

ABSTRACT

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 algorithm.

INDEX

Table of contents	Page No.
1. INTRODUCTION	1-4
1.1 Software Requirements	2
1.2 Python Libraries Used	2-3
1.3 Hardware Components Requirements	3
1.4 Advantages	3
1.5 Disadvantages	4
1.6 Applications	4
2. REQUIREMENT ANALYSIS DOCUMENT	5-9
2.1 Introduction	5
2.2 Purpose of the system	5
2.3 Scope of the system	5
2.4 Objective and success criteria of the system	5
2.5 Current system	6
2.6 Proposed system	6-9
3. SYSTEM DESIGN DOCUMENT	10-13
3.1 Introduction	10
3.2 Purpose of the system	10
3.3 Design goals	10
3.4 Current System Architecture	11
3.5 Proposed Software Architecture	12-13
4. METHODOLOGY	14-35
4.1 Haar Cascade Classifier	14
4.2 Local Binary Patterns Histogram	15
4.3 System flow diagram	16
4.4 How our proposed system works?	17
4.5 CODE	18-35
5. TESTING	36-39
5.1 System testing	36-39

6. SCREENS	40-45
6.1 Front view	40
6.2 Registration of new student	41
6.3 Training images	42
6.4 Taking Attendance	43
6.5 Student's Confirmation for attendance	44
7. REPORT	45
7.1 Attendance record	45
7.2 Student Details record	45
8. CONCLUSION	46
9. REFERENCES	47

FACE RECOGNITION ATTENDANCE **SYSTEM**

1. INTRODUCTION

The objective of this project is to build a face recognition-based attendance monitoring system for educational institutions to enhance and upgrade the current attendance system into more efficient and effective as compared to before. The current old system has a lot of ambiguity that causes inaccurate and inefficient attendance taking. Many problems arise when the authority is unable to enforce the regulations that exist in the old system. The technology working behind will be the face recognition system. The human face is one of the natural traits that can uniquely identify an individual. Therefore, it is used to trace identity as the possibilities for a face to deviate or be duplicated is low. In this project, face databases will be created to pump data into the recognizer algorithm. Then, during the attendance taking session, faces will be compared against the database to seek for identity. When an individual is identified, its attendance will be taken down automatically saving necessary information into an excel sheet.

There are two phases in Face Recognition Based Attendance System

- Face Detection

Face Detection is a method of detecting faces in the images. It is the first and essential step needed for face recognition. It mainly comes under object detection like for example a car in an image or any face in an image and can use in many areas such as security, biometrics, law enforcement, entertainment, personal safety, etc.

- Face Recognition

Face Recognition is a method of identifying or verifying a person from images and videos that are captured through a camera. Its Key role is to identify people in photos, videos, or in real-time.

1.1 Software Requirements

- Operating system - Windows 10
- Anaconda
- CSV file reader (MS Excel)
- Some Python Libraries
- Camera (Web Cam)

1.2 List of Python Libraries Used

- tk-tools
This repository holds useful high-level widgets written in pure python. This library used type hints and requires Python 3.5+; it could, however, be back-ported to earlier Python versions without difficulty.
- OpenCV-contrib-python
OpenCV is a great tool for **image processing and performing computer vision tasks**. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.
- Datetime
The datetime module supplies classes for manipulating dates and times. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.
- pytest-shutil
The shutil in Python is a module that offers several functions to deal with operations on files and their collections. It provides the ability to copy and remove files. In a way, it is similar to the OS Module; however, the OS Module does have functions dealing with collections of files.

- **python-csv**
The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel.
- **Pillow**
The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
- **Pandas**
pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Times**
This module provides various time-related functions.

1.3 Hardware Components Requirements

- Processor – i3
- Hard Disk – 4 GB
- RAM – 4 GB RAM

1.4 Advantages

- The software can be used for security purposes in organizations and in secured zones.
- The software stores the faces that are detected and automatically marks attendance.
- The system is convenient and secure for the users.
- It saves their time and efforts.

1.5 Disadvantages

- The system doesn't recognize properly in poor light so may give false results.
- It can only detect face from a limited distance.
- The system can mark attendance one person at a time.

1.6 Applications

- The system can be used for places that require security like bank, military etc.
- It can also be used in houses and society to recognize the outsiders and save their identity.
- The software can be used to mark attendance based on face recognition in organizations.

2. REQUIREMENT ANALYSIS DOCUMENT

2.1 Introduction

Every organization requires a robust and stable system to record the attendance of their students and every organization have their own method to do so, some are taking attendance manually with a sheet of paper by calling their names during lecture hours and some have adopted biometrics system such as fingerprint, RFID card reader, Iris system to mark the attendance. The conventional method of calling the names of students manually is time consuming event. The RFID card system, each student assigns a card with their corresponding identity but there is a chance of card loss or unauthorized person may misuse the card for fake attendance. While in other biometrics such as fingerprint, iris or voice recognition, they all have their own flaws and also, they are not 100% accurate.

2.2 Purpose of the system

- 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.

2.3 Scope of the system

- The main objective of this project is to offer systems that simplify and automate the process of recording and tracking students' attendance through face recognition technology.
- It is biometric technology to identify or verify a person from a digital image or surveillance video.

2.4 Objective and Success Criteria of the System

- 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.

2.5 Current System

Traditional attendance marking techniques i.e., pen and paper or signing attendance sheets are easy to bypass and trick as giving proxies or false signatures is a common practice among students nowadays, students take an unfair advantage of this at most times. But a facial recognition system is unassailable and cannot be fooled as each person has a set of unique and individual features common to that person and cannot be replicated or changed, it all comes down to one simple truth that is, unless you are physically present in the lecture your attendance will not get marked.

2.6 Proposed System

In the proposed system, initially all the students will be enrolled by storing their facial images with a unique ID. Local Binary Patterns Histogram (LBPH) algorithm is used for face recognition and training the stored dataset, that generates the histogram for stored images and the real time image.

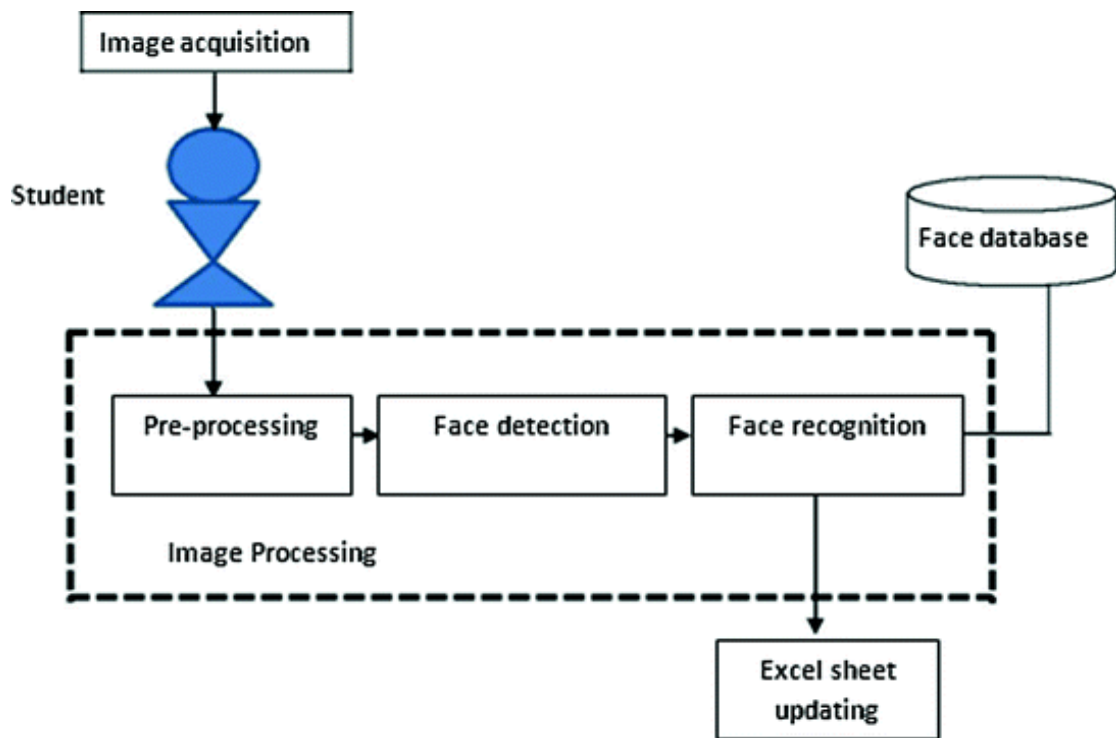
2.6.1 Functional Requirements

- Taking and tracking student attendance by facial recognition at specific times.
- Sending the names of the absent student directly to the lecturer
- Permitting the lecturer to modify the student absent or late.
- Showing the names of who is absent or late in the screen to avoid errors.

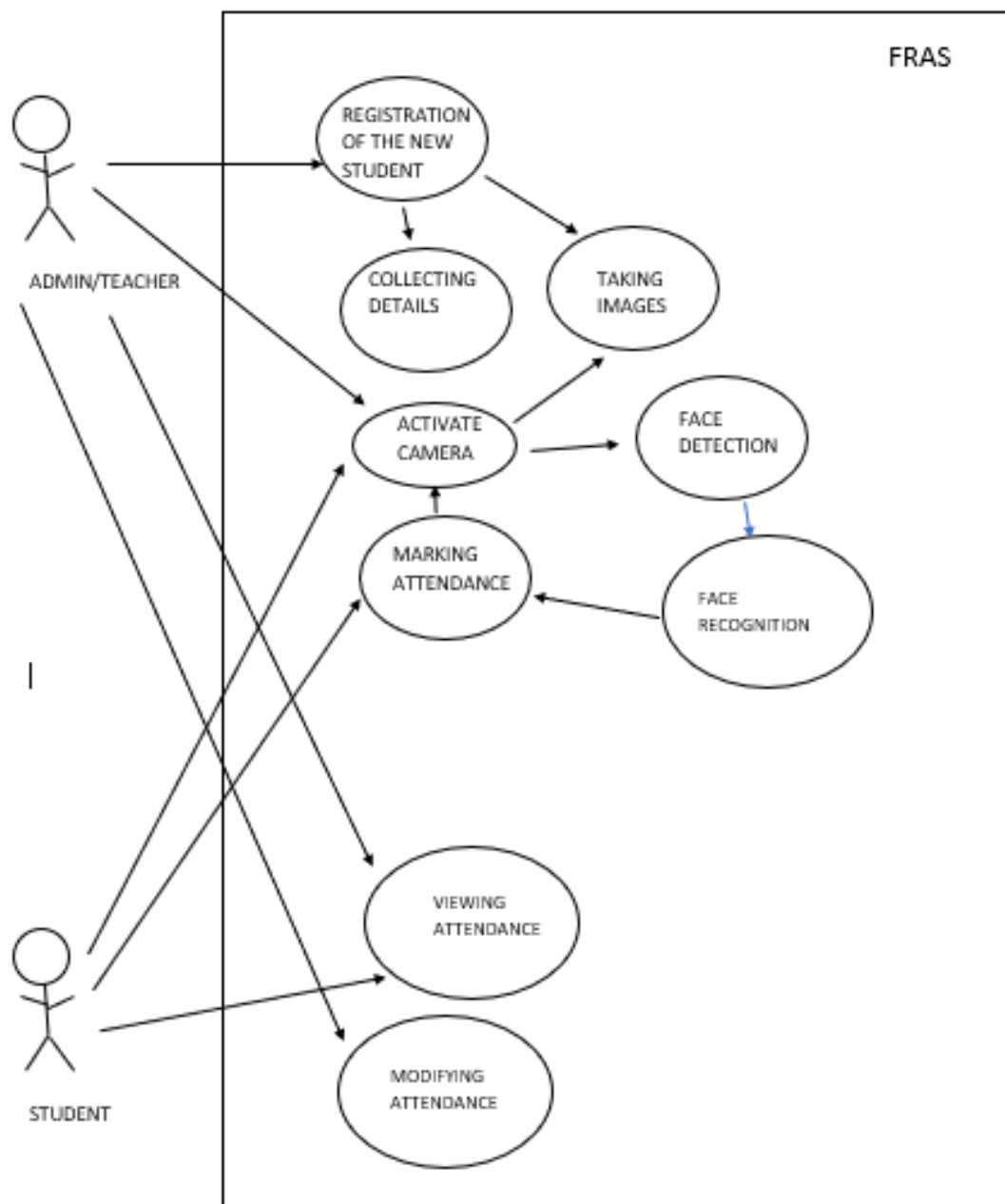
2.6.2 Non-functional requirements

- **Accuracy and Precision:** the system should perform its process in accuracy and Precision to avoid problems.
- **Modifiability:** the system should be easy to modify, any wrong should be correct.
- **Security:** the system should be secure and saving student's privacy.
- **Usability:** the system should be easy to deal with and simple to understand.
- **Maintainability:** the maintenance group should be able to fix any problem occur suddenly.
- **Speed and Responsiveness:** Execution of operations should be fast.

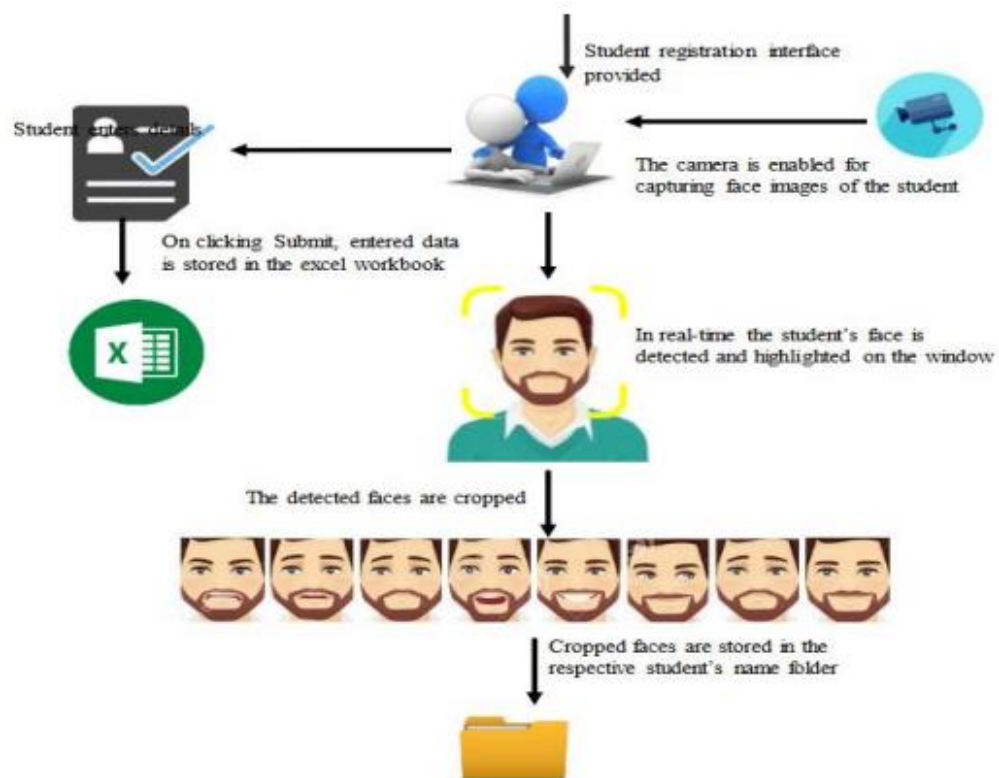
2.6.3 System Model



2.6.4 Use Case Diagram



2.6.5 Working Diagram of the System



3. System Design Document

3.1 Introduction

The system is developed for deploying an easy and a secure way of taking down attendance. The software first captures an image of all the authorized persons and stores the information into a database. The system then stores the image by mapping it into a face coordinate structure. Next time whenever the registered person enters the premises the system recognizes the person and marks his attendance along with the time.

3.2 Purpose of the system

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.

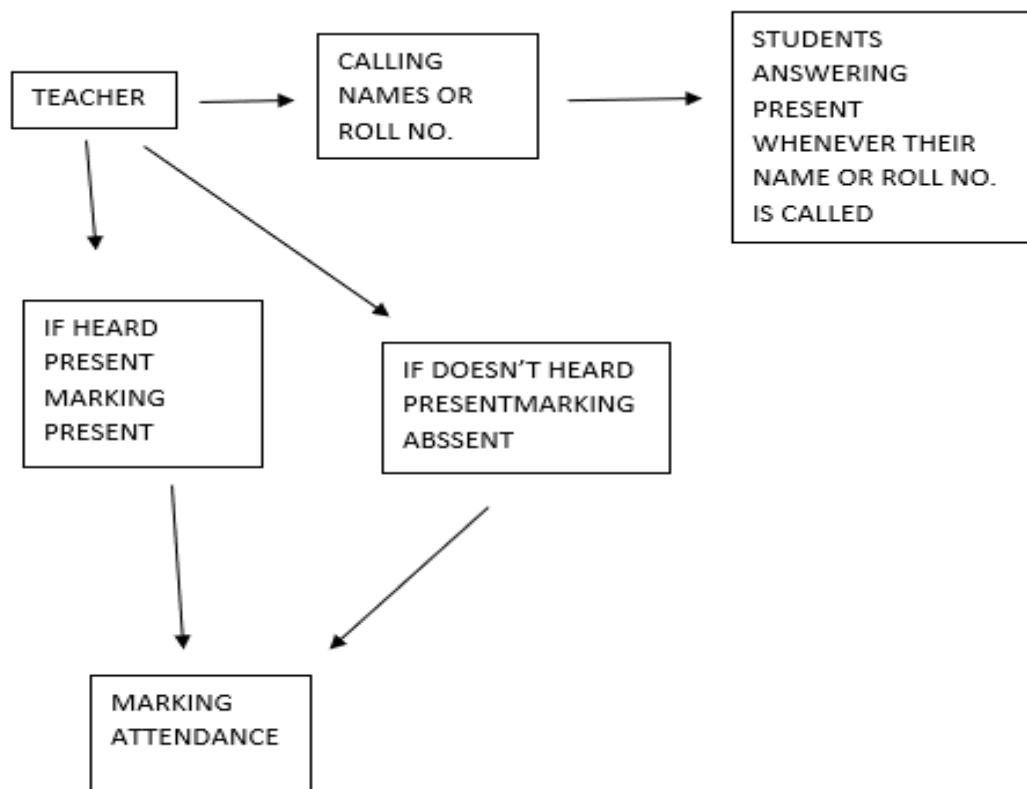
3.3 Design Goals

- **Accessibility:** the system should be accessible to the user so they can use it whenever they need it.
- **Fit for Purpose:** the system should be fit for purpose, (in this case, we need to make sure it's showing the right angles and counting)
- **Reusability:** the system should be reusable many times.
- **User Friendly:** the system must be user friendly for the user to access it easily.

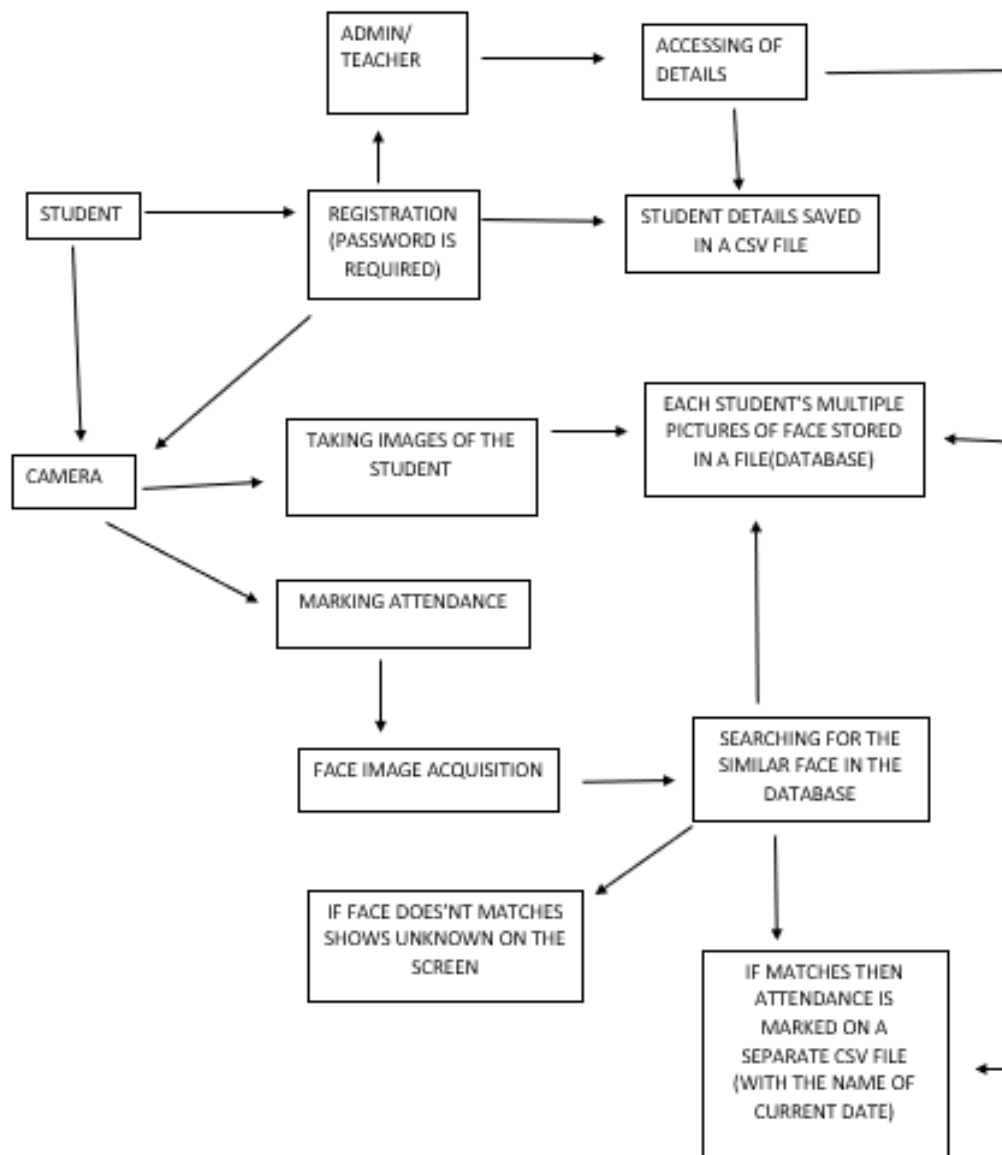
3.4 Current System Architecture

The current system of taking attendance is a pen and paper method where teachers use to call the names or roll number of the students and the students are supposed to answer present and if someone has not answered he or she will be simply marked absent. This method consumes time and often creates chaos in the class during the time of attendance.

Teacher taking attendance in a classroom



3.5 Proposed Software Architecture



3.5.1 Working of the system

- When we run the program, a window will get opened. In it we can find certain options or buttons and text fields for different purposes.
- We can register a new student in the student database by the help of the admin who has the password for registration of the new student, while registering the system ask for the student to enter **ID** and **name** and after that system takes multiple images of the student and stores in a file(database).
- A registered student can simply mark his or her attendance just by clicking on the button **take attendance**, and showing his/her face on the camera and then the system will identify the face and mark the attendance automatically in a CSV file (attendance recorded separately date wise).

3.5.2 Persistent data mapping

- The registered students data is stored in a CSV file.
- And the Attendance of the students are marked too in a CSV file.
- The attendance is marked for different dates in separate CSV files.

3.5.3 Access control and Security

- The details of the students can only be accessed by the teachers or the admin.
- For security there must be a password in the PC from which this project (System) is being executed.
- And while registration of students a password is required which only the admin knows or the teacher, and that password can be changed accordingly by the admin/teacher.
- These details can be accessed by the admin/teacher

4. METHODOLOGY

The following are the methods used in the project “FACE RECOGNITION ATTENDANCE SYSTEM”:

- Haar Cascade Classifier
- Local Binary Patterns Histogram

These two methodologies come under OpenCV. OpenCV comes with a trainer and as well as a detector. So, if you want to train your classifier for any object then you can use this classifier called Haar Cascade Classifier.

4.1 Haar Cascade Classifier

Haar Cascade classifier is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, “**Rapid Object Detection using a Boosted Cascade of Simple Features**” in 2001.

So, let’s try to understand what these Haar Cascade Classifiers are. This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

So how this works is they are huge individual .xml files with a lot of feature sets and each xml corresponds to a very specific type of use case.

For Example, if you go to the GitHub page of haarcascade you will see that there is a particular xml file containing the feature set to detect the full body, lower-body, eye, frontal-face and so on.

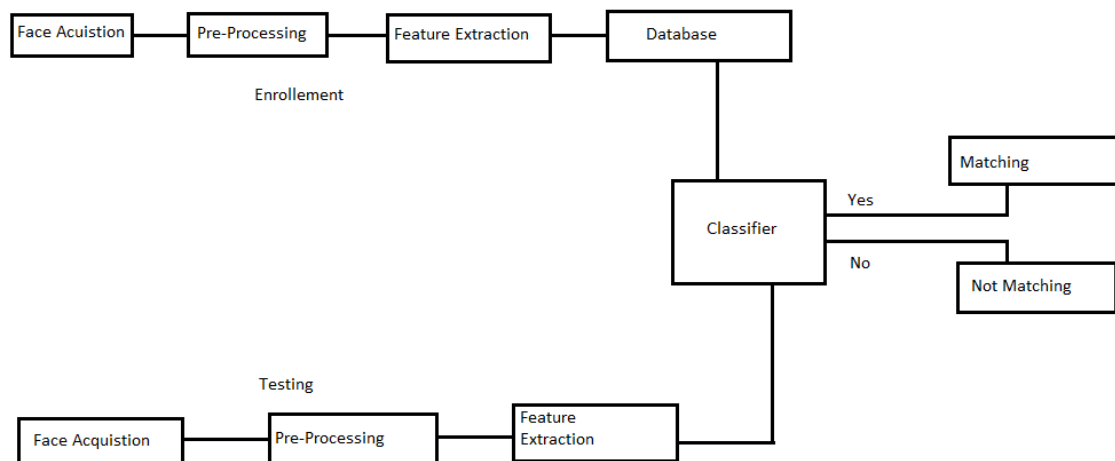
4.2 Local Binary Patterns Histogram

Local Binary Patterns Histogram algorithm (LBPH) is for face recognition. It is based on local binary operator, and it is one of the best performing textures descriptors. The need for facial recognition systems increasing day by day as per today's busy schedule. They are being used in entrance control, surveillance systems, smartphone unlocking etc. In this article, we will use LBPH to extract features from an input test image and match them with the faces in system's database.

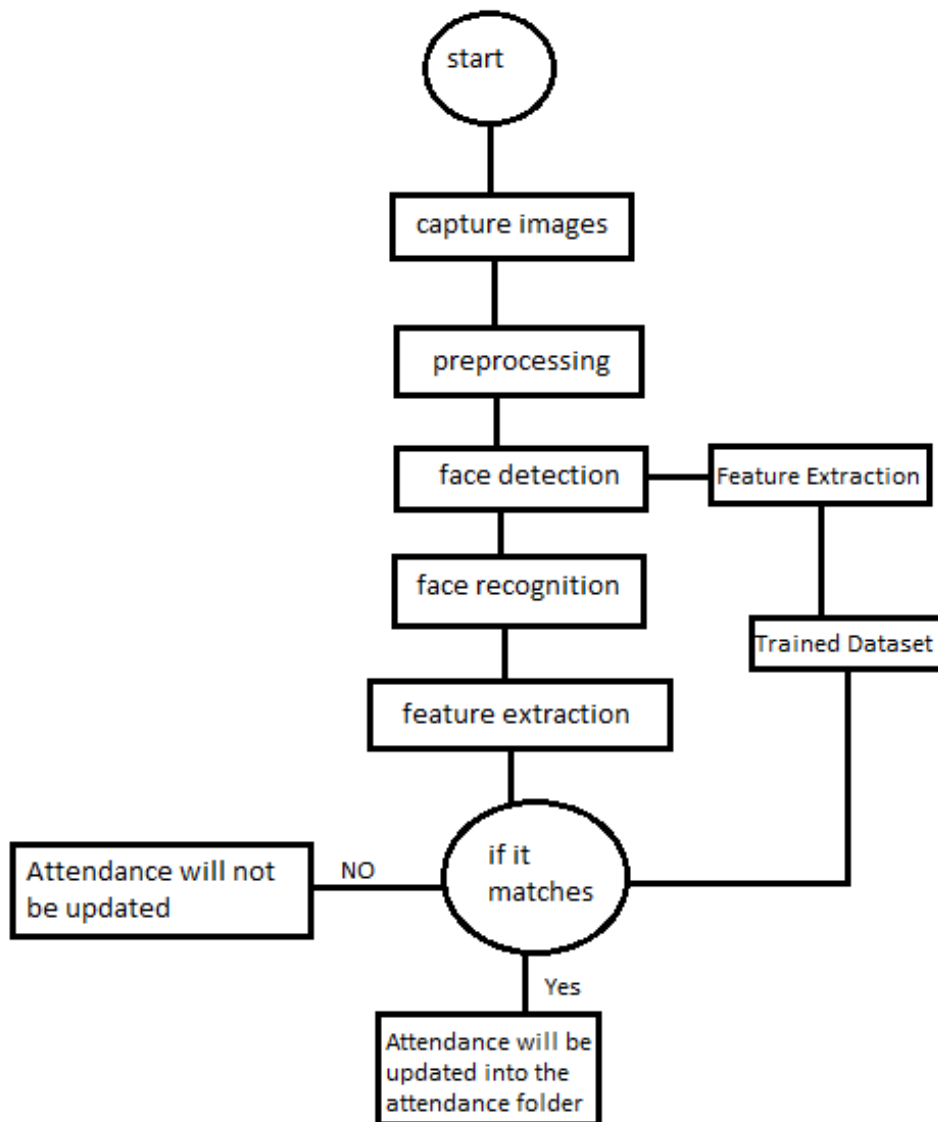
Local Binary Patterns Histogram algorithm was proposed in 2006. It is based on local binary operator. It is widely used in facial recognition due to its computational simplicity and discriminating power. The steps involved to achieve this are:

- creating datasets
- face acquisition
- feature extraction
- classification

Steps involved in LBPH:



4.3 System Flow Diagram



Step 1: First of all, it captures the input image

Step 2: After capturing the image it will pre-process the image and converts the image into gray scale Image.

Step 3: By using Haar Cascade Classifier face detection will be done and extracts features from the image and then stored in trained set database.

Step 4: Similarly face recognition is done by using Local Binary Patterns Histogram.

Step 5: And then extracted features will be compared with the trained data set.

Step 6: If it matches, attendance will be updated in the attendance folder.

Step 7: If not matches attendance will not be updated in the attendance folder.

4.4 How our Proposed System works?

- When you run the program, a window is opened in which we can see many various buttons with different functionalities.
- There you can find two sections one for registration of a new student and another one for marking attendance.
- For registration of a student, the student is supposed to enter his or her ID and NAME, after that he or she have to be stand in front of the camera so that the system can take his or her image.
- After this a message will be displayed that images taken successfully.
- After this all, the system asks for a password for saving the profile of the registered student in the database, only the admin has the password so therefore an admin is mandatory for the purpose of registration of a student.
- After the registration the details of the students who had registered, their information will be saved in a CSV file.
- And suppose a student who had already registered himself or herself wants to give attendance then he or she simply clicks on to take attendance button and stands in front of the camera's frame and waits for the system to recognize the student.
- If the student successfully recognises the student, then the system displays the name of the student below their face and then the student is supposed to press 'q' to quit his and her attendance.
- The list of students who have already given their attendance will also be displayed in the system's window.
- And the date wise attendance record will also be saved in an CSV file in a folder named Attendance, where different CSV files will be created automatically with name of the Attendance underscore and followed by the date of the that particular day.

4.5 CODE

IMPORTING

```
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 :  
'balakrishnambk7@gmail.com' ")
```

```
def check_haarcascade():  
    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=('comic', 10, ' bold
'))

lbl4.place(x=10,y=10)

global old

old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('comic', 10, ' bold
'),show='*')

old.place(x=180,y=10)

lbl5 = tk.Label(master, text='Enter New Password', bg='white', font=('comic', 10, ' bold
'))

lbl5.place(x=10, y=45)

global new

new = tk.Entry(master, width=25, fg="black",relief='solid', font=('comic', 10, ' bold
'),show='*')

new.place(x=180, y=45)

lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('comic', 10, '
bold '))

lbl6.place(x=10, y=80)

global nnew

nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('comic', 10, ' 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=('comic',
10, ' bold '))

cancel.place(x=200, y=120)

save1 = tk.Button(master, text="Save", command=save_pass, fg="black",
bg="#00ffcc", height = 1,width=25, activebackground="white", font=('comic', 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_haarcascade()

    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 milliseconds

```

```

        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.face_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\Trainer.yml")

res = "Profile Saved Successfully"

message1.configure(text=res)

message.configure(text="Total Registrations till now : ' + str(ID[0]))

def getImagesAndLabels(path):

    # get the path of all the files in the folder

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

    # create empty 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 imagePath:

        # 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_haarcascade()

    assure_path_exists("Attendance/")

    assure_path_exists("StudentDetails/")

    for k in tv.get_children():

        tv.delete(k)

    msg = "

    i = 0

    j = 0

    recognizer = cv2.face.LBPHFaceRecognizer_create() #
cv2.createLBPHFaceRecognizer()

    exists3 = os.path.isfile("TrainingImageLabel\Trainer.yml")

    if exists3:

        recognizer.read("TrainingImageLabel\Trainer.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='#2d420a')
```

```
frame1 = tk.Frame(window, bg="#c79cff")
```

```
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)
```

```
frame2 = tk.Frame(window, bg="#c79cff")
```

```
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)
```

```
message3 = tk.Label(window, text="Face Recognition Based Attendance Monitoring  
System" ,fg="white",bg="#2d420a" ,width=55 ,height=1,font=('comic', 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="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 18, ' bold '))
```

```
datef.pack(fill='both',expand=1)
```

```
clock = tk.Label(frame3,fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic',
18, ' bold '))
```

```
clock.pack(fill='both',expand=1)
```

```
tick()
```

```
head2 = tk.Label(frame2, text="                For New Registrations                ",
fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ' ) )
```

```
head2.grid(row=0,column=0)
```

```
head1 = tk.Label(frame1, text="                For Already Registered                ",
fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ' ) )
```

```
head1.place(x=0,y=0)
```

```
lbl = tk.Label(frame2, text="Enter ID",width=20 ,height=1 ,fg="black" ,bg="#c79cff"
,font=('comic', 17, ' bold ' ) )
```

```
lbl.place(x=80, y=55)
```

```
txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))
```

```
txt.place(x=30, y=88)
```

```
lbl2 = tk.Label(frame2, text="Enter Name",width=20 ,fg="black" ,bg="#c79cff"
,font=('comic', 17, ' bold '))
```

```
lbl2.place(x=80, y=140)
```

```
txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold ' ) )
```

```
txt2.place(x=30, y=173)
```

```
message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile" ,bg="#c79cff"
,fg="black" ,width=39 ,height=1, activebackground = "#3ffc00" ,font=('comic', 15, ' bold
'))
```

```
message1.place(x=7, y=230)
```

```
message = tk.Label(frame2, text="" ,bg="#c79cff" ,fg="black" ,width=39,height=1,
activebackground = "#3ffc00" ,font=('comic', 16, ' bold '))
```

```
message.place(x=7, y=450)
```

```
lbl3 = tk.Label(frame1,
text="Attendance",width=20 ,fg="black" ,bg="#c79cff" ,height=1 ,font=('comic', 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=('comic', 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="#ff7221" ,width=11 ,activebackground = "white"
,font=('comic', 11, ' bold '))

clearButton.place(x=335, y=86)

clearButton2 = tk.Button(frame2, text="Clear",
command=clear2 ,fg="black" ,bg="#ff7221" ,width=11 , activebackground = "white"
,font=('comic', 11, ' bold '))

clearButton2.place(x=335, y=172)

takeImg = tk.Button(frame2, text="Take Images",
command=TakeImages ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))
```

```

takeImg.place(x=30, y=300)

trainImg      =      tk.Button(frame2,      text="Save      Profile",      command=psw
,fg="white"    ,bg="#6d00fc"    ,width=34    ,height=1,    activebackground = "white"
,font=('comic', 15, ' bold '))

trainImg.place(x=30, y=380)

trackImg      =      tk.Button(frame1,      text="Take      Attendance",
command=TrackImages    ,fg="black"    ,bg="#3ffc00"    ,width=35    ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))

trackImg.place(x=30,y=50)

quitWindow      =      tk.Button(frame1,      text="Quit",
command>window.destroy    ,fg="black"    ,bg="#eb4600"    ,width=35    ,height=1,
activebackground = "white" ,font=('comic', 15, ' bold '))

quitWindow.place(x=30, y=450)

#END

window.configure(menu=menubar)

window.mainloop()

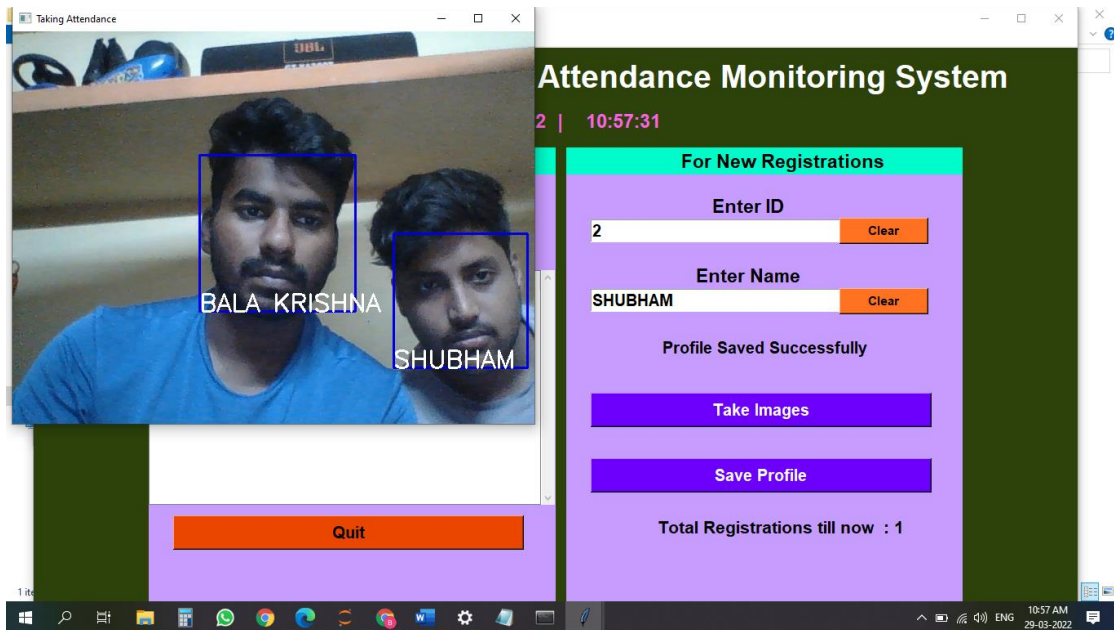
```

5. TESTING

5.1 System Testing:

Case 1:

We will test the system by marking the attendance of two person at a time and test whether the system can mark the attendance of 2 or more at a time.

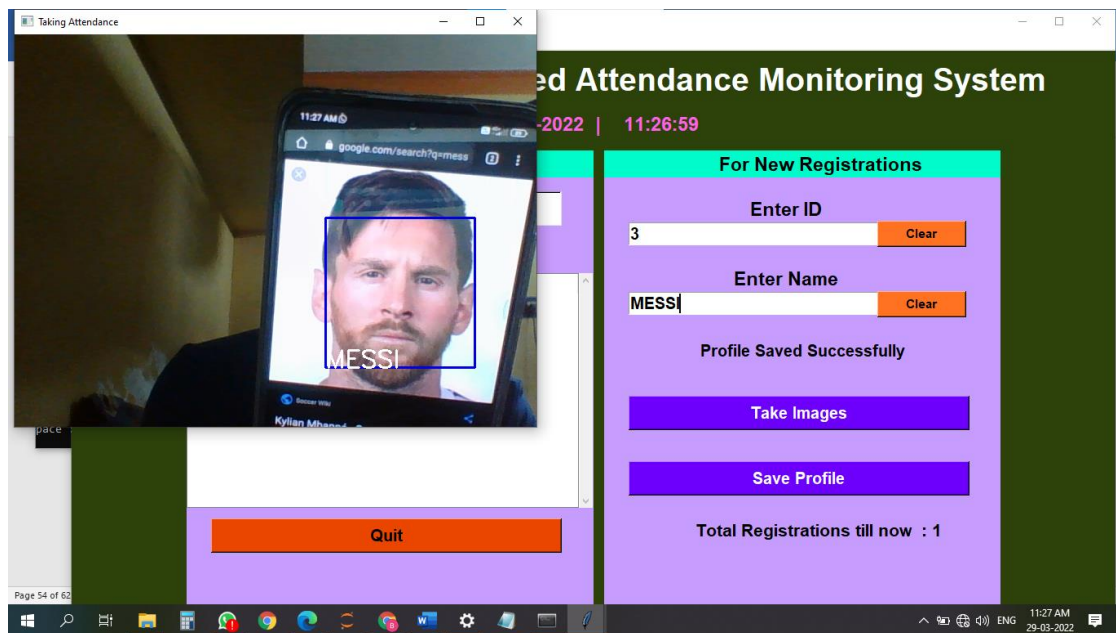


Result:

The system will mark attendance only for the last person who was present in the camera frame.

Case 2:

Using an image for registration and taking attendance of the student.

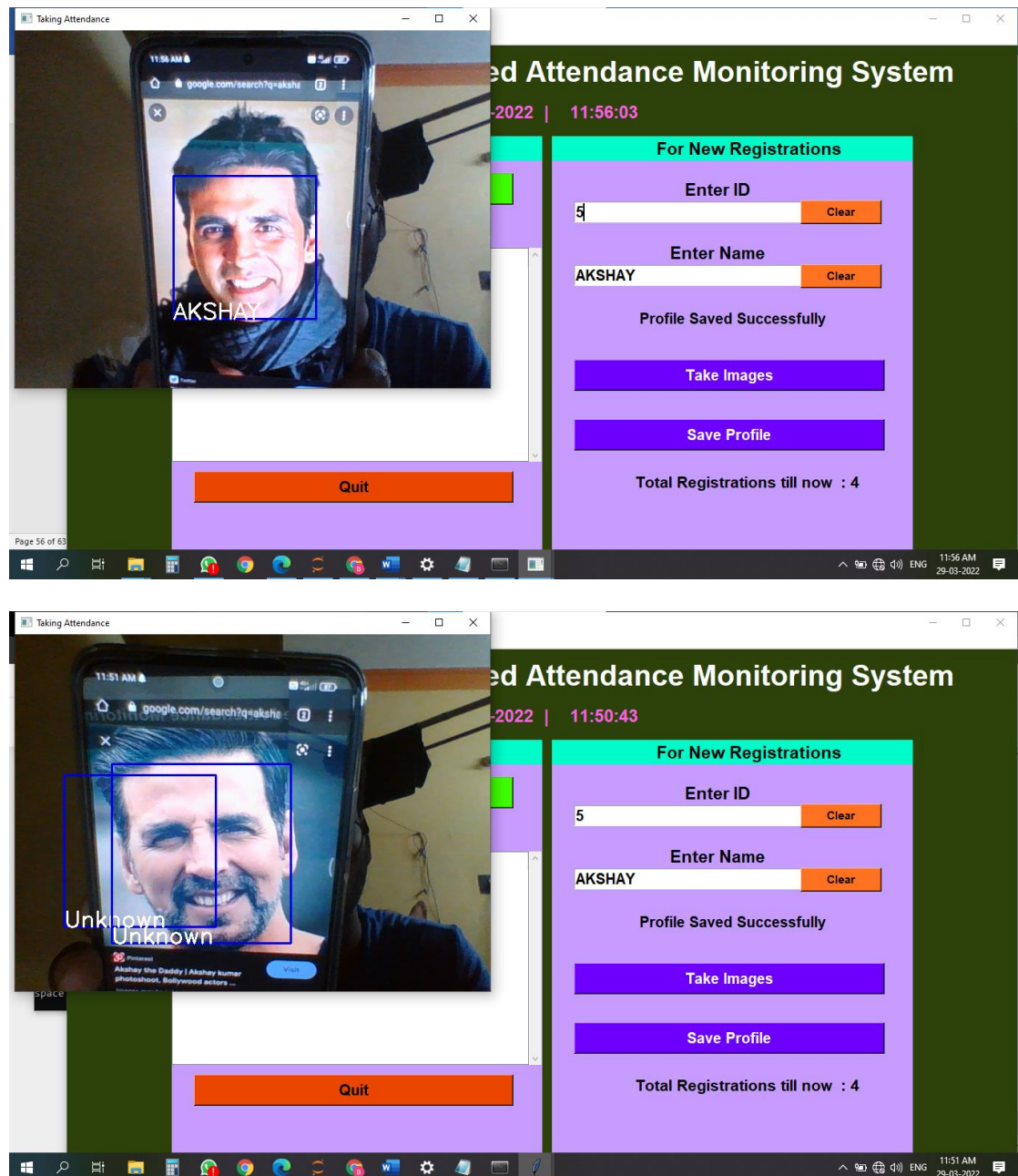


Result:

The system successfully registered the details and successfully marked the attendance.

Case 3:

Checking whether the system will be able to recognize a same person with beard and beardless and with different hairstyle.

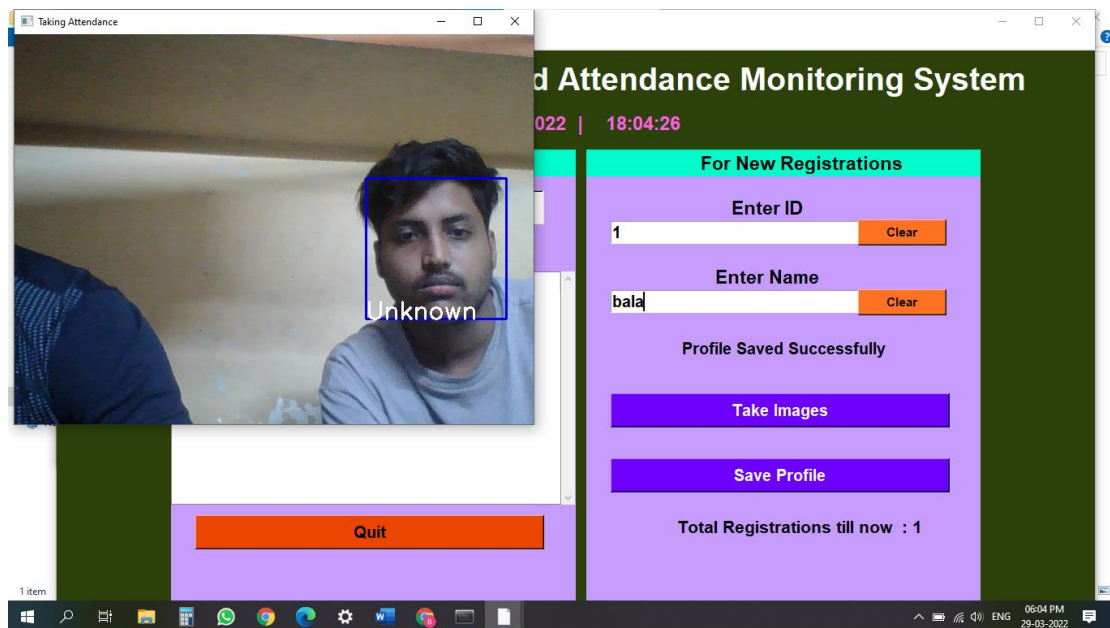


Result:

The system failed to recognize the person if the person has a more beard and moustaches. There can be slightly minor difference between the training Images and the person then the system can recognize the person, or else system won't be able to recognize the person.

Case 4:

What if an unknown person tries to give attendance?

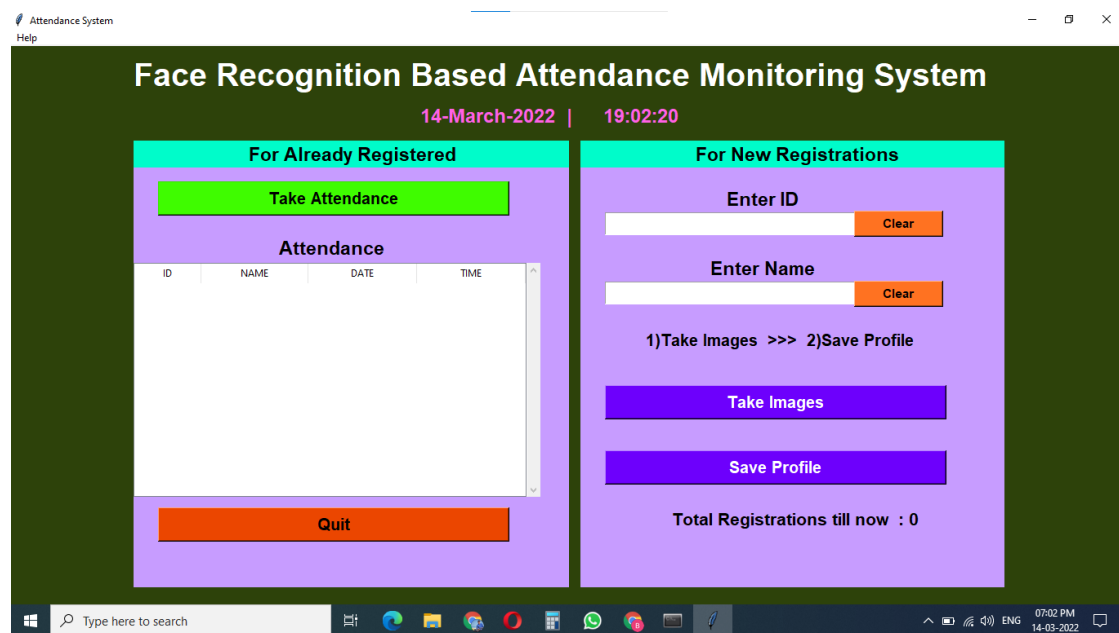


Result:

Then the system will display a message UNKNOWN below the face of that person who is in the frame of the camera.

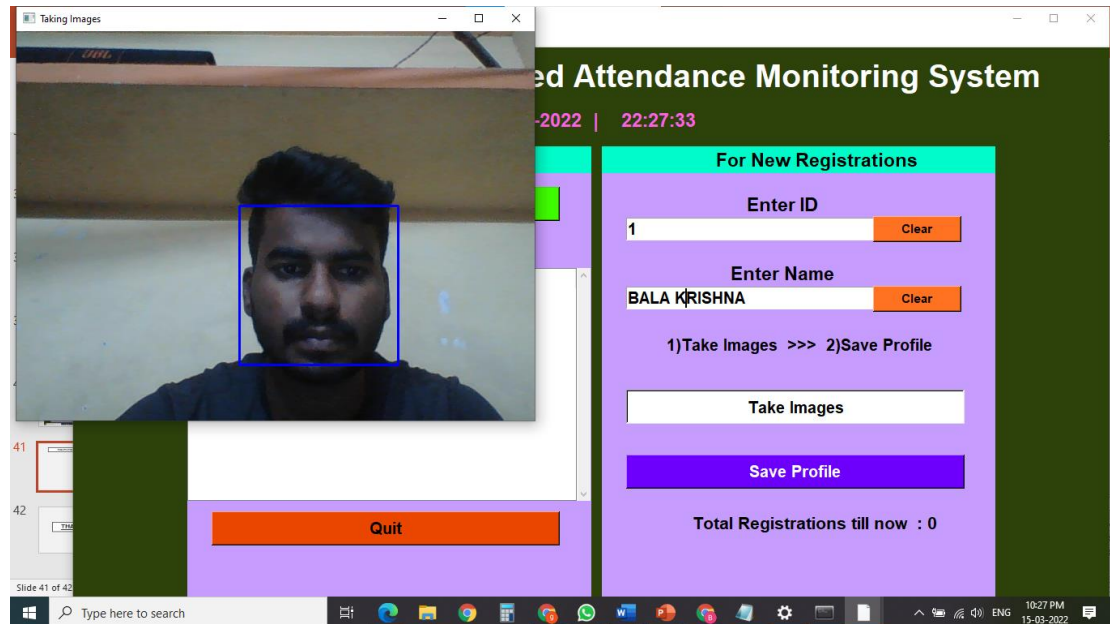
6. SCREENS

6.1 Front View



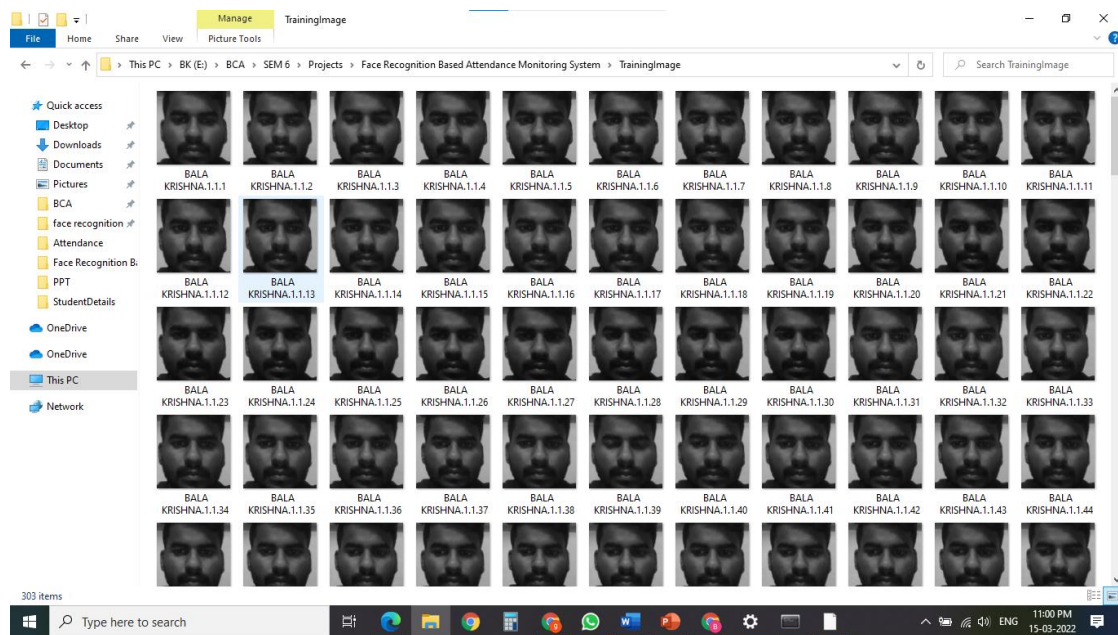
6.2 Registration of new student

In the process of registration, the system asks the student's ID and NAME and window will be opened in which the camera is used to collect the multiple images of the student, the student have to stay still in front of the camera while the system is collecting the images of him or her.



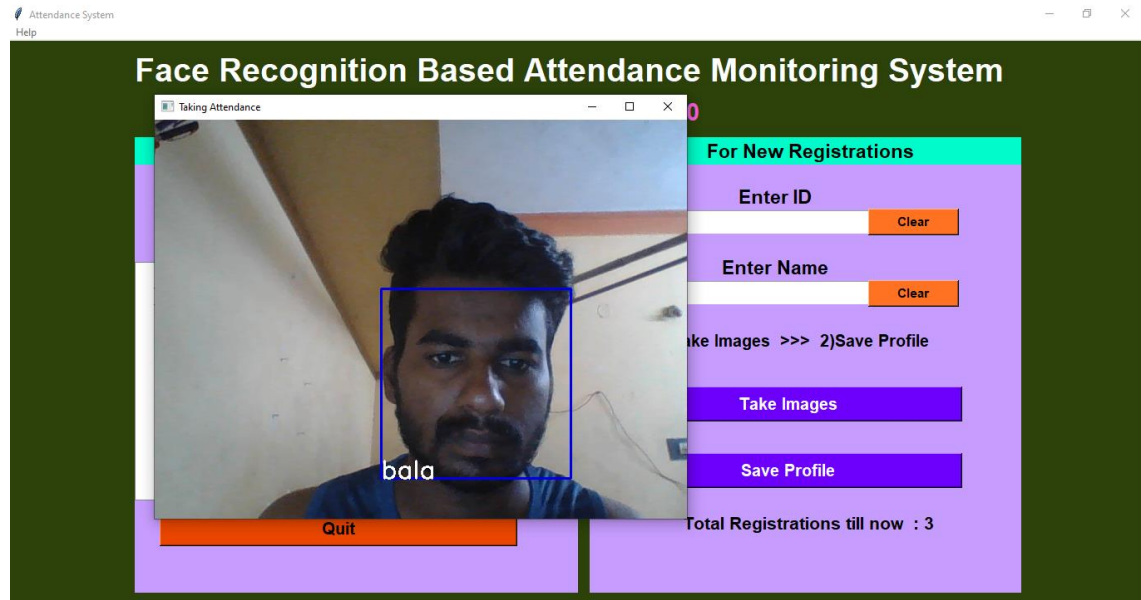
6.3 Training images

The system collects 101 images of the student while registration and all photos are stored in a database or folder called TrainingImages.



6.4 Taking Attendance

If the system identifies the person or student then it will display the NAME of the person below his or her face as shown below, to finish giving attendance we have to press 'q' to quit.



6.5 Student's Confirmation for attendance

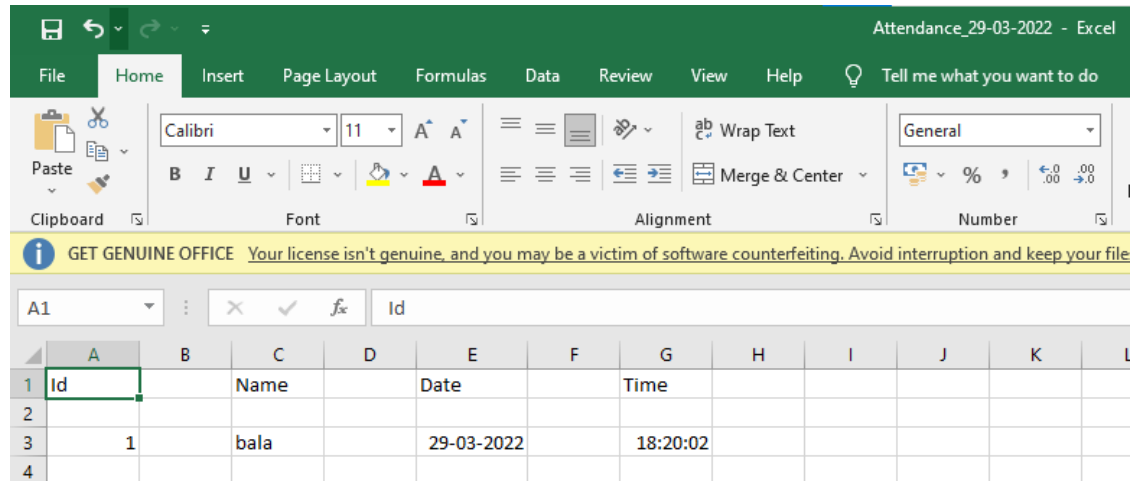
After giving attendance, for student's assurance that his or her attendance marked successfully, in the white part list of the student's details will be displayed who had already given the attendance. For reference below an image is provided:

Attendance			
ID	NAME	DATE	TIME
1	bala	29-03-2022	18:20:02
Quit			

7. REPORT

7.1 Attendance record (date wise separate)

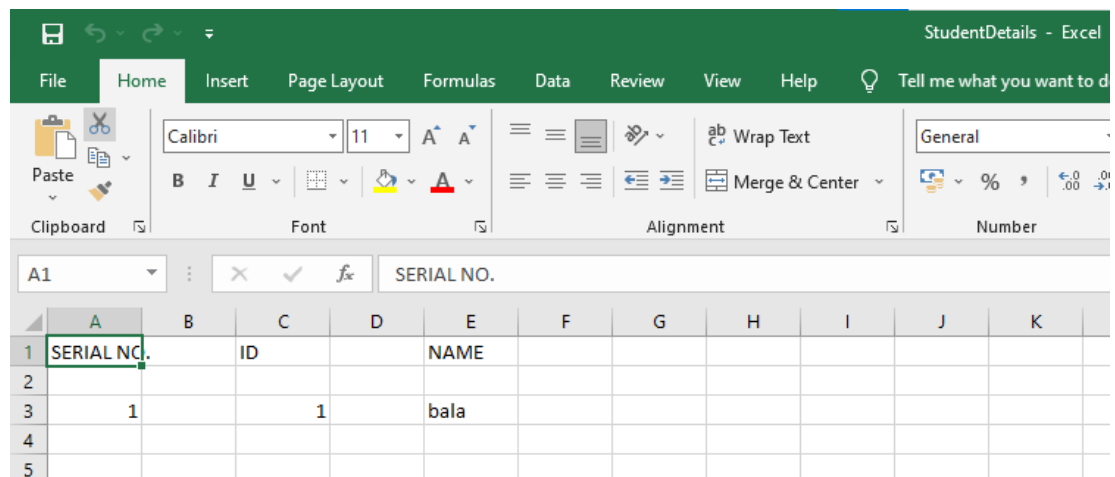
In the report we can see that the student's attendance is marked in CSV files, for different dates each CSV file is created automatically with name of the Attendance and following that particular date. This happens when the student gives the attendance.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Id		Name		Date		Time					
2												
3	1		bala		29-03-2022		18:20:02					
4												

7.2 Student registration record

When a student is registered successfully in the system his/her details like name and ID no. will be stored in CSV file as shown below:



	A	B	C	D	E	F	G	H	I	J	K	L
1	SERIAL NO.		ID		NAME							
2												
3	1		1		bala							
4												
5												

8. CONCLUSION

I have implemented an attendance management system for student's attendance. It helps to reduce time and effort, for marking attendance of the students. The whole system is implemented in the Python programming language. Facial recognition techniques used for the purpose of student attendance. And also, this record of student attendance can further be used mainly in exam related issues like who are attending the exams and who are not attending. On this project, there is some further work remaining like installing the system in the classrooms. It can be constructed using a camera and computer.

This system can also be used in offices for attendance purposes and since this system also records the timing, and the data of employee's arrival and departure timing will be well recorded.

9. REFERENCES

1. <https://iran-lms.com/images/images/Books/PDF/Object-Oriented-Software-Engineering-Using-UML-Patterns-and-Java-Prentice-Hall-2010-Bernd-Bruegge-Allen-H.-Dutoit.pdf>
2. https://www.researchgate.net/publication/341870242_Smart_Attendance_System_using_OPENCV_based_on_Facial_Recognition