Software Engineering Mini Project Report
(Smart Attendance System with Face Recognition using OpenCV)

Shumbul Arifa (181CO152), Keerti Chaudhary (181CO226)

Students of the Department of Computer Science and Engineering, National Institute of Technology Karnataka

Surathkal, Karnataka, India

shumbul.181co152@nitk.edu.in , keerti2001.kc@gmail.com

Abstract. The Oxford Dictionary defines a face as the part of a person's head from the forehead to the chin or the corresponding part of an animal. In human interactions, the face is the most significant factor because it contains important information about an individual. All humans will acknowledge people from their faces. The proposed solution is to develop an operating prototype of a system that may facilitate class attendance management for the lecturers within the lecture rooms by detecting the faces of scholars from an image taken in a classroom. The database can store the faces of scholars, once the face of the individual matches with one in all the faces held within the database then the attendance is recorded. In recent years, analysis has been dispensed and face recognition and detection systems are developed. A number of these are used on social media platforms like Facebook, banking apps, government offices, etc.

Keywords: Face Recognition, Face Detection, OpenCV, Attendance System, LBPH

1 Introduction

Maintenance of attendance is incredibly necessary altogether at the institutes for checking the performance of employees/students. Some institutes take attending manually using paper or file-based approaches and a few have adopted ways of automatic attendance using some biometric techniques. The recent methodology for taking attendance is by calling out the name or roll number of the scholar to record attendance. It's a long and less efficient method of marking attending as a result of as we all know the info written within the paper typically may be lost or is less accurate as a result of students ofresultten mark every other's attending proxy. Therefore, to unravel these issues and avoid errors, we recommend computerizing this method by providing a system that records and manages student's attending mechanically with no need for lecturers' interference.

Every biometric system consists of the enrollment process during which distinctive features of someone are stored within the database and then there are processes of identification and verification. These 2 processes compare the biometric feature of someone with previously stored biometric details captured at the time of enrollment. Biometric templates are of many varieties like Fingerprints, Eye Iris, Face, Signature, and Voice. Our system uses the face recognition approach for the automated Attendance Management System.

By considering this truth our system is going to be quicker and correct in marking the attendance of individual students. Face recognition consists of two steps, in the first step faces are detected in the image and then these detected faces are compared with the database for verification. Face detection is employed to find the position of face region and face recognition is employed for marking the attendance. The info can store the faces of scholars. Once the face of the scholar matches with one in each of the faces kept within the database then the attendance is recorded.

Various sturdy algorithms are developed that offer accurate performance to tackle face detection and recognition problems. These algorithms or methods are the most successfully and widely used for face detection and recognition applications. The algorithms are as follow:

1. *Principal Component Analysis (PCA)*

It is a dimensionality reduction technique that employs Eigenvalues and EigenVectors to reduce dimensionality and project a training sample/data on small feature space.

2. *Linear Discriminant Analysis (LDA)*

This methodology uses concepts of class. Its goal is to perform dimensionality reduction while saving as much of the class discriminatory information as possible. It minimizes variation within each class and maximizes class separation.

3. *Skin Color Based Approach*

One of the simplest algorithms for detecting skin pixels is to use a skin color algorithm. Each pixel is classified as skin color and non-skin color. This classification is based on its color component, which is modeled by Gaussian probability density.

4. *Artificial neural networks based algorithm*

An artificial neural network (ANN) is mostly used as a method for recognition. ANN will be implemented once a face has been detected to identify and recognize who the person is by calculating the weight of the facial information.

2 Literature review

Several algorithms and techniques for face recognition developed within the past few years by researchers. It has likewise been a functioning exploration subject for analysts till this current date. This examination has not many requests, for example, image processing, machine learning approach, pattern recognition, computer vision, neural network.

The authors described in the paper [1] *(Face Recognition Based Attendance Marking System), K.Senthamil Selvi, P.Chitrakala,* and *A.Antony Jenitha,* have described the concept to find real-time human faces, for that, Principal Component Analysis has been used to recognize the faces detected with a high accuracy rate. The system consists of a camera that captures the pictures of the worker and sends it to the image enhancement module. Attendance is maintained on the server thus anyone will access it for functions like administration in the institution. On recognizing, the second attendance database contains data concerning the workers, and conjointly uses it to mark attendance. Once it's compared to ancient attendance marking, this technique seems to save time and also helps to monitor the scholars.

In paper [2] *Face Recognition Based Student Attendance System with OpenCV* by *CH. Vinod Kumar* and, *Dr. K. Raja Kumar*, it is suggested that face recognition based (mostly on student attendance systems) with OpenCV takes the number of individuals present within the classroom and takes attendance of each of them using face detection and recognition algorithms to see the particular identification of persons such that which of them are present. In this proposed system faces are detected and recognized using the Eigen object detector algorithm. This is done with the help of OpenCV with haar cascades that are present in the OpenCV inbuilt. For this, the system associate HD digital camera is needed to require the input pictures in an exceedingly fastened space wherever the camera is located. The photographs that are taken from the camera are detected with a haar cascade. The frontal faces and eyes are then trained with the Eigen algorithm, the trained faces are stored in a database first, and compared to the trained images after comparing it makes attendance to the recognized persons.

The objectives of the thesis [3] *Facial Recognition using OpenCV* by *Divyarajsinh N. Parmar* and *Brijesh B. Mehta,* are to supply a group of detection algorithms which will be later prepackaged in an easily-portable framework amongst the various processor architectures we tend to see in machines (computers) nowadays. A way is described in this paper, which supported the way to resize a picture along with keeping its aspect-ratio an equivalent. It uses a sort of face detector referred to as a Haar Cascade classifier, provided by OpenCV. Given a picture, which might come from a file or live video, the face detector examines every image location and classifies it as "Face" or "Not Face". The drawbacks like the image probably would not recognize them in a bright room problem (referred to as "lumination dependent"), and lots of different problems, like the face, ought to even be in a very consistent position are mentioned during this paper.

In the projected system [4] by *Shervin Emami* and *Valentin Petruţ*, common ways like a holistic matching technique, feature extraction technique, and hybrid ways that are used. The paper relies on analysis in face recognition. The paper provides readers with an additional sturdy understanding of face recognition ways & applications. Face recognition systems establish people by their face photos. Face recognition systems establish the presence of an accredited person rather than merely checking whether or not or not a sound identification or secret's being utilized or key's being employed or whether or not the user is aware of the key personal identification numbers(pins) or passwords. The face recognition system directly compares the face photos of individuals and does not use ID numbers to differentiate one from the others. Once the very best two matched faces are extremely just like the query face image, manual review is needed to create certain they're so different persons therefore on eliminating duplicates.

The paper [4] *Smart Attendance Monitoring System* by *Sudhir Bussa, Shruti Bharuka, Ananya Mani,* and *Sakshi Kaushik* presents the automatic attendance management system for convenience or data reliability. The system is developed by the integration of ubiquitous components to make a portable device for managing the students' attendance using Face Recognition technology. This system achieves high frame rates working only with the information present in a single grey-scale image. These alternative sources of information can also be integrated with our system to achieve even higher frame rates.

In the paper *Smart Attendance System using OPENCV based on Facial Recognition [6] by Sudhir Bussa, Ananya Mani, Shruti Bharuka,* and *Sakshi Kaushik*, the Open CV primarily based face recognition approach has been planned. This model integrates a camera that captures associate degree input image, an algorithmic rule for detecting a face from an input image, encoding and identifying the face, marking the attendance in a spreadsheet, and changing it into a PDF file. The training database is formed by training the system with the faces of the approved students. The cropped pictures are then stored as a database with respective labels. The features are extracted using the LBPH algorithmic rule. The expected system engages the face recognition for automating the attendance system of scholars or staff without their involvement. A digital camera is employed for capturing photographs of scholars and staff. The faces within the captured pictures square measure detected and compared with the photographs within the database and also the attendance is marked

Smart Attendance System with Face Recognition

The main intention of this project is to unravel the problems encountered within the previous attending system whereas reproducing a fresh innovative sensible system that may offer convenience to the institution. In this project, a sensible device will be developed that is capable of recognizing the identity of every individual and eventually record down the info into a database system. Apart from that, an interface will be developed to supply visual access to the data. The followings are the expected work to be done:

- The targeted groups of the attendance monitoring system are the students and staff of an educational institution.
- Users should be able to register through their already existing accounts.
- The user's face should be sensed and captured by the camera
- The database of the attendance management system can hold an individual's information.
- The detected face's data should be matched with the database
- The facial recognition process can only be done for one person at a time.

Motivation

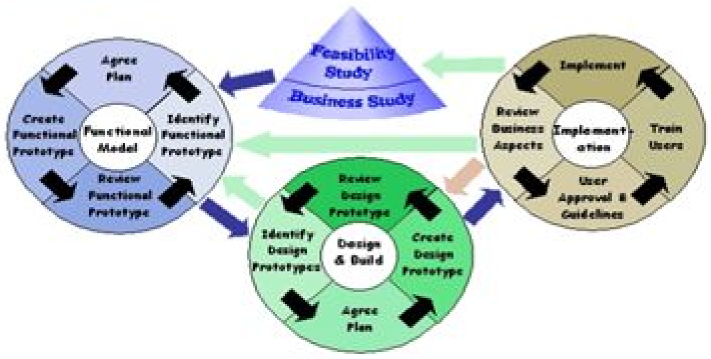
The project is chosen to resolve the real-time drawback of manually taking attendance by line the name or roll number of the student to record attendance. It was a long and lesser efficient method of marking attendance because as we know the data written in the paper can be lost or can be less accurate because students often mark each other's attendance proxy. As a result, to resolve these issues and avoid errors, we recommend computerizing this method by providing a system that records and manages students' attending mechanically while not having lecturers' interference.

3.1 Software Engineering Concepts

Software Engineering has emerged as a discipline just like civil, mechanical, and electrical engineering, wherever **engineering principles such as modeling, prototyping, designing, developing, testing, delivery, installation, and maintenance** square measure the broad steps plain-woven along to succeed the final engineering objective. Whereas these engineering disciplines square measure a lot of or less a science, based mostly on arithmetic, material sciences, physics, and chemistry, the constant isn't true for software system engineering. Software system engineering tends to be nearer to science owing to its approach being scientific, systematic, and also the use of standards, protocols tools, and techniques, Further; it encourages the principles of sensible and design. In the view of software system engineering at the core level is a subject area.

3.2 Software Process Model

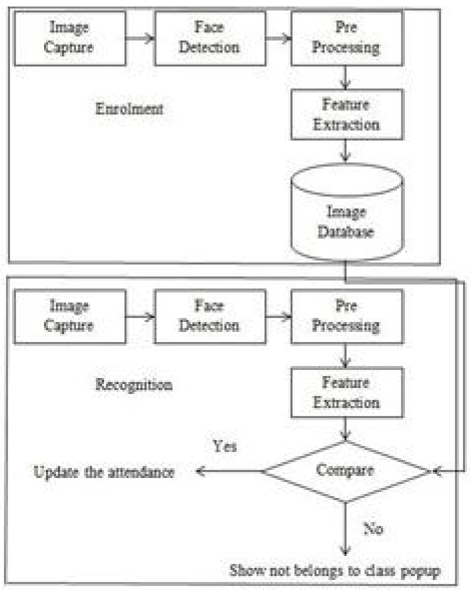
The agile project delivery framework is the approach that may be used for the event of the system during this project. DSDM is an agile methodology approach primarily used as a code development method. The event of this project is requiring user involvement to own timely visible results. data gathered from the literature review shows researchers using completely different algorithms in face detection and recognition. However, because it is a current analysis space, this project needs progressive implementation in smaller functionalities that can be placed along at the tip of a whole system. The project objectives laid out in the proposal will solely be achieved with the experience of the user, as functionalities are going to be prioritized so as of importance aboard continuous user involvement. In contrast to alternative approaches (Waterfall) wherever the stages of implantations are clearly outlined, it's preferred to use the approach which can adapt simply to changes created throughout the implementation.



DSDM Development Process.

3.3 Conceptual Model

The stages in the proposed Smart Attendance System with Face Recognition are as shown in the figure below. This is the block diagram and also the conceptual model of the attendance system.

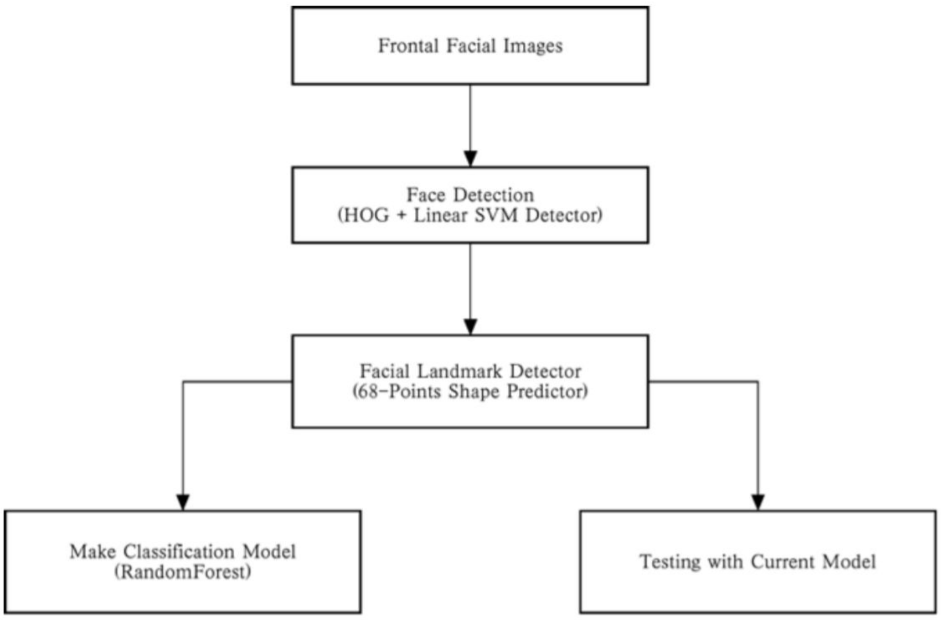


System block diagram

3.4 Overview of Architecture

Smart Attendance System with Face Recognition has the architecture as the following step. The camera needs to install in the front which can capture an entire face of the student inside the class. In the first phase after the camera has been captured; the captured image is transferred into the system as an input. The image capture from the camera sometimes comes with darkness or brightness which needs to do an enhancement on it such as convert to a gray image.

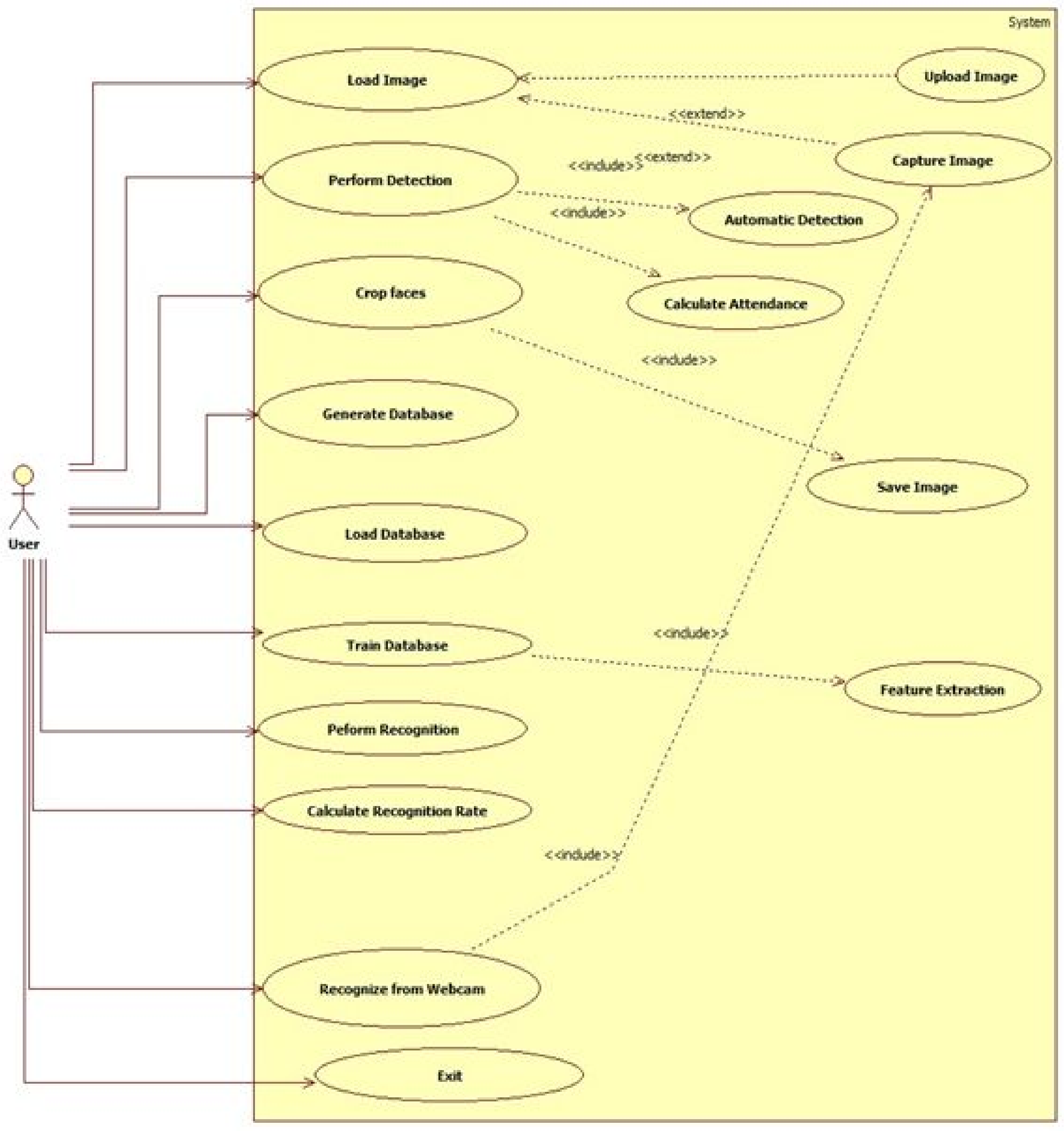
Face detection performs locating and extracting face image operations for face recognition systems using the 68-points shape predictor. The model design of this system will form a blueprint for the implementation that will be put together to achieve the project objectives and best performance for the final product. This system design consists of activities that fit between software requirements analysis and software construction.


Frontal Facial Images
Face Detection (HOG + Linear SVM Detector)
Facial Landmark Detector (68-Points Shape Predictor)
Make Classification Model (RandomForest)
Testing with Current Model

The framework of a basic Face Detector

3.5 User Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior. It only summarizes **some of the relationships** between use cases, actors, and systems. It does **not show the order** in which steps are performed to achieve the goals of each use case.



3.6 The User Story:

As a user, the client wants a system where they can load an image that will automatically detect the number of faces on the image. The system should have the option to capture an image using an inbuilt webcam on a laptop. As a user, the system should be able to crop the faces on an image after detection and store them on a folder/dataset that will be used for recognition purposes in the second phase of the system. The system should be able to automatically count the number of faces detected on the image.

As a user, the client requests the second phase of the system to be able to match faces stored on a dataset against input images which are either detected from the first phase or captured by an input device (camera). When the face is detected while taking attendance, it is recognized and details of name and time of attendance taken are entered in the database.

4 Tools and Technologies

The software technologies we have used for the mini-project include:

- Python is the artificial language for our project, and additionally one in every of the extremely powerful and renowned programming languages better-known for its large use in machine learning and computing.
- Visual Studio code editor, because the editor is incredibly reliable with loads of options enclosed in it. It's a robust tool for developers as virtually each programming language may be executed in it.
- Pycharm Libraries:

○ Numpy - could be a library for Python, adding support for multi-dimensional arrays and matrices, in conjunction with an enormous assortment of high-level mathematical functions to operate on these arrays.

○ 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.

○ Haar Cascade - is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

○ Datetime - It's a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and info.

○ Face_Recognition - Recognize and manipulate faces from Python or the command line with the world's simplest face recognition library.

○ OpenCV - a library of programming functions primarily geared toward real-time computer vision.

Local Binary Patterns Histogram (LBPH) Algorithm

Local Binary Pattern (LBP) can be a clear yet productive surface that puts marks on parts of the picture by impeding the area, all things considered, and seeing the outcome as a paired width. It was resolved that when LBP is joined with histograms of characterized angles (HOG) directed, it improves the securing execution of more informational indexes. Utilizing LBP joined with histograms we can represent to confront pictures with a vector of direct information.

The LBPH algorithm works in 5 steps as given below:

1. Parameters: LBPH uses 4 parameters:

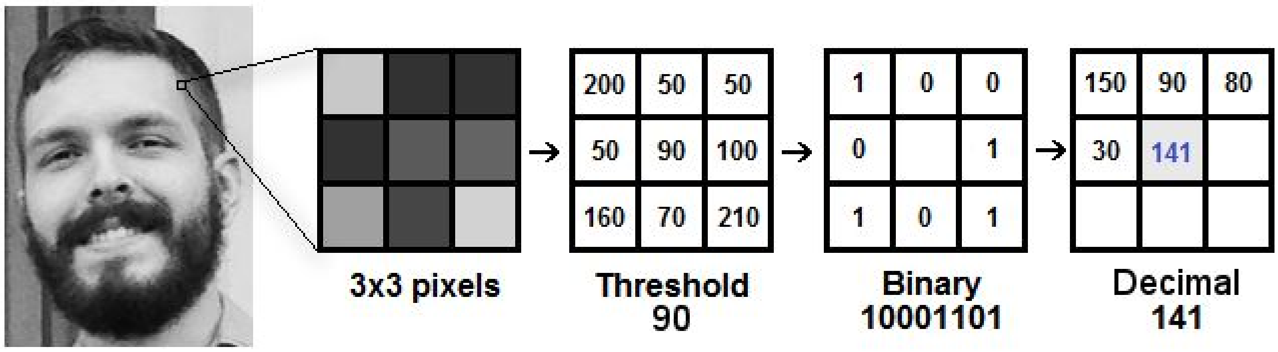
- Radius: the radius is used to create a local binary pattern and represents the radius around the center pixel. The frequency is set to 1.
- Neighbors: the number of sample points to form a circular area for a binary. Keep in mind: the more points you enter, the higher the computer cost. The frequency is set to 8.
- Grid X: the number of cells in a horizontal direction. The more cells, the better grid, the vector size of the emerging element. The frequency is set to 8.
- Grid Y - the number of cells in a straight line. The more cells, the better grid, the vector size of the emerging element. The frequency is set to 8.

2. Algorithm Training

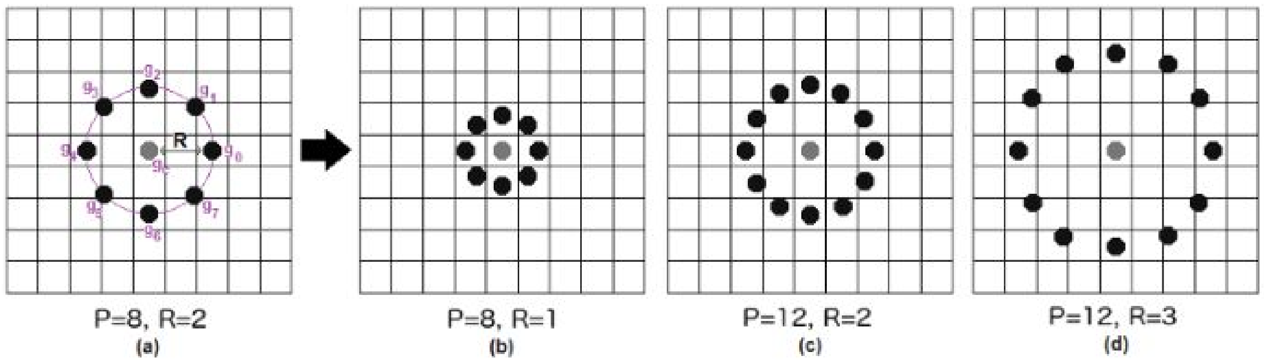
The first step is to train the model. To do so, we use a database that contains photos of people we would like to inform. Next, an ID (also a number or personal name) for all images is added, so the algorithm can use this data to identify the input image and give the result. photos of the same person must have the same ID. Since the training set has already been created, let's take a look at the LBPH process steps.

3. Applying for LBP operation

Next thing is to create an intermediate image that defines the original image on a highway, with the light of the face. To do so, algorithmic law uses a window view, supporting the range of frames and neighbors.

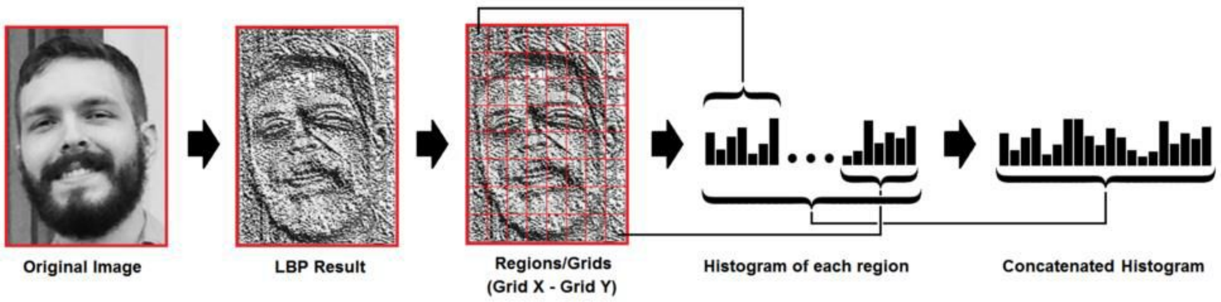


- We have a gray face picture.
- We can find part of this image as a 3x3 pixel window.
- It can also be represented as a 3x3 matrix containing the thickness of each pixel (0 ~ 255).
- After that, we need to take the median value of the matrix that will be used as the limit.
- This number will be used to describe new values from eight neighbors.
- For each median of the middle value (limit), we set a new binary value. We set 1 value equal to or higher than the limit and 0 value less than the limit.
- Now, the matrix will contain only binary values (regardless of average value). We need to estimate each binary value from each location from the matrix line by line to the new binary value (e.g. 10001101). Note: some authors use other methods to synchronize binary values (e.g. clock direction), but the result will be the same.
- After that, we convert this binary number to a decimal value and set it to the center value of the matrix, which is a pixel from the first image.
- At the end of this process (LBP process), we have a new image that better represents the features of the original image.



It can be done by using bilinear interpolation. If a certain point of data is within a pixel, it uses values from the nearest 4 (2x2) pixels to measure the point of the new data point.

4. Recording Histograms:
Now, using the image created in the last step, we can use the Grid X and Grid Y parameters to split the image into multiple grids, as can be seen in the following picture:



- Since we have a gray image, each histogram (from each grid) will only contain 256 (0 ~ 255) positions representing the potential for each pixel power.
- After that, we need to sync each histogram to create a new and larger histogram. If we assume we have 8x8 grids, we will have 8x8x256 = 16.384 positions in the final histogram. The final histogram represents the features of the first image.

5. Performing face recognition:
At this point, the algorithm is already trained. Each created histogram is used to represent each image from the training database. So, when we are given an image to insert, we perform steps again for this new image and create a histogram representing the image.
- So to get an image similar to an input image we just need to compare the two histograms and replace the image with the nearest histogram.
- We can use various methods to compare histograms (calculating the distance between two histograms), for example, Euclidean distance, square-chi, total value, etc. In this example, we can use the Euclidean (most popular) range based on the following formula:

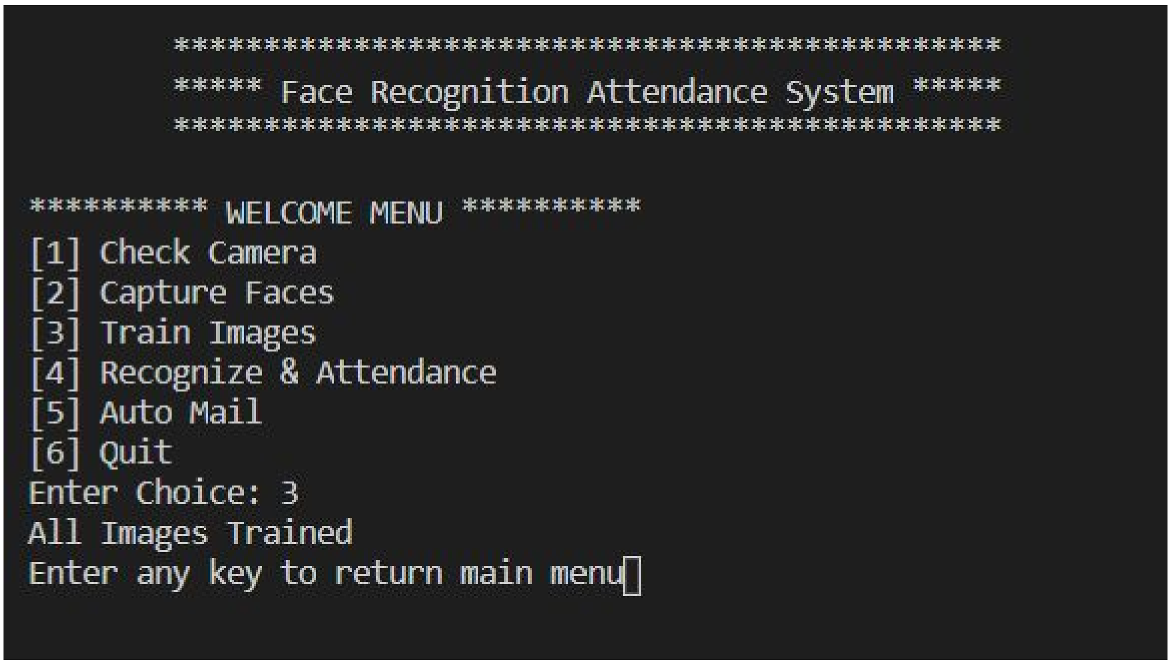
$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So the output of the calculation is an ID from a picture with a close-by histogram. The calculation ought to likewise restore a determined reach, which can be utilized as a proportion of 'certainty'. Note: don't be tricked by the word 'certainty', since low camouflage is better since it implies that the separation between the two histograms is nearer.
- We can then use the threshold limit and 'confidence' to automatically adjust if the algorithm has detected the image correctly. We can assume that the algorithm has detected success when confidence is below the defined limit.

5 Working

In our model, we will be providing the users the options to choose a given function from the above possible functions, the pseudocode for the same is given below:

1. Check Camera
2. Capture Faces
3. Train Images
4. Recognize & Attendance
5. Auto Mail
6. Quit



```
        **********************************************
        ***** Face Recognition Attendance System *****
        **********************************************


********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 3
All Images Trained
Enter any key to return main menu
```

The main menu of the working model

<u>Pseudocode:</u>

```
Begin
  ● if choice = 1
    call the function to check if the system/external camera is working
  ● elif choice = 2
    call the function which captures faces
  ● elif choice = 3
    call the function which trains images
  ● elif choice = 4
     the function that recognizes the person from the set of saved images from the database is
    called
  ● elif choice = 5
```

send Mail to the admin

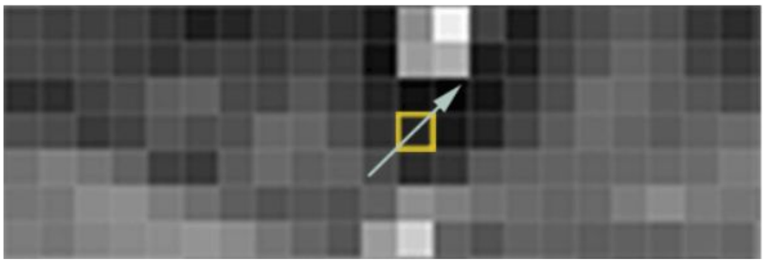
- else exit

End

Step - 1: Finding all the Faces

We start by looking out for faces in an image, we tend to start with making our image black and white as a result of color data isn't required to go looking out for faces. Then we'll investigate every single pixel in our image one at a time. For every single pixel, we might wish to appear at the pixels that directly encompass it. Our goal is to work out however dark this component is compared to the pixels directly encompassing it.

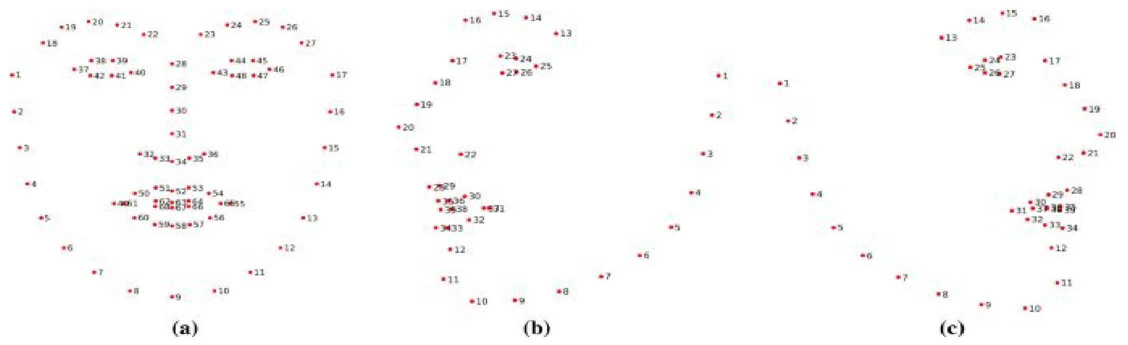
Then we would like to draw the arrow showing during which direction the image is obtaining darker. If you repeat that method for every single component within the image, you finish up with each component being replaced by an arrow. These arrows are referred to as gradients and that they show the flowing light to dark across the whole image:



A pixel in an image detected during the face detection stage

Step 2: Posing and Projecting Faces

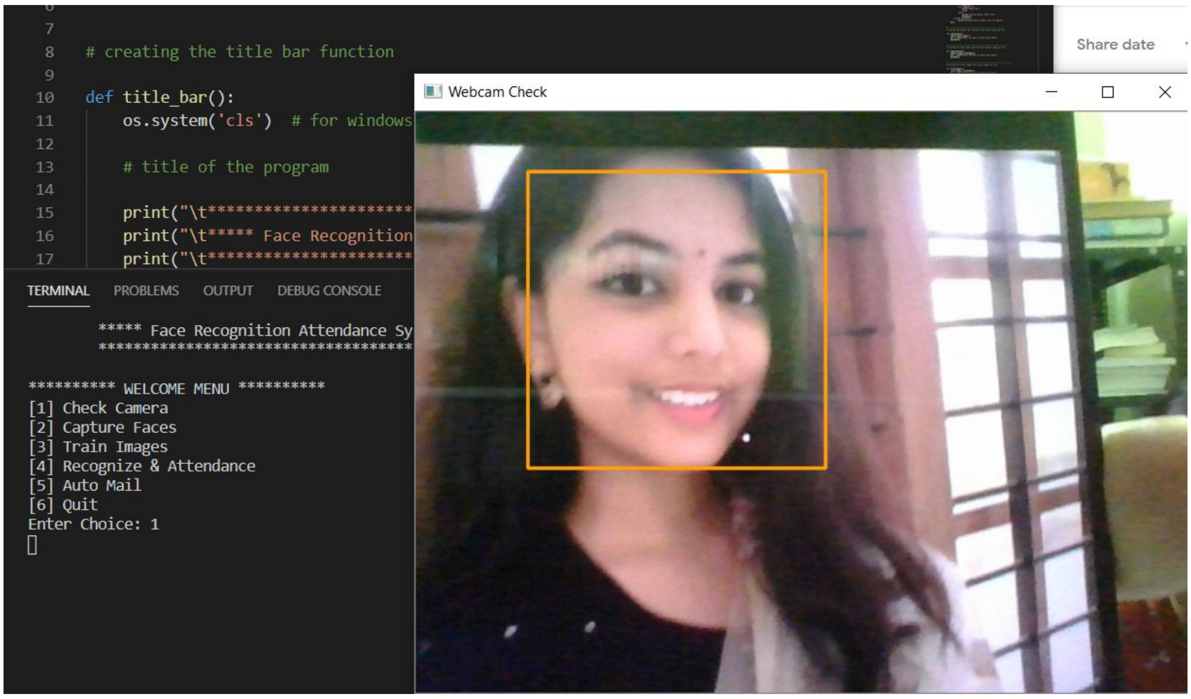
For this, we have planned to use an algorithm known as face landmark estimation. The essential plan is we'll return up with sixty-eight specific points (landmarks) — the upper part of the chin, the skin fringe of every eye, the inner fringe of the eyebrow, etc. Then we'll train a machine-learning model to be able to realize these sixty-eight specific points on any face:



Given below is the python code for recognizing the 68-points in a person's face using the face-recognition library.

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```

Thus we can locate the eyes and mouth, we can also rotate, scale, and shear the image which results in proper centering of the eyes and mouth to its best. Therefore it does not depend on the rotation of the face, we will be able to center the eyes and mouth in roughly the same position in the image.



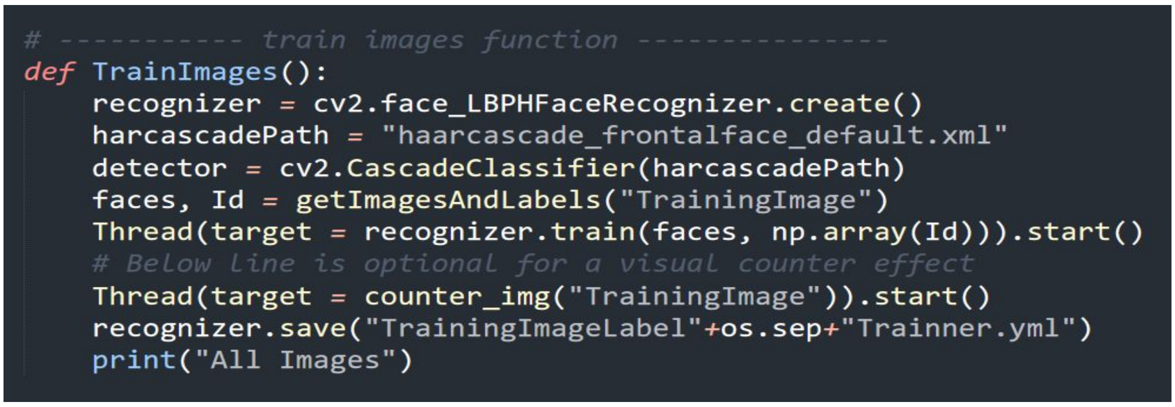
```
# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program

    print("\t**********************
    print("\t***** Face Recognition
    print("\t**********************
```

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE

        ***** Face Recognition Attendance Sy
        ***********************************

********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 1
```

Face getting detected by the model

Step 3: Encoding Faces

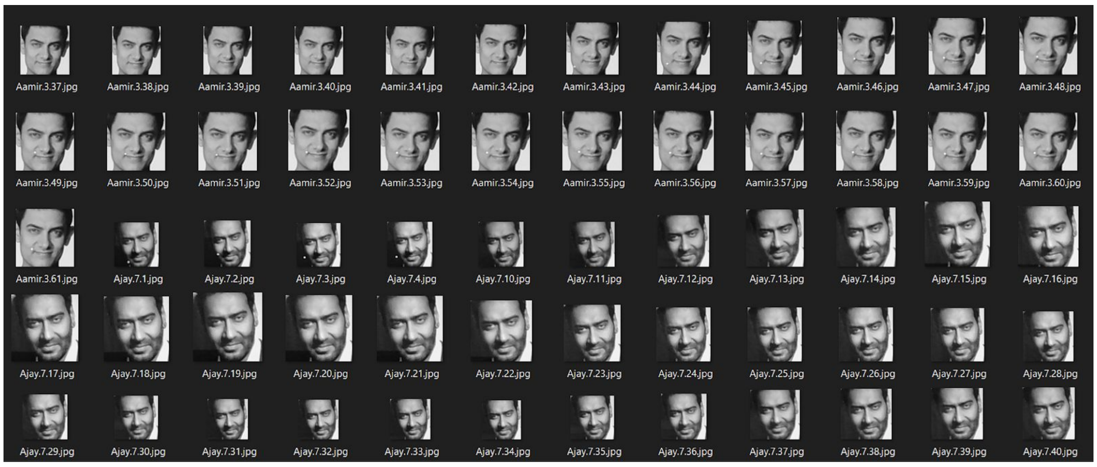
In this step, the face image would be encoded. We plan to find the most reliable way to measure a face. We tend to conjointly train our model during this method. If the system output provides over one rectangle, that indicates the position of the face, the space of center points of those rectangles has been calculated. If this distance is smaller than a pre-set threshold, the mean of those rectangles is going to be computed and set as the final position of the detected face.



```
# ----------- train images function ---------------
def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, Id = getImagesAndLabels("TrainingImage")
    Thread(target = recognizer.train(faces, np.array(Id))).start()
    # Below line is optional for a visual counter effect
    Thread(target = counter_img("TrainingImage")).start()
    recognizer.save("TrainingImageLabel"+os.sep+"Trainner.yml")
    print("All Images")
```

Function to train images in Python

It is a machine learning-based approach in which a cascade function is trained from a lot of positive and negative images. OpenCV comes with a trainer as well as a detector. If you want to train your classifier for any object like a car, planes, etc. you can use OpenCV to create one.



Snapshots captured while training the faces

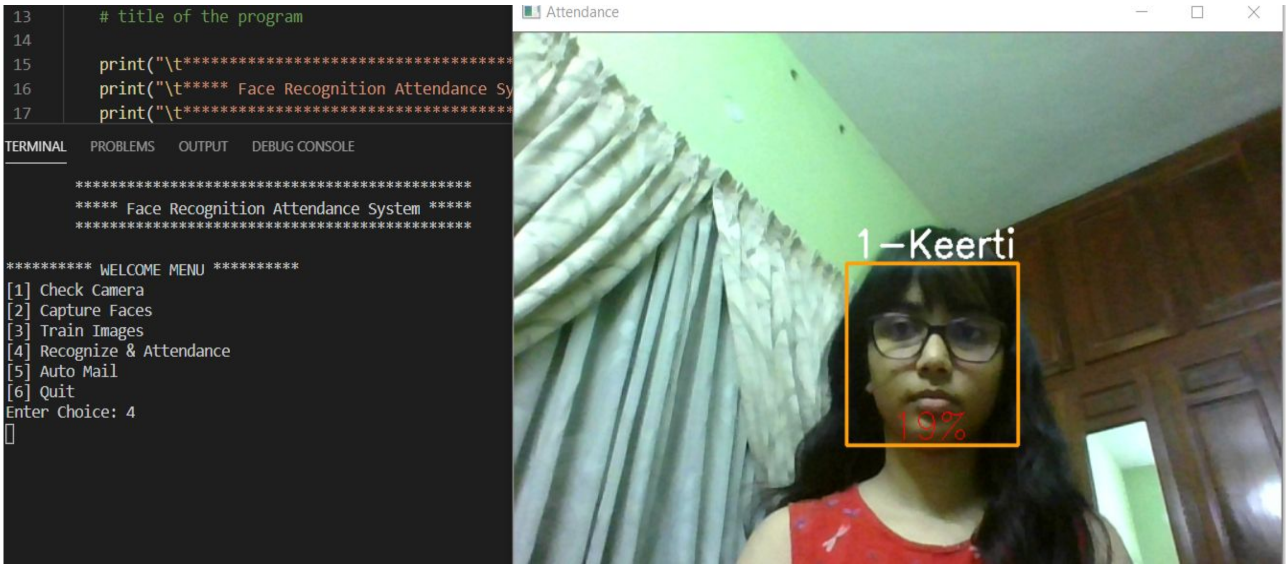
From the figure above, it can be noticed that ~30-40 images gets captured and stored in the database/ local storage. These images are then used to detect and recognize the person so that the attendance can be marked.

Step 4: Finding the person's name from the encoding

This last step is going to be the easiest in the whole process. All we have to do is identify the person in our database who has the closest measurements to our test image.

In other words, the steps involved in face detection for each of the faces found in an image are as follows:

1. Find face in an image
2. Analyze facial features
3. Compare against known faces
4. Make a prediction



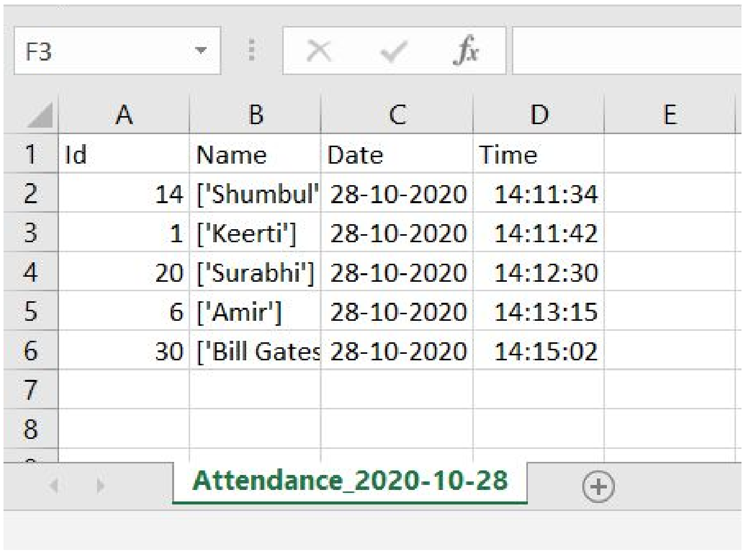
```
    # title of the program

    print("\t************************************
    print("\t***** Face Recognition Attendance Sy
    print("\t************************************
```

```
        ***********************************************
        ***** Face Recognition Attendance System *****
        ***********************************************

********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 4
```

Final step: predicting face and marking attendance

Step 5: Recording attendance

As suggested above, the face is detected by following the mentioned steps and in the final step the face is recognized and the attendance is marked for the recognized person, along with a timestamp. The system can be evaluated bypassing the face recognition function. The face recognition function returns the index of the recognized face on the database.



F3

	A	B	C	D	E
1	Id	Name	Date	Time	
2	14	['Shumbul'	28-10-2020	14:11:34	
3	1	['Keerti']	28-10-2020	14:11:42	
4	20	['Surabhi']	28-10-2020	14:12:30	
5	6	['Amir']	28-10-2020	14:13:15	
6	30	['Bill Gates	28-10-2020	14:15:02	
7					
8					

Attendance_2020-10-28

Fig. CSV file for the marked attendance for a small dataset

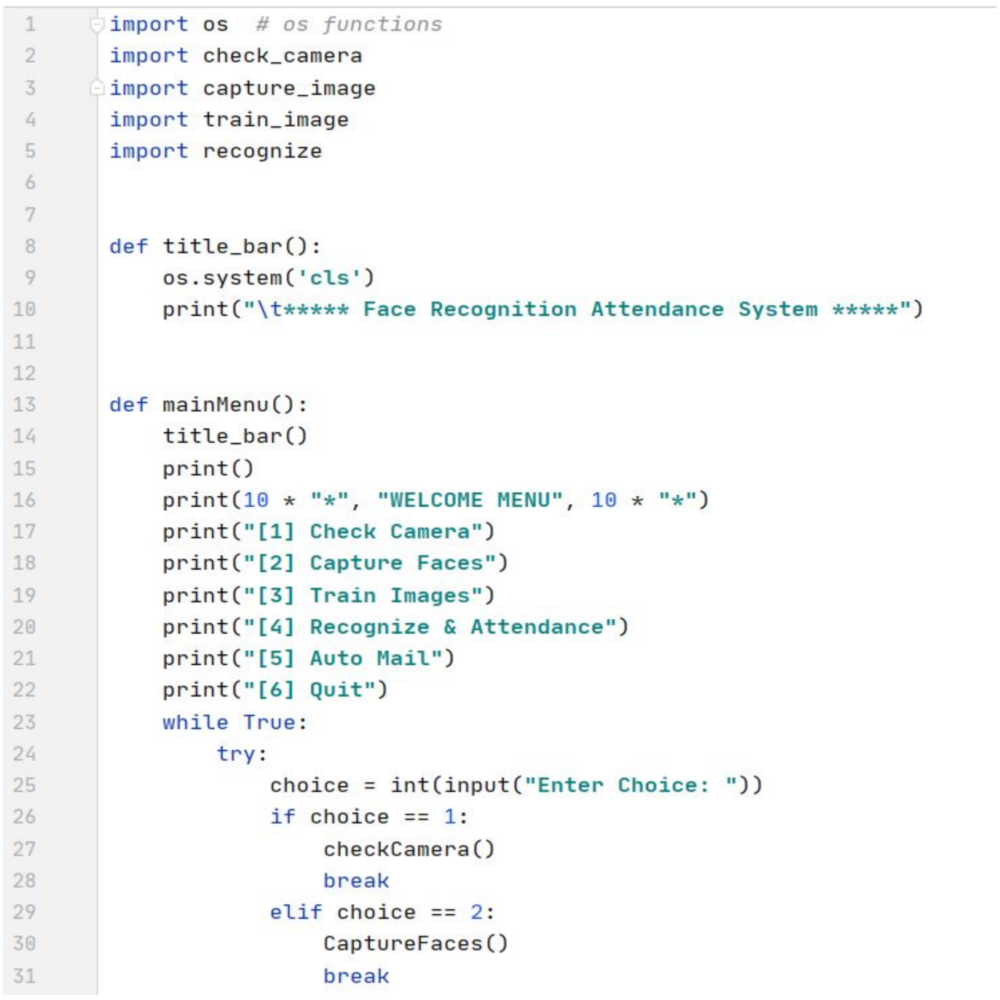
6 Code Implementation

All our code is written in Python language. Below are the important python codes which are the core of our implementation of this software engineering project.

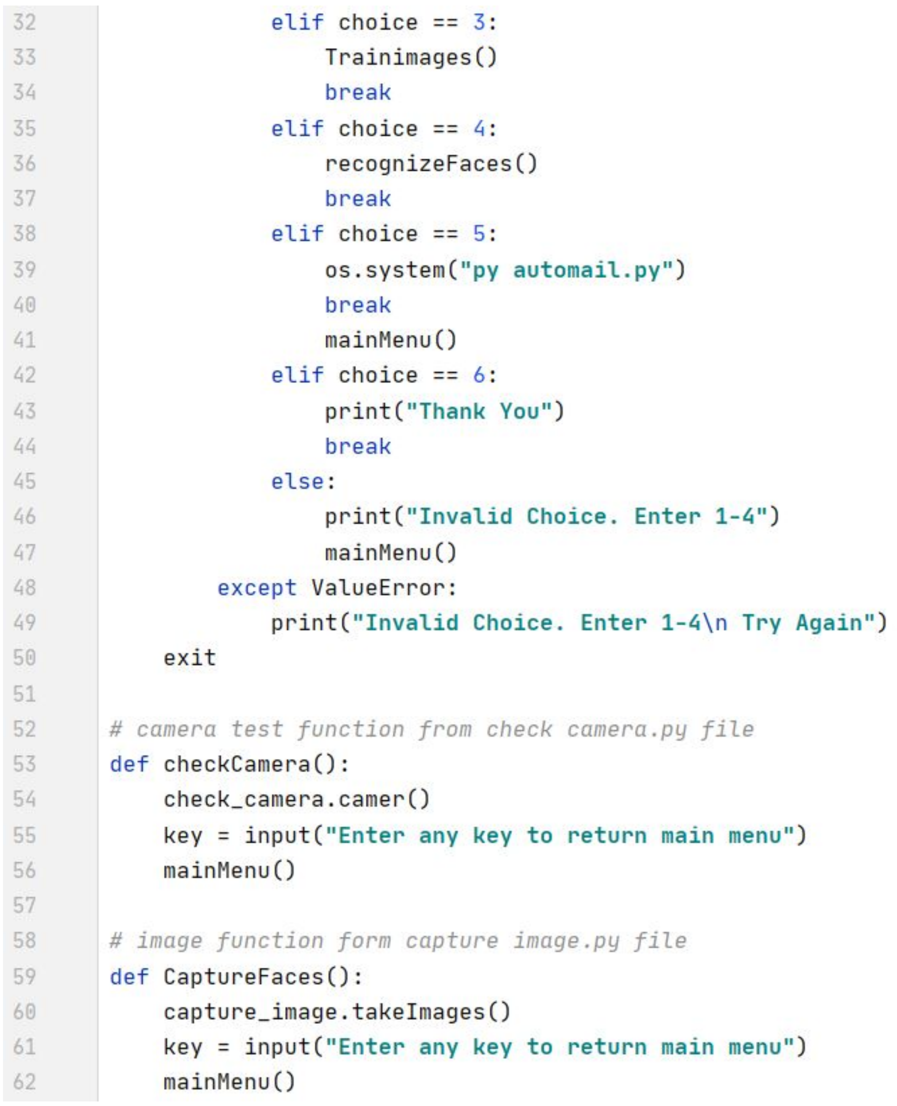
1. Dataset: Where all the face images are saved.
2. main.py: Main program file to run the program.
3. train_image.py: Train the captured images and work on datasets.
4. recognize.py: To recognize and mark attendance
5. automail.py: To send mail to a specific mail id.

1. main.py

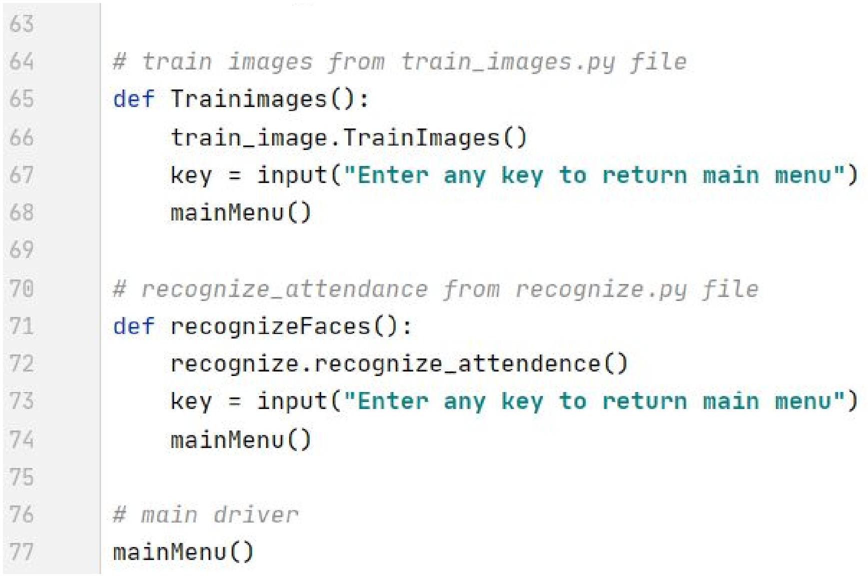
This is the main python code for our project that includes functions from other files and has all the options which are allowed in the user interface. Once this file is run, the user gets to choose whatever action has to be taken place, say whether it is to capture an image or to train an image, etc. The main function calls the other functions and specifies options to the user.



```python
import os  # os functions
import check_camera
import capture_image
import train_image
import recognize


def title_bar():
    os.system('cls')
    print("\t***** Face Recognition Attendance System *****")


def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")
    while True:
        try:
            choice = int(input("Enter Choice: "))
            if choice == 1:
                checkCamera()
                break
            elif choice == 2:
                CaptureFaces()
                break
```



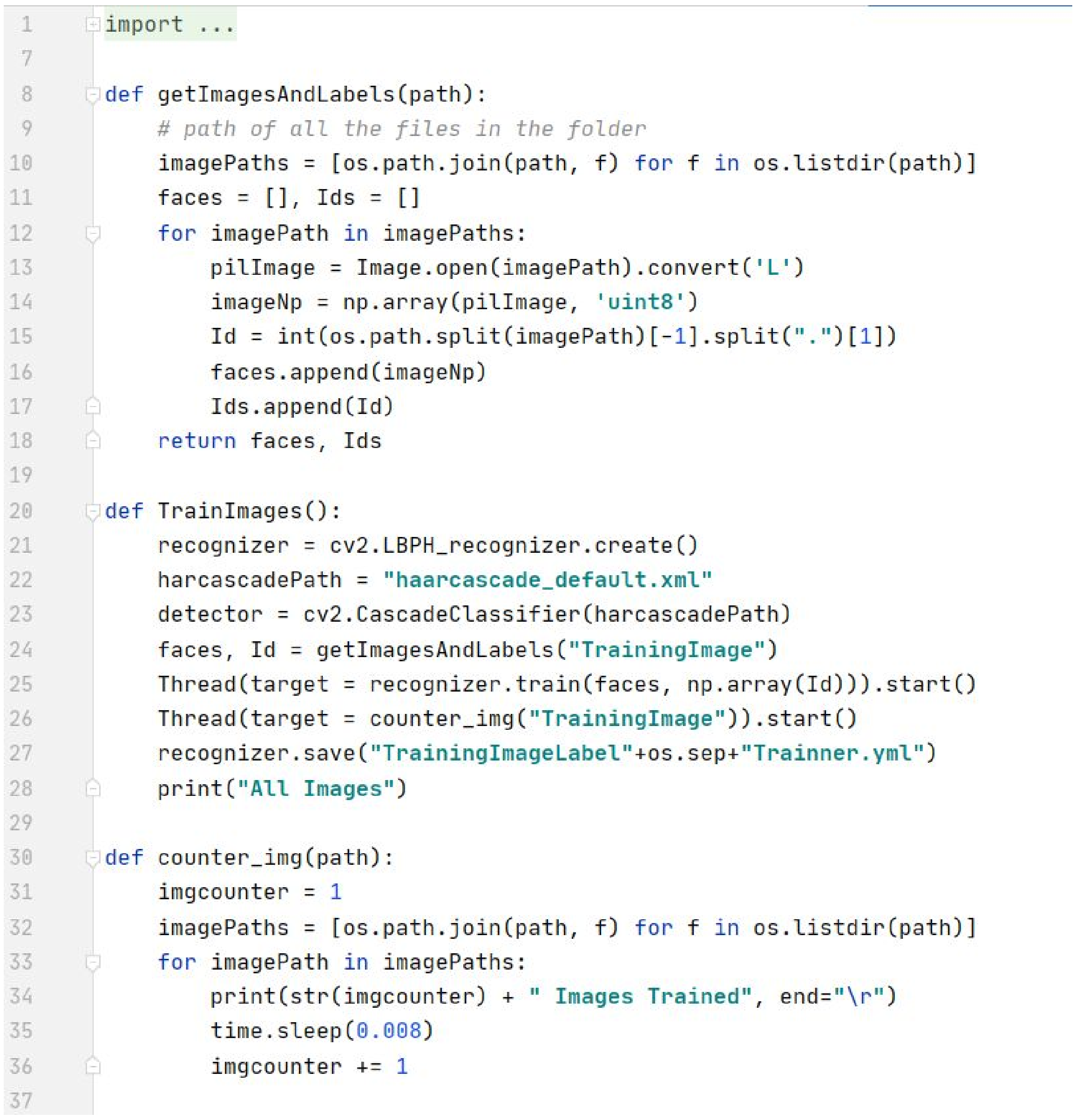
```python
            elif choice == 3:
                Trainimages()
                break
            elif choice == 4:
                recognizeFaces()
                break
            elif choice == 5:
                os.system("py automail.py")
                break
                mainMenu()
            elif choice == 6:
                print("Thank You")
                break
            else:
                print("Invalid Choice. Enter 1-4")
                mainMenu()
        except ValueError:
            print("Invalid Choice. Enter 1-4\n Try Again")
    exit

# camera test function from check camera.py file
def checkCamera():
    check_camera.camer()
    key = input("Enter any key to return main menu")
    mainMenu()

# image function form capture image.py file
def CaptureFaces():
    capture_image.takeImages()
    key = input("Enter any key to return main menu")
    mainMenu()
```



```python

# train images from train_images.py file
def Trainimages():
    train_image.TrainImages()
    key = input("Enter any key to return main menu")
    mainMenu()

# recognize_attendance from recognize.py file
def recognizeFaces():
    recognize.recognize_attendence()
    key = input("Enter any key to return main menu")
    mainMenu()

# main driver
mainMenu()
```

2.train_image.py

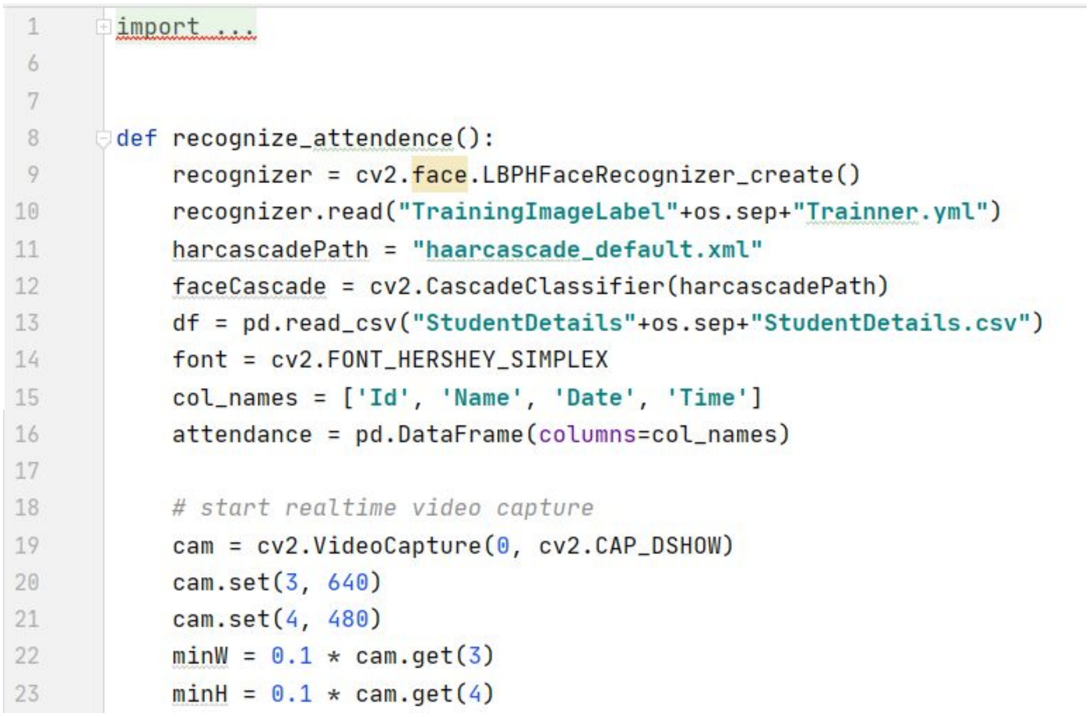
In this step, the image gets trained with database images and machine learning techniques. This is done by making use of a haar cascade classifier. Once the images are trained and the model is ready for the next step, the “image trained“ message appears on the screen.



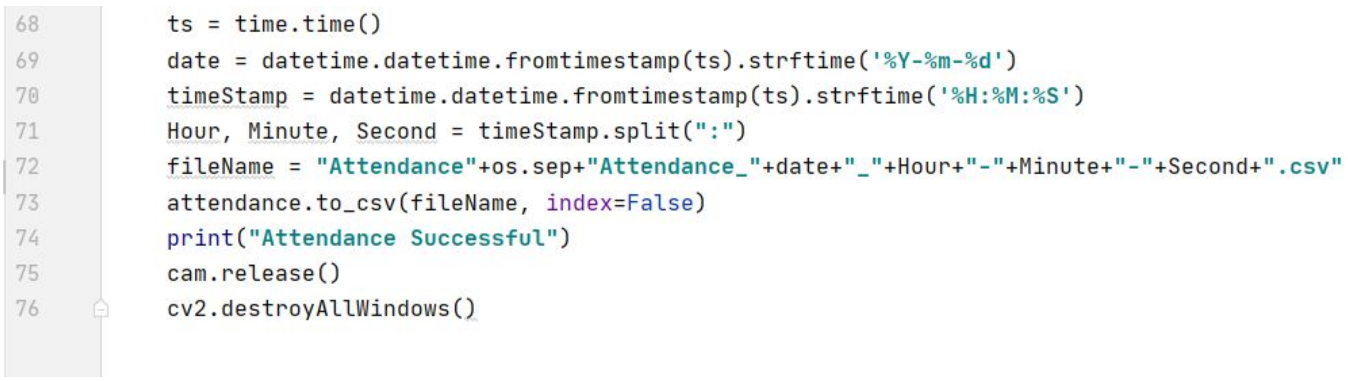
```python
import ...

def getImagesAndLabels(path):
    # path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = [], Ids = []
    for imagePath in imagePaths:
        pilImage = Image.open(imagePath).convert('L')
        imageNp = np.array(pilImage, 'uint8')
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids

def TrainImages():
    recognizer = cv2.LBPH_recognizer.create()
    harcascadePath = "haarcascade_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, Id = getImagesAndLabels("TrainingImage")
    Thread(target = recognizer.train(faces, np.array(Id))).start()
    Thread(target = counter_img("TrainingImage")).start()
    recognizer.save("TrainingImageLabel"+os.sep+"Trainner.yml")
    print("All Images")

def counter_img(path):
    imgcounter = 1
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    for imagePath in imagePaths:
        print(str(imgcounter) + " Images Trained", end="\r")
        time.sleep(0.008)
        imgcounter += 1
```

3. recognize.py

In this step, the faces are recognized with the help of the LBPH Face recognizer function. Along with this, the time and date also get recorded. If the function works successfully then the attendance is marked for the recognized face.



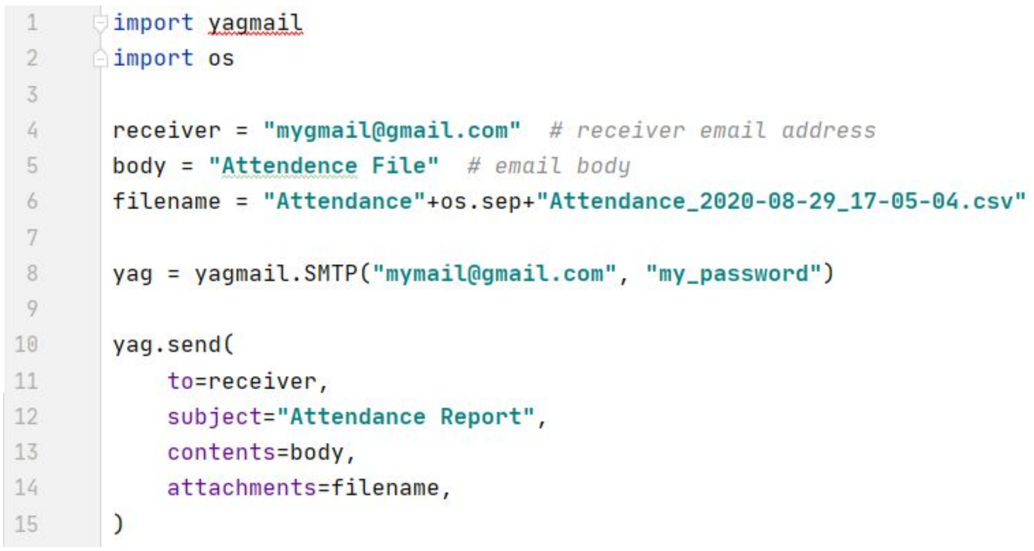
```python
import ...


def recognize_attendence():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read("TrainingImageLabel"+os.sep+"Trainner.yml")
    harcascadePath = "haarcascade_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    df = pd.read_csv("StudentDetails"+os.sep+"StudentDetails.csv")
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', 'Name', 'Date', 'Time']
    attendance = pd.DataFrame(columns=col_names)

    # start realtime video capture
    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cam.set(3, 640)
    cam.set(4, 480)
    minW = 0.1 * cam.get(3)
    minH = 0.1 * cam.get(4)
```



```python
    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)
    print("Attendance Successful")
    cam.release()
    cv2.destroyAllWindows()
```

4. automail.py

In this project, we have added an extra feature called auto-mail. It automatically sends the attendance file to specific mail (specified in the system model). The code for the same is given below:



```python
import yagmail
import os

receiver = "mygmail@gmail.com"  # receiver email address
body = "Attendence File"  # email body
filename = "Attendance"+os.sep+"Attendance_2020-08-29_17-05-04.csv"

yag = yagmail.SMTP("mymail@gmail.com", "my_password")

yag.send(
    to=receiver,
    subject="Attendance Report",
    contents=body,
    attachments=filename,
)
```

7 Testing

Testing is the technique for assessing a framework or its parts to search out whether it fulfills the ideal requirements or not. In direct words, testing is executing a framework to recognize any holes, mistakes, or missing requirements. It is a fundamental advance in a product designing cycle to develop any product. Software Testing is fundamental since we as a whole commit errors. A portion of those errors are irrelevant, however, some of them are costly. We have to check everything and anything we produce since things can generally turn out wrong.

Black box testing

In black-box testing, the structure of the program isn't thought about. It considers the usefulness of the application as it was. It is likewise called practical testing. In this sort of testing, the analyzers focus on utilitarian testing, that is, on giving known information and checking if the realized yield is acquired. This strategy is by and large followed while completing acknowledgment testing when the end-client isn't a product engineer however just a client.

White-box testing

White-box Testing is a kind of testing technique where the tester (a developer testing the application) knows about the framework internals. Its usage is straightforward. The goal is to guarantee that each line of the code is tried. The tester recognizes all consistent, plan, and typographical mistakes. The tester likewise needs to approve the interior structure of the thing viable alongside the yield.

Requirement Traceability Matrix

A traceability matrix is a document that associates any two-gauge records that require a many-to-numerous relationship to check the fulfillment of the relationship. It is utilized to follow the prerequisites and to check the current venture necessities are met.
Requirement Traceability Matrix, RTM catches all necessities proposed by the customer client and advancement group client and their detectability in a solitary archive conveyed after the life-cycle.In different words, it is a report that guides and follows client necessities with experiments. The fundamental reason for the Requirement Traceability Matrix is to see that all experiments are covered so no usefulness should miss while testing.

Table1: Functional Requirements Tables

Functional Requirement - ID	Functional Requirements Description
FR-1	Face Detection
FR-2	Capturing Faces and Training Model
FR-3	Storing images in the database
FR-4	Face Recognition
FR-5	Recording Attendance
FR-6	Emailing Attendance

Table2: Test Cases

Test Case - ID	Test Case Description
TC-1	Web camera detecting faces live
TC-2	Capturing multiple images
TC-3	Storing images with student details
TC-4	Encoding faces
TC-5	Training model with training images
TC-6	Recognizing faces registered in the database
TC-7	Marking attendance along with time
TC-8	Emailing the attendance along with details of timestamp

Table3: REQUIREMENTS TRACEABILITY MATRIX

Functional Requirements Document FRD			Test Case Document
FR - ID	**FR Use case**	**Priority**	**TC - ID**
FR -1	Face Detection	High	TC-1
FR -2	Capturing Faces and Training Model	High	TC-2 TC-4 TC-5
FR -3	Storing images in the database	High	TC-3
FR -4	Face Recognition	High	TC-6
FR -5	Recording Attendance	High	TC-7
FR -6	Emailing Attendance	Medium	TC-8

8 Results

This system is capable of detecting the faces from the captured image from HD Video to investigate and discover the face. Face detection determines when a picture, a face is found and it's being done by scanning the various image scales and extracting the precise patterns to discover the face. The entire range of pictures saved within the folder can actuate the numeric attendance throughout the class. The images within the folder are going to be matched by the recognition part to faces already held on and trained on the database images. This may facilitate confirmation of attendance for a specific student. It'll conjointly modify the lecturer to require full management of the class by calling out every student by name once it's needed.

Table 4: Detection and Recognition rate concerning face orientation

Face Orientations	Detection Rate	Recognition Rate
O⁰ (Frontal face)	98.7 %	95%
18°	80.0 %	78%
54°	59.2 %	58%
72°	0.00 %	0.00%
90°(Profile face)	0.00 %	0.00%

We performed a set of experiments to demonstrate the efficiency of the proposed method. 50 different images of 5 persons are used in the training set. The evaluation of the face recognition part of the system produced results that were not as expected. The evaluation showed that the recognition part can achieve approximately 70-80% recognition rate based on the image resolution. With a poor image resolution, there is a high chance the system will fail. However, to evaluate the system against images taken in controlled backgrounds, there is a high chance of excellent results.

The random sequence of input images when tested will most likely be the sequence from the output image. Although it does not change the overall percentage of recognition, it was not possible to tell the user at what percentage they could decide a face is a face, in other words, it hardly fails to recognize a face. The only way out was to carry out a test on all five input images and at least with three matches, the user can confirm a face, based on the output match displayed side by side. The performance of the system has impacted the reliability of the system. Because it is still an ongoing research area, the system will not be available for use at the end of the project. However, it can be used for research purposes by the supervisor and experimented with in a lecture room before approval.

9 Conclusion and Future Scope

The entire project has been developed from the requirements to a complete system alongside evaluation and testing. The system developed has achieved its aim and objectives. More careful analysis is required on a project intrinsically. The ways used may be combined with others to attain nice results. Completely different ways are enforced within the past in keeping with the literature review. The conclusion to set the parameters of this part of the system based on a very small class size was due to the failures obtained from the recognition part of the system. The size of the image is very important in face recognition as every pixel counts. The algorithm we've used would be analyzed with images of different sizes and these images would be showing students in a classroom setting with natural sitting positions showing faces of different sizes. The basic idea is, no matter how the face is turned, we should be able to center the eyes and mouth in roughly the same position in the image, thus recognizing the person and marking their attendance.

Login practicality would be enforced on the system for security functions. Data confidentiality is incredibly vital. At the beginning of every year, the photographs of the latest people can be taken and kept by the organization. Each person will have the right to be told regarding the employment of their faces for a face recognition attendance system. This should be in line with government laws on moral problems and information protection laws and rights. The individuals can consent to their pictures used for the aim of attendance.

In future work, we need to improve face detection effectiveness with the help of the interaction among our system, the scholars, and the faculty. On the other hand, our system is improved by desegregation video-streaming service and lecture archiving system, to give additional profound applications in the field of distance education, course management system (CMS), and support for college development (FD). We will improve this technique thus as we tend to run this system with additional than two students on a bench and permitting them to vary their positions.

- Can improve security by adding administrator login.
- Can use Neural Network for high accuracy.
- Can be used in a big factory or employee attendance.
- Can build on a fully web-based system.

Table 5: Work Division w.r.t. time

Sr. no.	Task	Date
1.	Installations of Pycharm ((python environment) along with OpenCV libraries	2 Sept
2.	Conduct a literature survey to identify the algorithms suitable for developing our model	8 Sept
3.	Developing Design and Architecture of our model, by resource implementation and initial prototyping	27 Sept
4.	Implementing necessary changes with given feedback to counter underlying problems and tackle necessities	10 Oct
5.	Review and Evaluation of our model with full-scale testing and corrections	19 Oct
6.	System testing to improve the features	5 Nov
7.	Complete documentation of the project with details in the Project Report	10 Nov

Acknowledgments

A big thank you to Prof.K Chandrasekaran for his support and guidance throughout this project. We gained so much knowledge about software engineering processes that we are very thankful to our professor. We also thank our friends for their help and encouraging words. Most of all a big thanks to God Almighty for his guidance throughout the course.
Thank you all for your encouragement and support!

References

[1] K.Senthamil Selvi et al, "Face Recognition Based Attendance Marking System" in International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, February-2014.

[2] CH. Vinod Kumar and Dr. K. Raja Kumar, "Face Recognition Based Student Attendance System with OpenCV" in International Journal of Advanced Technology and Innovative Research Volume. 08, issue.24, December-2016.

[3] Shervin EMAMI and Valentin Petrut SUCIU, "Facial Recognition using OpenCV", ResearchGate article March 2012.

[4] Divyarajsinh N. Parmar and Brijesh B. Mehta, "Face Recognition Methods & Applications", an article in International Journal of Computer Applications in Technology · January 2014.

[5] Sudhir Bussa, Shruti Bharuka, Ananya Mani, and Sakshi Kaushik, "Smart Attendance System using OPENCV based on Facial Recognition", Vol. 9 Issue 03, March-2020.

[6] Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, "Software Engineering – Concepts and Implementations".

[7] Ali Rehman Shinwari, Asadullah Jalali Balooch, Ala Abdulhakim Alariki, Sami Abduljalil Abdulhak, International Conference on Advanced Communication Technology (ICACT), "A Comparative Study of Face Recognition Algorithms under Facial Expression and Illumination", 2019 21st.

[8] Bobby R. BruceEmail author Jonathan M. AitkenEmail author Justyna PetkeEmail author, "Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV", LNCS, volume 9962.

[9] Sudhir Bussa , Ananya Mani , Shruti Bharuka , Sakshi Kaushik, Department of Electronics and Telecommunication, Bharati Vidyapeeth, University, College of Engineering, Dhankawadi, "Smart Attendance System using OPENCV based on Facial Recognition", Volume 09, Issue 03, March 2020.

[10] Mamata S.Kalas, Associate Professor, Department Of It, Kit's College Of Engg., Kolhapur, "Real-time face Detection And Tracking Using Opencv", 5th & 6th February 2014.

[11] Saravanan P. and R.Padmavathi, “An Innovative RTM Providing Test Maintenance in An Automated Scripting New Frameworks”, Volume 2, Issue 5, May – 2017.