 'NAME']
    assure_path_exists("StudentDetails/")
    assure_path_exists("TrainingImage/")
    serial = 0
    exists = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists:
        with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
            reader1 = csv.reader(csvFile1)
            for l in reader1:
                serial = serial + 1
        serial = (serial // 2)
        csvFile1.close()
    else:
        with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(columns)
            serial = 1
        csvFile1.close()
    Id = (txt.get())
    name = (txt2.get())
    if ((name.isalpha()) or (' ' in name)):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0
        while (True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x, y, w, h) in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                # incrementing sample number
                sampleNum = sampleNum + 1
                # saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ " + name + "." + str(serial) + "." + Id
+ '.' + str(sampleNum) + ".jpg",
                            gray[y:y + h, x:x + w])
                # display the frame
                cv2.imshow('Taking Images', img)
            # wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum > 100:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Taken for ID : " + Id
        row = [serial, '', Id, '', name]
        with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message1.configure(text=res)
    else:
        if (name.isalpha() == False):
            res = "Enter Correct name"
            message.configure(text=res)
```

```
#####################################################################################
####

def TrainImages():
    check_haarcascadefile()
    assure_path_exists("TrainingImageLabel/")
    recognizer = cv2.createLBPHFaceRecognizer()
    #recognizer = cv2.faces.LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, ID = getImagesAndLabels("TrainingImage")
    try:
        recognizer.train(faces, np.array(ID))
    except:
        mess._show(title='No Registrations', message='Please Register someone
first!!!')
        return
    recognizer.save("TrainingImageLabel\Trainner.yml")
    res = "Profile Saved Successfully"
    message1.configure(text=res)
    message.configure(text='Total Registrations till now  : ' + str(ID[0]))

#####################################################################################
########3

def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    # create empth face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image
        ID = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(ID)
    return faces, Ids

#####################################################################################
#######
```

```
def TrackImages():
    check_haarcascadefile()
    assure_path_exists("Attendance/")
    assure_path_exists("StudentDetails/")
    for k in tv.get_children():
        tv.delete(k)
    msg = ''
    i = 0
    j = 0
    recognizer = cv2.createLBPHFaceRecognizer()  # cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")
    if exists3:
        recognizer.read("TrainingImageLabel\Trainner.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile to
reset data!!')
        return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)

    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', '', 'Name', '', 'Date', '', 'Time']
    exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists1:
        df = pd.read_csv("StudentDetails\StudentDetails.csv")
    else:
        mess._show(title='Details Missing', message='Students details are missing,
please check!')
        cam.release()
        cv2.destroyAllWindows()
        window.destroy()
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if (conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values
                ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
                ID = str(ID)
                ID = ID[1:-1]
                bb = str(aa)
                bb = bb[2:-2]
                attendance = [str(ID), '', bb, '', str(date), '', str(timeStamp)]

            else:
                Id = 'Unknown'
                bb = str(Id)
            cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
        cv2.imshow('Taking Attendance', im)
        if (cv2.waitKey(1) == ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
    exists = os.path.isfile("Attendance\Attendance_" + date + ".csv")
    if exists:
        with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(attendance)
        csvFile1.close()
    else:
        with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(col_names)
            writer.writerow(attendance)
        csvFile1.close()
    with open("Attendance\Attendance_" + date + ".csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for lines in reader1:
            i = i + 1
            if (i > 1):
                if (i % 2 != 0):
                    iidd = str(lines[0]) + '    '
                    tv.insert('', 0, text=iidd, values=(str(lines[2]),
str(lines[4]), str(lines[6])))
    csvFile1.close()
    cam.release()
    cv2.destroyAllWindows()

######################################## USED STUFFS
############################################

global key
key = ''

ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime(' %d-%m-%Y')
day,month,year=date.split("-")

mont={'01':'January',
      '02':'February',
      '03':'March',
      '04':'April',
      '05':'May',
      '06':'June',
      '07':'July',
      '08':'August',
      '09':'September',
      '10':'October',
      '11':'November',
      '12':'December'
      }

######################################## GUI FRONT-END
###########################################

window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
window.title("Attendance System")
window.configure(background='#262523')

frame1 = tk.Frame(window, bg="#00aeff")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#00aeff")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="Face Recognition Based Attendance System"
,fg="white",bg="#262523" ,width=55 ,height=1,font=('times', 29, ' bold '))
message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="#c4c6ce")
frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="#c4c6ce")
frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+"  |  ",
fg="orange",bg="#262523" ,width=55 ,height=1,font=('times', 22, ' bold '))
datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="orange",bg="#262523" ,width=55 ,height=1,font=('times',
22, ' bold '))
clock.pack(fill='both',expand=1)
tick()
```

```
head2 = tk.Label(frame2, text="                       For New
Registrations                       ", fg="black",bg="#3ece48" ,font=('times', 17, '
bold ') )
head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text="                       For Already
Registered                       ", fg="black",bg="#3ece48" ,font=('times', 17, '
bold ') )
head1.place(x=0,y=0)

lbl = tk.Label(frame2, text="Enter
ID",width=20  ,height=1  ,fg="black"  ,bg="#00aeff" ,font=('times', 17, ' bold ') )
lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('times', 15, ' bold '))
txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter Name",width=20  ,fg="black"  ,bg="#00aeff"
,font=('times', 17, ' bold '))
lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('times', 15, ' bold ')  )
txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images  >>>  2)Save Profile" ,bg="#00aeff"
,fg="black"  ,width=39 ,height=1, activebackground = "yellow" ,font=('times', 15, '
bold '))
message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#00aeff" ,fg="black"  ,width=39,height=1,
activebackground = "yellow" ,font=('times', 16, ' bold '))
message.place(x=7, y=450)

lbl3 = tk.Label(frame1,
text="Attendance",width=20  ,fg="black"  ,bg="#00aeff"  ,height=1 ,font=('times',
17, ' bold '))
lbl3.place(x=100, y=115)

res=0
exists = os.path.isfile("StudentDetails\StudentDetails.csv")
if exists:
    with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            res = res + 1
    res = (res // 2) - 1
    csvFile1.close()
else:
    res = 0
message.configure(text='Total Registrations till now  : '+str(res))

##################### MENUBAR #################################

menubar = tk.Menu(window,relief='ridge')
filemenu = tk.Menu(menubar,tearoff=0)
filemenu.add_command(label='Change Password', command = change_pass)
filemenu.add_command(label='Contact Us', command = contact)
filemenu.add_command(label='Exit',command = window.destroy)
menubar.add_cascade(label='Help',font=('times', 29, ' bold '),menu=filemenu)

################## TREEVIEW ATTENDANCE TABLE ####################

tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))
tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)
tv.heading('#0',text ='ID')
tv.heading('name',text ='NAME')
tv.heading('date',text ='DATE')
tv.heading('time',text ='TIME')

###################### SCROLLBAR ################################

scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)
scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')
tv.configure(yscrollcommand=scroll.set)

###################### BUTTONS ##################################

clearButton = tk.Button(frame2, text="Clear",
command=clear  ,fg="black"  ,bg="#ea2a2a"  ,width=11 ,activebackground = "white"
,font=('times', 11, ' bold '))
clearButton.place(x=335, y=86)
clearButton2 = tk.Button(frame2, text="Clear",
command=clear2  ,fg="black"  ,bg="#ea2a2a"  ,width=11 , activebackground = "white"
,font=('times', 11, ' bold '))
clearButton2.place(x=335, y=172)
takeImg = tk.Button(frame2, text="Take Images",
command=TakeImages  ,fg="white"  ,bg="blue"  ,width=34  ,height=1, activebackground
= "white" ,font=('times', 15, ' bold '))
takeImg.place(x=30, y=300)
trainImg = tk.Button(frame2, text="Save Profile", command=psw
,fg="white"  ,bg="blue"  ,width=34  ,height=1, activebackground = "white"
,font=('times', 15, ' bold '))
```

```
trainImg.place(x=30, y=380)
trackImg = tk.Button(frame1, text="Take Attendance",
command=TrackImages  ,fg="black"  ,bg="yellow"  ,width=35  ,height=1,
activebackground = "white" ,font=('times', 15, ' bold '))
trackImg.place(x=30,y=50)
quitWindow = tk.Button(frame1, text="Quit",
command=window.destroy  ,fg="black"  ,bg="red"  ,width=35 ,height=1,
activebackground = "white" ,font=('times', 15, ' bold '))
quitWindow.place(x=30, y=450)

##################### END ######################################

window.configure(menu=menubar)
window.mainloop()
```

README.md

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

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

TECHNOLOGY USED:
1) tkinter for whole GUI
2) OpenCV for taking images and face recognition
(cv2.face.LBPHFaceRecognizer_create())
3) CSV, Numpy, Pandas, datetime etc. for other purposes.

FEATURES:
1) Easy to use with interactive GUI support.
2) Password protection for new person registration.
3) Creates/Updates CSV file for deatils of students on registration.
4) Creates a new CSV file everyday for attendance and marks attendance with proper
date and time.
5) Displays live attendance updates for the day on the main screen in tabular format
with Id, name, date and time.
```

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

So the project completes here. There are many changes and improvements to be done which will be worked on.

THANK YOU

DESHIREDDY KRISHNAREDDY

20MIS1069

APPIREDDY GOWTHAM REDDY

20MIS1175

References

https://www.freecodecamp.org/news/facial-recognition-using-opencv-in-python-92fa40c22f62/

https://link.springer.com/chapter/10.1007/978-3-030-69921-5_47

OpenCV - Face Detection in a Picture (tutorialspoint.com)