Course Project Report

on

SMART ATTENDANCE MANAGEMENT SYSTEM USING FACIAL RECOGNITION

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (IOT)

Team members	Roll no
M. Gowtham	22E51A6942
M. SaiDatta	22E51A6940
A.V. Rohith Reddy	22E51A6905
L. Sivananda	22E51A6939

Department of ET, HITAM

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Autonomous, Approved by AICTE, Accredited by NAAC, NBA.

Gowdavelly (Village), Medchal (Mandal), Medchal - Malkajgiri (Dist.), Hyderabad, TS- 501401.

2023–2024

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC, NBA – TS 501401**)**

DEPARTMENT OF COMPUTER SCIENCE [IOT]

CERTIFICATE

This is to certify that the Course Level Project work entitled bearing **"SMART ATTENDANCE MANAGEMENT SYSTEM USING FACIAL RECOGNITION"** is being submitted by **M.Gowtham , M.SaiDatta , A.V.Rohith Reddy , L.Sivananda** bearing Roll No. **22E51A6942, 22E51A6940, 22E51A6905, 22E51A6939** in partial fulfilment of the academic requirement, at Hyderabad Institute of Technology and Management, Hyderabad is a record ofbonafied work carried out by them under our guidance. The matter contained in this document has not been submitted to any other University or institute.

Internal Guide

Mr. S. Periasamy

Associate professor

Department of ET, HITAM

Head of Department

Dr. M.V.A. Naidu

Professor & Head

Department of ET, HITAM

DECLARATION

We here by declare that the internship project entitled **"SMART ATTENDANCE MANAGEMENT SYSTEM USING FACIAL RECOGNITION"** submitted to **Hyderabad Institute of Technology and Management affiliated to Jawaharlal Nehru TechnologicalUniversity**, Hyderabad (JNTUH) as part of academic requirement in Coarse level project.

Team members	**Roll no**
M.Gowtham	**22E51A6942**
M.SaiDatta	**22E51A6940**
A.V.Rohith Reddy	**22E51A6905**
L.Sivananda	**22E51A6939**

ABSTRACT

To maintain the attendance record with day to day activities is a challenging task. The conventional method of calling name of each student is time consuming and there is always a chance of proxy attendance. Face is the crucial part of the human body that uniquely identifies a person. Using the face characteristics as biometric, the face recognition system can be implemented. The most demanding task in any organization is attendance marking. In traditional attendance system, the students are called out by the teachers and their presence or absence is marked accordingly. However, these traditional techniques are time consuming and tedious.

In this project, the Open CV based face recognition approach has been proposed. This model integrates a camera that captures an input image, an algorithm for detecting face from an input image, encoding and identifying the face, marking the attendance in a spreadsheet and converting it into PDF file. The training database is created by training the system with the faces of the authorized students. The cropped images are then stored as a database with respective labels. The features are extracted using LBPH (Local Binary Pattern Histogram) Algorithm.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

1.1 PROJECT SCOPE

The project scope of an "Attendance Management System using Facial Recognition" would involve the development of a software application that utilizes facial recognition technology, a database of enrolled individuals, and a user interface to manage attendance records and generate reports. The project could also involve hardware integration to support the system in various settings.

1.2 PROJECT PURPOSE

The purpose of an "Attendance Management System using Facial Recognition" is to automate the process of taking attendance in various settings, such as schools, universities, and businesses. The system would use facial recognition technology to identify individuals and record their attendance in real-time. The project aims to improve the efficiency and accuracy of attendance-taking, while providing additional benefits such as security and access control.

1.3 PROJECT FEATURES

A smart attendance management system utilizing facial recognition technology offers several advanced features that enhance the efficiency and accuracy of attendance tracking in various settings. The system employs advanced facial recognition algorithms that can accurately identify individuals in real-time, eliminating the need formanual attendance marking and reducing the chances of errors or attendance fraud. Thesystem can also handle large volumes of data, making it suitable for organizations of allsizes.

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS

2.1 Hardware Requirements:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor: Intel Core i5 10th Gen
- Hard disk: 1 TB HDD

RAM : 8GB

2.2 Software Requirements:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 11
- Languages : Python 3.11
- Tools : Python IDLE 3.11 version , VS Code
- Libraries :- Open CV contrib 4.0.1 Pillow

Numpy Pandas CSV
PysimpleGUI

- Face Recognition Algorithms :- LBPH (Local Binary Pattern Histogram)

CHAPTER 3

DESIGN AND METHODOLOGY

Software Design:

'Mark Attendance' button:

Teacher/Administrator starts the process by pressing the 'Mark Attendance' button and entering the lecture/meeting duration.

Camera gives image/video as input to the model

Model uses face recognition algorithm to recognize the person.

If person is recognized and fulfils the required parameters then person gets the option to clock in or clock out.

Each person is clocked in or out based on their choice and a record is kept of total time for which they were present.

When Teacher presses the 'Save Attendance' button, attendance report is generated and saved.

'Add Person' button:

Teacher/Administrator starts the process by pressing the 'Add Person' button.

Teacher/Administrator enters the ID and Name for the new person. Camera captures 100 pictures of the person in real time.

'Train Images' button:

Teacher/Administrator starts the process by pressing the 'Train Images' button.

Training of the model is done with the new database.

'View Attendance' button:

Teacher/Administrator starts the process by pressing the 'View Attendance' button.

Teacher/Administrator is taken to the directory where all the attendancereports are saved

Teacher/Administrator can view attendance report by clicking ‘Open’
button.

Teacher/Administrator can get back to home by clicking ‘Back’ button.

‘Quit’ button:

Teacher/Administrator starts the process by pressing the ‘Quit’ button.

Window is closed.

Software Architecture:

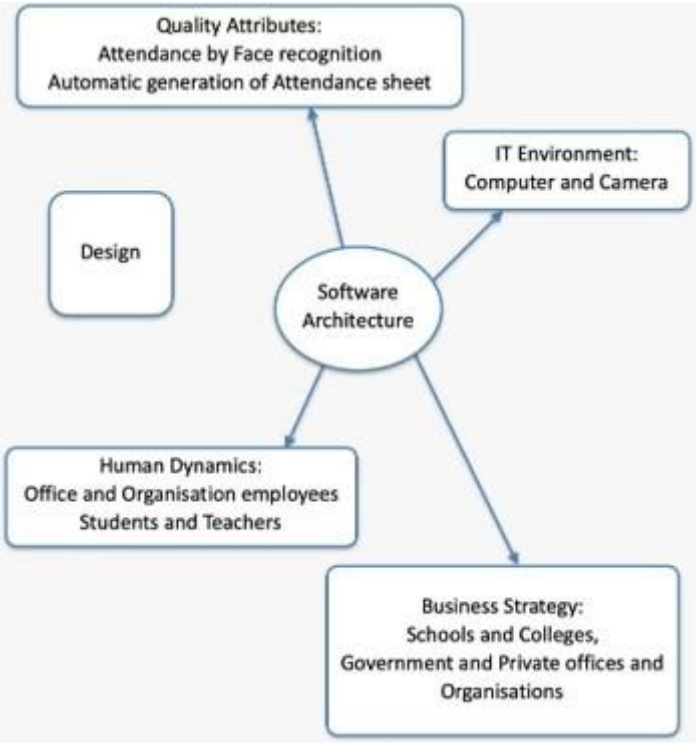

Figure 3.1.1

Automatically attendance system based on face recognized and alsodecreases manual work.

Automatic Attendance System Using Face Recognition for lecturers or staffs, implemented the attendance system. In this system, they use the algorithm of face recognition and attendance marking are developed andused.

This system captures two images of a student at two pints in time using adigital camera; one is from the start time of the class and other one is in the end time of the class. Both images will process by this system and will make important role to recognize student using facial recognition. If the student recognized both in the start time and end time classes attendancewill be marked for that student.

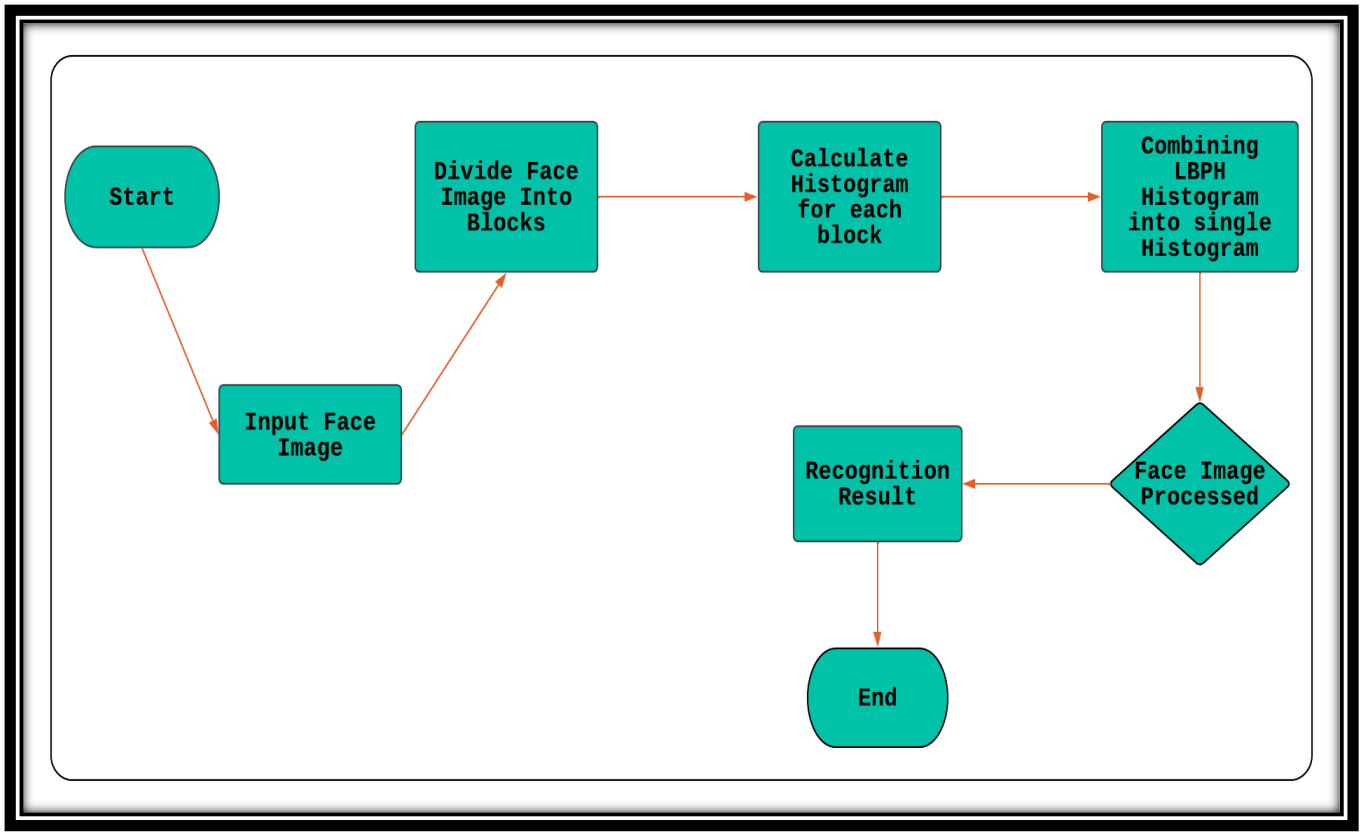

Figure 3.1.2

SAMPLE OUTPUT:

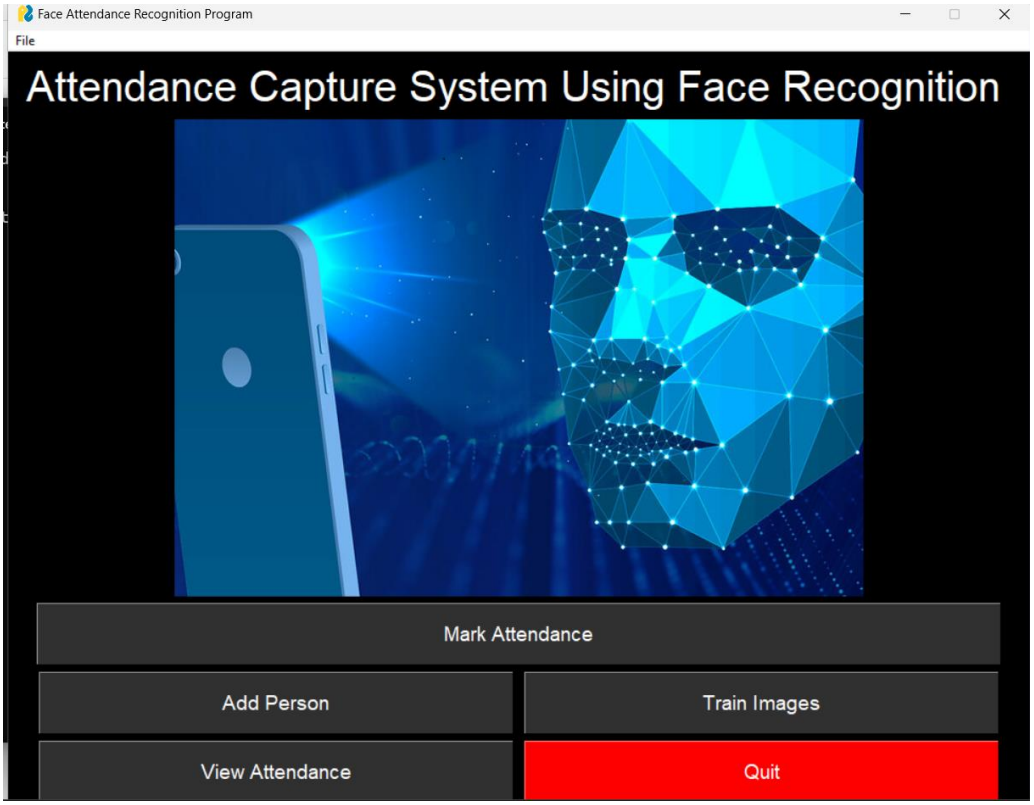

Figure 3.2.1

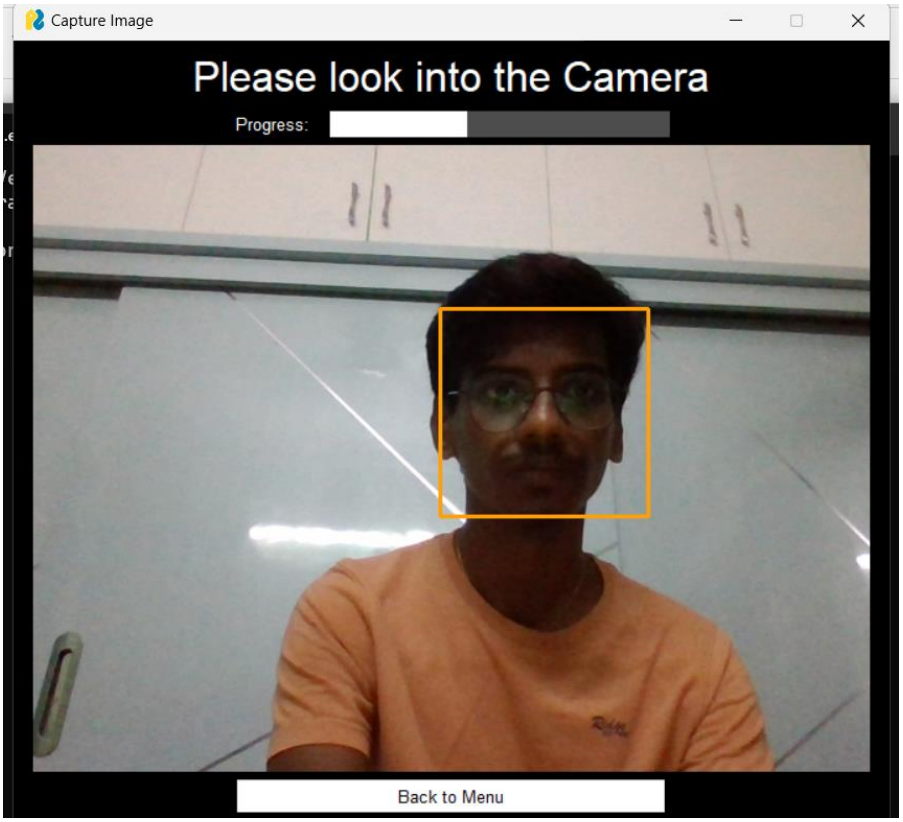

Figure 3.2.2

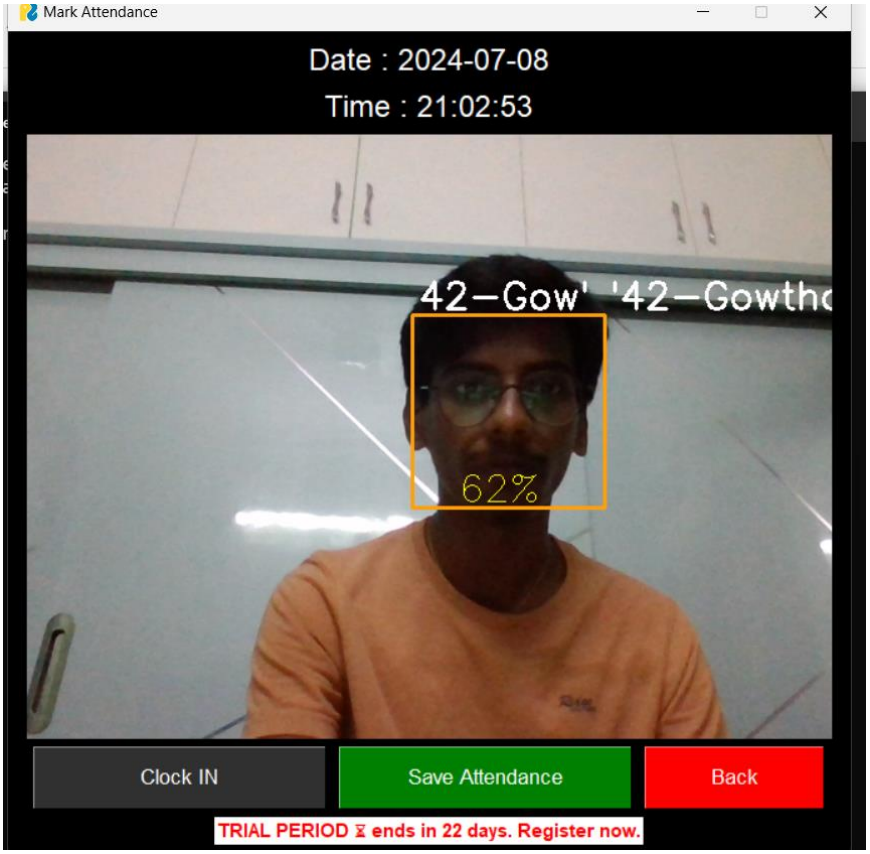

Figure 3.2.3

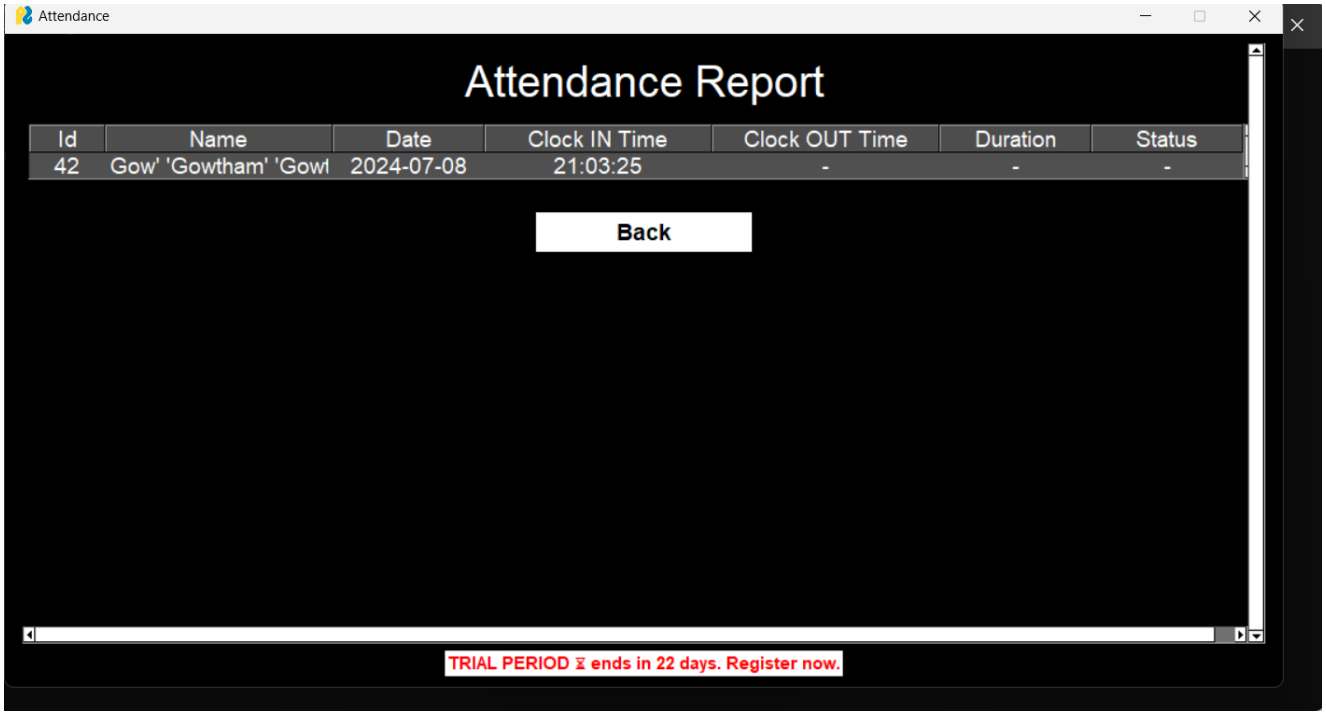

Figure 3.2.4

3.3 WORKING

1. have their faces captured and stored in the database along with their details.
2. *Attendance Marking:*
   - The camera captures video.
   - Face detection identifies faces in each frame.
   - For each detected face:
     - Face recognition compares it to the database.
     - If a match is found, the system marks the individual's attendance for the current time.
     - If no match is found, the system might prompt for manual intervention or handle it as an absence depending on configuration.
3. *Data Storage:*
   - Attendance data is typically stored in a database with timestamps and individual identification.
   - It can include additional details like location (classroom, office) and class/meeting ID (if applicable for multiple attendances within a period).

Multiple Attendances in a Class:

- The system can be configured to handle multiple attendance markings within a class period. This might involve:
  - *Time-based windows:* Attendance is marked only within specific time slots (e.g., beginning and end of class).
  - *Unique identifiers:* A separate class/meeting ID is used for each attendance point, allowing for tracking participation throughout the session.

3.4 CODE

```
import os
import Capture_Image
import Train_Image
import Recognize
import view_attendance
import PySimpleGUI as sg

def mainMenu():
    menu_def = [['&File', ['&Open Attendance Folder', '&Open Student Records','---', 'E&xit', ]]]
    sg.theme('Black')

    layout = [[sg.Menu(menu_def, tearoff=False, pad=(200, 1))],
              [sg.Text('Attendance Capture System Using Face Recognition ', font='Helvetica 30',
justification = 'center')],
              [sg.Image(r'Images/Facial_Recognition_logo.png', size=(650,450))],
              [sg.Button("Mark Attendance", size=(82, 2), font='Helvetica 14', button_color=('white',
'#303030'))],
              [sg.Button("Add Person", size=(40, 2), font='Helvetica 14', button_color=('white',
'#303030')), sg.Button("Train Images", size=(40, 2), font='Helvetica 14', button_color=('white',
'#303030'))],
              [sg.Button("View Attendance", size=(40, 2), font='Helvetica 14', button_color=('white',
'#303030')), sg.Button("Quit", size=(40, 2), font='Helvetica 14', button_color=('white', 'red'))] ]
    window = sg.Window('Face Attendance Recognition Program', layout,auto_size_buttons=False,
element_justification='c')

    while True:
        event, values = window.read(timeout=0.1)
        if event == "Quit" or event == "Exit" or event == sg.WIN_CLOSED:
            window.close()
            break
        elif event == "Open Attendance Folder":
            path = "Attendance"
            path = os.path.realpath(path)
            os.startfile(path)
        elif event =="Open Student Records":
            path = "StudentDetails"
            path = os.path.realpath(path)
            os.startfile(path)
        elif event == "Add Person":
            window.close()
            Capture_Image.takeImages()
            mainMenu()
            break
        elif event == "Train Images":
            window.close()
            Train_Image.TrainImages()
            mainMenu()
            break
        elif event == "Mark Attendance":
```

```
            window.close()
            Recognize.recognize_attendence()
            mainMenu()
            break
        elif event == "View Attendance":
            window.close()
            view_attendance.vcsv()
            # os.system("py view_attendance.py")
            mainMenu()
            break

mainMenu()
```

#CAPTURE IMAGE

```
import csv
import PySimpleGUI as sg
import cv2
import os

# counting the numbers
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass
    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass
    return False

# Take image function
def takeImages():
    sg.theme('Black')
    layout = [[sg.Text('ID:', size =(7, 1), font='Helvetica 14'), sg.InputText('', font='Helvetica 14')],
              [sg.Text('Name:', size =(7, 1), font='Helvetica 14'), sg.InputText('', font='Helvetica 14')],
              [sg.Button('Submit', button_color=('white', '#303030'), font='Helvetica 14', size=(20,1)),
              sg.Button('Cancel', button_color=('white', '#303030'), font='Helvetica 14', size=(20,1))]]
    window = sg.Window('Student Details', layout, element_justification='c')
    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == 'Cancel':  # if user closes window or clicks cancel
            window.close()
            break
        elif event == 'Submit':
            Id = values[0]
```

```
        name = values[1]
        window.close()
    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
        detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        sampleNum = 0
        layout = [ [sg.Text("Please look into the Camera",font='Helvetica 24')],
                [sg.Text("Progress: "),sg.ProgressBar(101, orientation='h', size=(20, 20),
key='progressbar')],
                [sg.Image(filename='', key='image')],[sg.Button("Back to Menu",size=(40,1))] ]
        window = sg.Window('Capture Image', layout, auto_size_buttons=False,
element_justification='c', location=(350, 75))
        progress_bar = window['progressbar']
        while(True):
            event, values = window.read(timeout=1)
            if event == "Back to Menu" or event == sg.WIN_CLOSED:
                cam.release()
                cv2.destroyAllWindows()
                window.close()
                break
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5, minSize=(30,30),flags =
cv2.CASCADE_SCALE_IMAGE)
            for(x,y,w,h) in faces:
                cv2.rectangle(img, (x, y), (x+w, y+h), (10, 159, 255), 2)
                #incrementing sample number
                sampleNum = sampleNum+1
                progress_bar.UpdateBar(int(sampleNum))
                #saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage" + os.sep +name + "."+Id + '.' +
                        str(sampleNum) + ".jpg", gray[y:y+h, x:x+w])
                imgbytes = cv2.imencode(".png", img)[1].tobytes()
                window["image"].update(data=imgbytes)
            #wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is more than 100
            elif sampleNum > 100:
                break
        window.close()
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id + " Name : " + name
        row = [Id, name]
        with open("StudentDetails"+os.sep+"StudentDetails.csv", 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
    else:
        takeImages()
```

TRAIN IMAGES

```
import os
import cv2
import numpy as np
from PIL import Image
import PySimpleGUI as sg

# image labesl
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)]
    value = len(imagePaths)
    # create empth face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the images
    i = 1
    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)
        sg.one_line_progress_meter('Image Training Model', i, value, 'key', 'Training Time Left: ',orientation="h")
        i+=1
    return faces, Ids

# train images function
def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces, Id = getImagesAndLabels("TrainingImage")
    target = recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImageLabel"+os.sep+"Trainner.yml")
    sg.popup_auto_close('All Images Trained')
```

#Recognize:

```
import datetime
import os
import time
import PySimpleGUI as sg
import cv2
import pandas as pd


def recognize_attendence():
    sg.theme('Black')
    layout = [  [sg.Image(filename='', key='image'),
                [sg.Text(f'Date : {datetime.datetime.fromtimestamp(time.time()).strftime("%Y-%m-%d")}', key='_date_', font=('Helvetica 18'))],
                [sg.Text(f'Time : {datetime.datetime.fromtimestamp(time.time()).strftime("%H:%M:%S")}', key='_time_', font=('Helvetica 18'))]],
                [sg.Button("Clock IN",size=(25,2), font=('Helvetica 13'), button_color=('white', '#303030')),sg.Button("Clock OUT",size=(25,2), font=('Helvetica 13'), button_color=('white', '#303030')), sg.Button("Save Attendance",size=(25,2), font=('Helvetica 13'), button_color=('white', 'green')), sg.Button("Back",size=(15,2), font=('Helvetica 13'), button_color=('white', 'red')) ] ]
    window = sg.Window('Mark Attendance', layout, auto_size_buttons=False, element_justification='c', location=(350, 75))
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read("TrainingImageLabel"+os.sep+"Trainner.yml")
    faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    df = pd.read_csv("StudentDetails"+os.sep+"StudentDetails.csv")
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', 'Name', 'Date', 'Clock IN Time', 'Clock OUT Time', 'Duration', 'Status']
    attendance = pd.DataFrame(columns=col_names)
    # Initialize and start realtime video capture
    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cam.set(3, 640)  # set video width
    cam.set(4, 480)  # set video height
    # Define min window size to be recognized as a face
    minW = 0.1 * cam.get(3)
    minH = 0.1 * cam.get(4)


    lecture = sg.popup_get_text('Please Enter Lecture Duration', 'HH:MM:SS')


    while True:
        event, values = window.read(timeout=1)
        window.find_element('_time_').update(f'Date : {datetime.datetime.fromtimestamp(time.time()).strftime("%Y-%m-%d")}')
        window.find_element('_time_').update(f'Time : {datetime.datetime.fromtimestamp(time.time()).strftime("%H:%M:%S")}')
```

```
if event == 'Back':
    c = sg.PopupYesNo(f'Save Attendance ?')
    if c == 'N0':
        cam.release()
        cv2.destroyAllWindows()
        window.close()
    elif c == 'Yes':
        ts = time.time()
        date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
        timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
        Hour, Minute, Second = timeStamp.split(":")
        fileName = "Attendance"+os.sep+"Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
        attendance.to_csv(fileName, index=False)
        cam.release()
        cv2.destroyAllWindows()
        window.close()
        sg.popup_timed('Attendance Successful')
    break
elif event == "Save Attendance" or event == sg.WIN_CLOSED:
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour, Minute, Second = timeStamp.split(":")
    fileName = "Attendance"+os.sep+"Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName, index=False)
    cam.release()
    cv2.destroyAllWindows()
    window.close()
    sg.popup_timed('Attendance Successful')
    break
elif event == 'Clock IN':
    check = sg.PopupYesNo(f'{aa[0]} are you clocking In?')
    if check == 'Yes':
        ts = time.time()
        date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
        timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
        aa = str(aa)[2:-2]
        attendance.loc[len(attendance)] = [Id, aa, date, timeStamp, '-', '-', '-']
    elif check == 'N0':
        print('Not clocked IN')
elif event == 'Clock OUT':
    check = sg.PopupYesNo(f'{aa[0]} are you clocking OUT?')
    if check == 'Yes':
```

```
            ts = time.time()
            timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
            attendance.at[attendance[attendance['Id'] == Id].index.values, 'Clock OUT Time'] = timeStamp
            co = attendance.loc[attendance[attendance['Id'] == Id].index.values, 'Clock OUT Time'].to_string().split(' ')
            ci = attendance.loc[attendance[attendance['Id'] == Id].index.values, 'Clock IN Time'].to_string().split(' ')
            FMT = '%H:%M:%S'
            duration = datetime.datetime.strptime(co[-1], FMT) - datetime.datetime.strptime(ci[-1], FMT)
            attendance.at[attendance[attendance['Id'] == Id].index.values, 'Duration'] = duration
            d = datetime.datetime.strptime(lecture, FMT) - datetime.datetime.strptime(str(duration), FMT)
            if int(str(d).split(':')[1]) in range(-5, 6):
                attendance.at[attendance[attendance['Id'] == Id].index.values, 'Status'] = 'Present'
            else:
                attendance.at[attendance[attendance['Id'] == Id].index.values, 'Status'] = 'MCR'
        elif check == 'No':
            print('Not clocked OUT')


    ret, im = cam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, 1.2, 5,minSize = (int(minW), int(minH)),flags =
cv2.CASCADE_SCALE_IMAGE)
    for(x, y, w, h) in faces:
        cv2.rectangle(im, (x, y), (x+w, y+h), (10, 159, 255), 2)
        Id, conf = recognizer.predict(gray[y:y+h, x:x+w])
        if conf < 100:
            aa = df.loc[df['Id'] == Id]['Name'].values
            confstr = "  {0}%".format(round(100 - conf))
            tt = str(Id)+"-"+aa
        else:
            Id = '  Unknown  '
            tt = str(Id)
            confstr = "  {0}%".format(round(100 - conf))
        tt = str(tt)[2:-2]
        if(100-conf) > 67:
            tt = tt + " [Pass]"
            cv2.putText(im, str(tt), (x+5,y-5), font, 1, (255, 255, 255), 2)
        else:
            cv2.putText(im, str(tt), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
        if (100-conf) > 67:
            cv2.putText(im, str(confstr), (x + 5, y + h - 5), font,1, (0, 255, 0),1 )
        elif (100-conf) > 50:
            cv2.putText(im, str(confstr), (x + 5, y + h - 5), font, 1, (0, 255, 255), 1)
        else:
            cv2.putText(im, str(confstr), (x + 5, y + h - 5), font, 1, (0, 0, 255), 1)
```

```
        attendance = attendance.drop_duplicates(subset=['Id'], keep='first')
        imgbytes = cv2.imencode(".png", im)[1].tobytes()
        window["image"].update(data=imgbytes)

    cam.release()
    cv2.destroyAllWindows()
    os.system('cls')
```

 # VIEW ATTENDANCE

```
import PySimpleGUI as sg

import csv


sg.theme('Black')

def vcsv():

    filename = sg.PopupGetFile('Get required file', no_window = True,file_types=(("CSV Files","*.csv"),))

    data = []

    #read csv

    with open(filename, "r") as infile:

        reader = csv.reader(infile)

        for i in range (1):

            #get headings

            header = next(reader)

            #read everything else into a list of rows

            data = list(reader)

    col_layout = [[sg.Text('Attendance Report', font='Helvetica 28', justification='center', pad=(0,10))], [sg.Table(values=data, headings=header,col_widths = (5, 15, 10, 15, 15, 10, 10), auto_size_columns=False,

                    max_col_width = 30, size=(None, len(data)), font='Helvetica 14', justification = 'center', background_color='#303030', text_color='white', alternating_row_color='#505050')],

                    [sg.ReadButton('Back', font = ('Arial', 14, 'bold'), size = (15,1), pad=(0,25))]]


    layout = [[sg.Column(col_layout, size=(1050,500), scrollable=True, element_justification='center')]]
```

```
    window = sg.Window('Attendance',layout, grab_anywhere = False, element_justification='c',
location=(200, 150))

    event, values = window.Read()

    while True:

        if event == 'Back' or event == sg.WIN_CLOSED:

            window.close()

            break
```

3.5 CONCLUSION

Attendance is one of the most important aspect for all the organizations, schools and offices. Our project makes attendance capturing easy, efficient and faster. Our project also makes the attendance system secure since attendance is marked using face recognition algorithm and all the users will be allowed specific ID. The project also allows adding a new user by taking on the spot 100 photographs of the new user, allowing a new user ID with the user's credentials as given by the user, and training of the new model for better prediction. The project automatically keeps track of time and marks attendance with respect it hence it keeps record of the total time for which each employee/student/user was present. This project generates an attendance report and automatically saves it and thus makes the process of taking attendance a lot easier and faster than manually marking attendance.

3.6. References

[1] N. K. Balcoh, M. H. Yousaf, W. Ahmad and M. I. Baig, “Algorithm for efficient attendance management: face recognition based approach,” International Journal of Computer Science Issues, vol. 9, no. 4, pp. 146-150, 2012.

[2] FBI Face Recognition. Available at: https://.fbi.gov%2Ffilerepository%_2Fabout-us-cjisfingerprints_biometrics-biometric-centerof-excellences-facerecognition.pdf. Accessed 14 January 2017.

[3] A. Sato, H. Imaoka, T. Suzuki and T. Hosoi, “Advances in face detection and recognition technologies,” NEC Journal of Advance Technology, vol. 2, no. 1, pp. 28-34, 2005.

[4] Roger (Ruo-gu) Zhang, Henry Chang, “A literature survey of face recognition and reconstruction techniques,” Dec. 2005.

[5] A. Rahim, N. Ismail, F. Razak, Z. Zulkifli, H. Jamian, F. Razi, and N. Mohammad, “Automated attendance management and alert system,” Journal of Fundamental and Applied Sciences, vol. 9, no. 65, pp. 59-80, 2017.

[6] S. C. Gaddam, N. V. Ramesh and H. Dhanekula. “Face recognition based attendance management system with raspberry Pi2 using eigen faces algorithm,” ARPN Journal of Engineering and Applied Sciences,