# Institute of Information Technology & Management

# New Delhi – 1100 58

**Batch (2018 - 2021)**


# ATTENDANCE MANAGEMENT SYSTEM

## A MACHINE LEARNING PROJECT

**Bachelor of Computer Applications**

**Guide:**

Leena Gupta

(Associate Professor)

**Submitted by:**

ARYAN SAXENA

(00721102018)

# Certificate

I, Aryan Saxena (00721102018) certify that the Major Project Report (BCA-356) entitled "_ATTENDANCE MANAGEMENT SYSTEM_" is done by me and it is an authentic work carried out by us at IITM. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

1. Signature of the Student

Date:

Certified that the Project Report (BCA-356) entitled "ATTENDANCE MANAGEMENT SYSTEM_" done by the above student is completed under my guidance.

Signature of the Guide
Date:
Name of the Guide: LEENA GUPTA
Designation:

Countersigned

Prof. (Dr. Sudhir K. Sharma)
HOD-IT

Prof.(Dr.Prerna Mahajan)
Director

# TABLE OF CONTENTS

# Synopsis

## 1. Title of Project

Attendance Management System for Institute of Information and Technology and Management.

## 2. Introduction of the Project

In the existing system for attendance making systems work manually which sometimes causes mistakes and the accuracy level wasn't able to reach 100%. Sometimes students make proxies for other students on their behalf though students are missing from the classes. Hence we will propose a system through which the attendance will be taken by detecting their faces in the dataset and then the algorithm will work in the backend to recognize the name and id of the particular face which is shown in the camera and then the timestamp will be populated in the tables.

## 3. Objective

The main objectives of this project are:

- The risk of manual errors is ejected.
- Updating and training the dataset is quite easy.
- Backup data can be easily generated.
- Reduced wastage of time.
- Increase accuracy and reliability.
- Providing information to faculty members and the students becomes much easier.

Hence, it is a flexible approach to manage everything in a computerized way which can make things much sorted and efficient .

## 4. Scope

Today is the era of computers. This software project solves all the problems of interaction. The main objective of developing this project is to save time and money. The proposed system provides the following features on different tasks:

- A system like this will make the flow of process easy .
- Basic information can be represented in a graphical way, which helps the user to understand when he/she can apply for their own use.
- It stimulates the user a new experience
- The chances of proxy will be diminished.
- The accuracy will be at its peak.
- Real time information publishing through system alerts.
- To save the environment by using paper free work.
- The same application can be used for the students and faculty members.

# 1. SRS Requirements

## 5.1 Functional Module

- **Add a student (admin):** The system is under the admin/teacher who supervises and views the details of the students. It adds and manages the students as well as the information in this system.

- **Training Module:** The user can change and upload their photos for training of the machine.

- **User Module:** The module allows the user to scan their faces and if it recognizes them and populate the time stamp in the tables.

### 5.2 Non Functional Module

None

## 6. Tools and Platforms

### 6.1 Hardware specifications

- Preprocessor – Intel i3 and beyond
- RAM- 6GB or More
- Hard Disk- 500GB

### 6.2 Software specifications

- Windows 10
- *SQL workbench
- Python
- Opencv

## 7. Methodology

### 7.1 SDLC Model to be used

Incremental Model

**Justification for selection of model:**

The main aim of using this model is to add more features in the existing modules to increase project reliability and usability. Using this model we can adapt to the changing requirements of the user which helps in developing the project in relatively small amount of time. The next increment implements user suggestions plus some additional requirements in the previous increment. The process is repeated until the project is completed.

**Advantages –**

1. Generates working Software quickly and early during the software life cycle.
2. More Flexible-less costly to change Scope and Requirements.
3. Easier to test and debug during a smaller iteration.
4. User can respond to each built

## 7.2 Data Collection Methods

- **Observation Method**

  An observation is a data collection method, by which you gather knowledge of the researched phenomenon through making observations of the phenomena, as and when it occurs. You should aim to focus your observations on human behaviour, the use of the phenomenon and human interactions related to the phenomenon. You can also make observations on verbal and nonverbal expressions. In making and documenting observations, you need to clearly differentiate your own observations from the observations provided to you by other people. The range of data storage genre found in Archives and Collections, is suitable for documenting observations e.g. audio, visual, textual and digital including sub-genres of note taking, audio recording and video recording.

# 8. Future Scope

There are also few features which can be integrated with this system to make it more flexible. Below list shows the future points to be consider.

- Eye scanner.
- Different UI phases with updates.
- Mobile application for the quick sneak

# CHAPTER – 1: INTRODUCTION

## 1. Description of Organization

### 1.1 Introduction

In the existing system for attendance making systems work manually which sometimes causes mistakes and the accuracy level wasn't able to reach 100%. Sometimes students make proxies for other students on their behalf though students are missing from the classes. Hence we will propose a system through which the attendance will be taken by detecting their faces in the dataset and then the algorithm will work in the backend to recognize the name and id of the particular face which is shown in the camera and then the timestamp will be populated in the tables.

### 1.2 Objective

The main objectives of this project are:

- The risk of manual errors is ejected.
- Updating and training the dataset is quite easy.
- Backup data can be easily generated.
- Reduced wastage of time.
- Increase accuracy and reliability.
- Providing information to faculty members and the students becomes much easier.

Hence, it is a flexible approach to manage everything in a computerized way which can make things much sorted and efficient.

### 1.3 Functions (or modules)

- **Add a student (admin):** The system is under the admin/teacher who supervises and views the details of the students. It adds and manages the students as well as the information in this system.

- **Training Module:** The user can change and upload their photos for training of the machine.

- **User Module:** The module allows the user to scan their faces and if it recognizes them and populate the time stamp in the tables.

## 1.3 Software Requirement Specifications

### 1.3.1 Introduction –

In the existing system for attendance making systems work manually which sometimes causes mistakes and the accuracy level wasn't able to reach 100%. Sometimes students make proxies for other students on their behalf though students are missing from the classes. Hence we will propose a system through which the attendance will be taken by detecting their faces in the dataset and then the algorithm will work in the backend to recognize the name and id of the particular face which is shown in the camera and then the timestamp will be populated in the tables.

## 1.4 Scope

Today is the era of computers. This software project solves all the problems of interaction. The main objective of developing this project is to save time and money. The proposed system provides the following features on different tasks:

- A system like this will make the flow of process easy .
- Basic information can be represented in a graphical way, which helps the user to understand that when he/she can apply for their own use.
- It stimulates the user a new experience
- The chances of proxy will be diminished.
- The accuracy will be at its peak.
- Real time information publishing through system alerts.
- To save the environment by using paper free work.
- The same application can be used for the students and faculty members.

## 1.5 Hardware Interfaces

- Preprocessor – Intel i3 and above.
- RAM- 6GB or More
- Hard Disk- 500GB

## Software Interfaces

- Windows / Linux / Mac
- My SQL
- Python
- Opencv

## 1.6 References


- https://opencv.org/
- https://www.python.org/doc/
- https://www.youtube.com/
- https://www.javatpoint.com/python-tutorial


## 1.7 Software System Attributes

- Reliability
- Availability
- Security
- Maintainability
- Portability

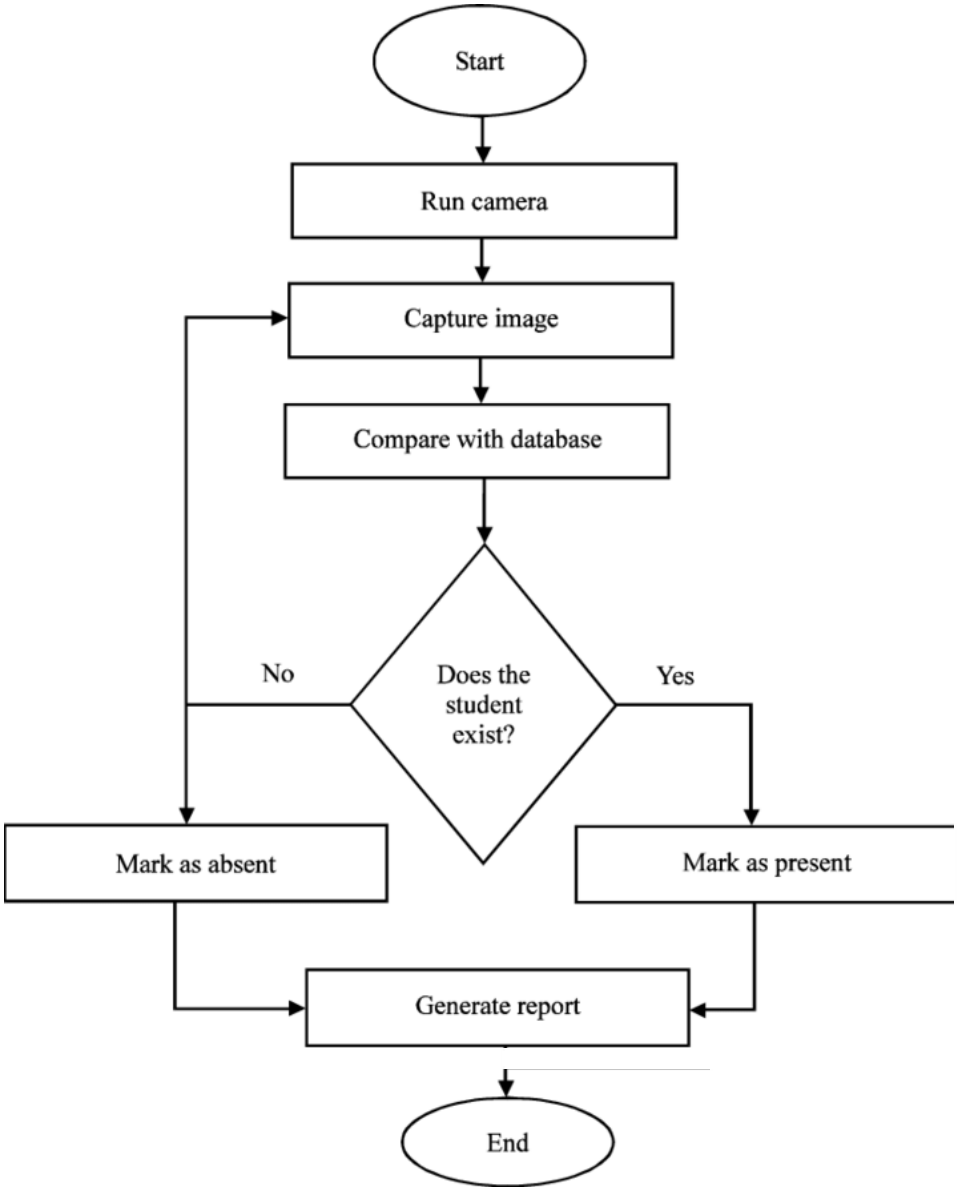
---

# CH-2&3 System Design And Requirement Analysis

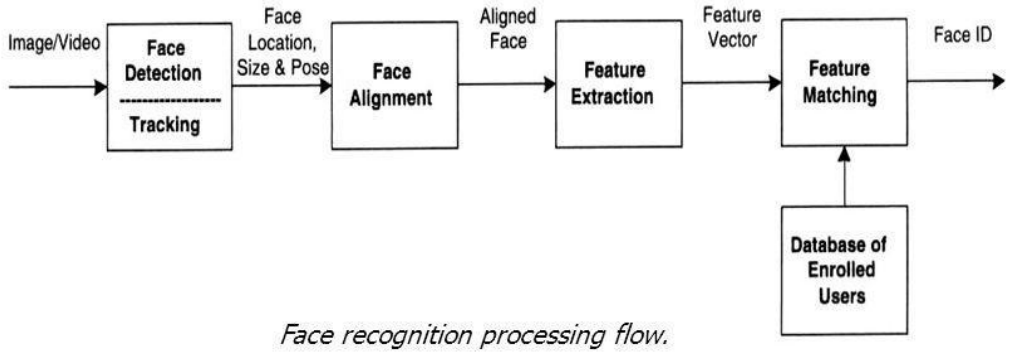
## 2.1 Physical Design

The proposed system Attendance Management System through face detection should be designed in such a way that students whose face is detected in the camera will analyze the points on the face and then show the credentials of the student. Then the credentials matching in the excel sheet will be populated with the particular timestamp. The records should be modified by only administrators and no one else. The user should always be in control of the application and not vice versa. The user interface should be consistent so that the user can handle the application with ease and speed. The application should be visually, conceptually clear.

### 2.1.1 Data Flow Diagram



## 2.1.2 Face Recognition process flow




*Face recognition processing flow.*

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems can be used to identify people in photos, video, or in real-time. Law enforcement may also use mobile devices to identify people during police stops.

But face recognition data can be prone to error, which can implicate people for crimes they haven't committed. Facial recognition software is particularly bad at recognizing African Americans and other ethnic minorities, women, and young people, often misidentifying or failing to identify them, disparately impacting certain groups.

Face recognition systems use computer algorithms to pick out specific, distinctive details about a person's face. These details, such as distance between the eyes or shape of the chin, are then converted into a mathematical representation and compared to data on other faces collected in a face recognition database. The data about a particular face is often called a face template and is distinct from a photograph because it's designed to only include certain details that can be used to distinguish one face from another.

Some face recognition systems, instead of positively identifying an unknown person, are designed to calculate a probability match score between the unknown person and specific

face templates stored in the database. These systems will offer up several potential matches, ranked in order of likelihood of correct identification, instead of just returning a single result.

Face recognition systems vary in their ability to identify people under challenging conditions such as poor lighting, low quality image resolution, and suboptimal angle of view (such as in a photograph taken from above looking down on an unknown person).

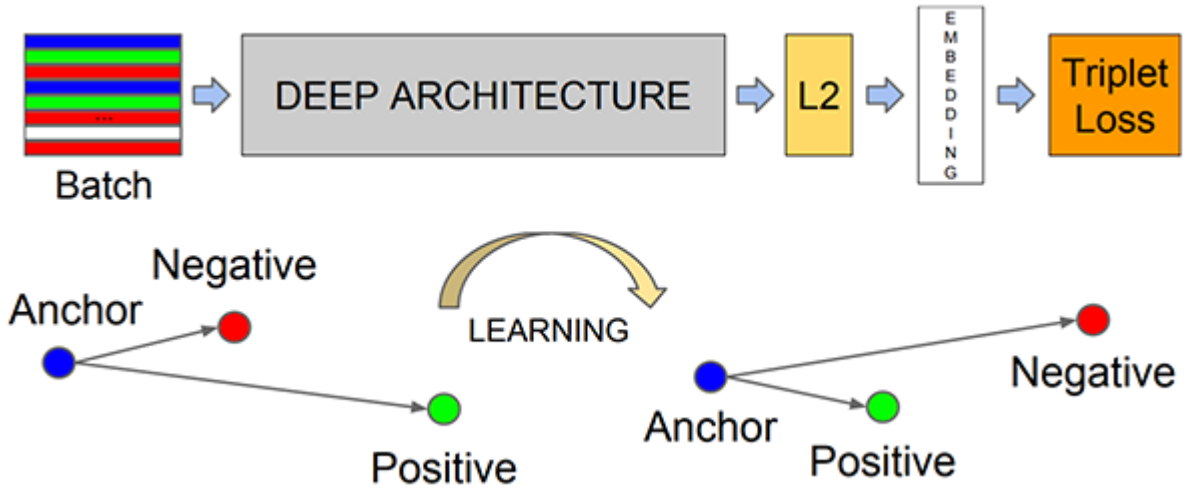
When it comes to errors, there are two key concepts to understand:

A "false negative" is when the face recognition system fails to match a person's face to an image that is, in fact, contained in a database. In other words, the system will erroneously return zero results in response to a query.

A "false positive" is when the face recognition system does match a person's face to an image in a database, but that match is actually incorrect. This is when a police officer submits an image of "Joe," but the system erroneously tells the officer that the photo is of "Jack."

When researching a face recognition system, it is important to look closely at the "false positive" rate and the "false negative" rate, since there is almost always a trade-off. For example, if you are using face recognition to unlock your phone, it is better if the system fails to identify you a few times (false negative) than it is for the system to misidentify other people as you and lets those people unlock your phone (false positive). If the result of a misidentification is that an innocent person goes to jail (like a misidentification in a mugshot database), then the system should be designed to have as few false positives as possible.

### 2.1.3 Open CV internal learning structure



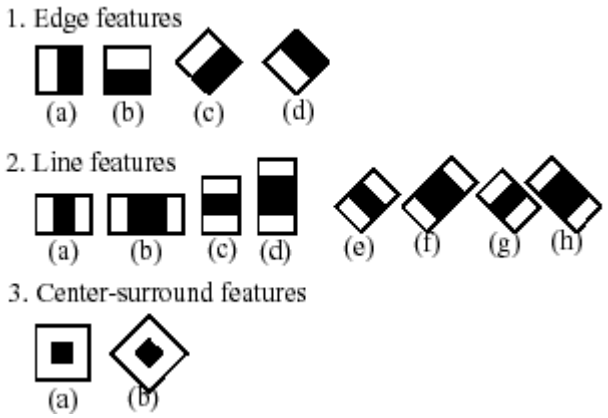

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

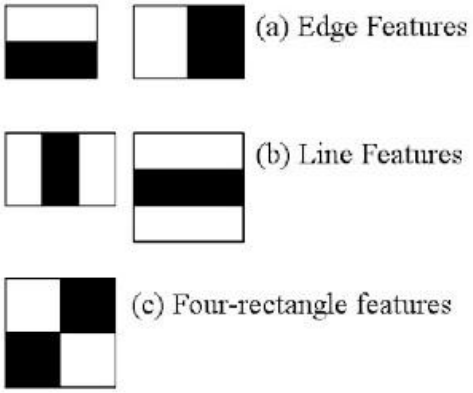
Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

## 2.1.4 Harr Cascade Classifier Algorithm structure

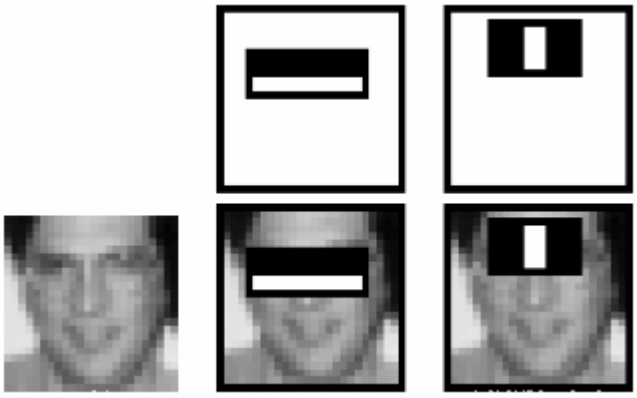



Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.




Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.



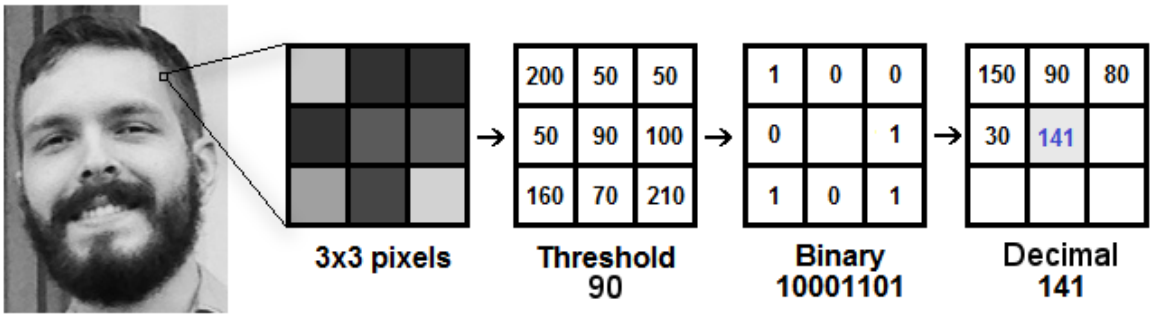
For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region

## 2.1.5 LBPH ALGORITHM



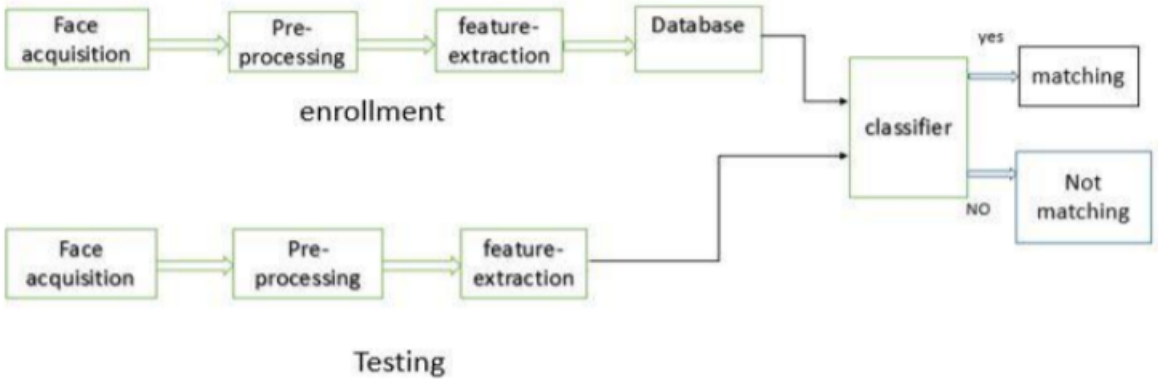

Local Binary Patterns Histogram algorithm was proposed in 2006. It is based on local binary operator. It is widely used in facial recognition due to its computational simplicity and discriminative power.

The steps involved to achieve this are:

- creating dataset
- face acquisition
- feature extraction
- classification

The LBPH algorithm is a part of opencv.

## Steps

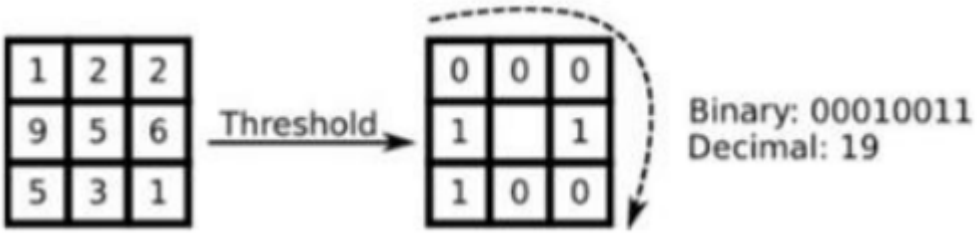


- Suppose we have an image having dimensions N x M.
- We divide it into regions of same height and width resulting in m x m dimension for every region.

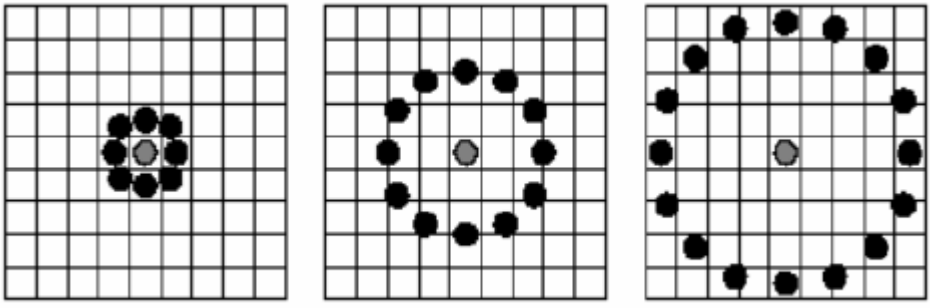
- Local binary operator is used for every region. The LBP operator is defined in window of 3x3.

here '(Xc,Yc)' is central pixel with intensity 'Ic'. And 'In' being the intensity of the the neighbor pixel

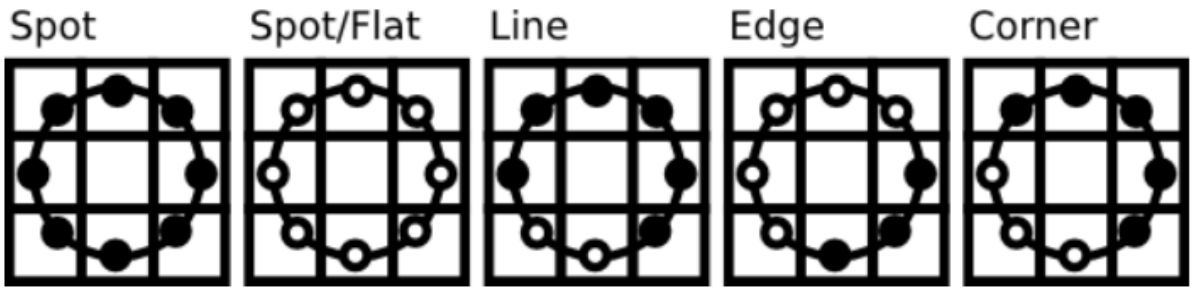
- Using median pixel value as threshold, it compares a pixel to its 8 closest pixels using this function.
- If the value of neighbor is greater than or equal to the central value it is set as 1 otherwise it is set as 0.
- Thus, we obtain a total of 8 binary values from the 8 neighbors.
- After combining these values we get a 8 bit binary number which is translated to decimal number for our convenience.
- This decimal number is called the pixel LBP value and its range is 0-255.



- Later it was noted that a fixed neighborhood fails to encode details varying in scale .The algorithm was improved to use different number of radius and neighbors , now it was known as circular LBP.



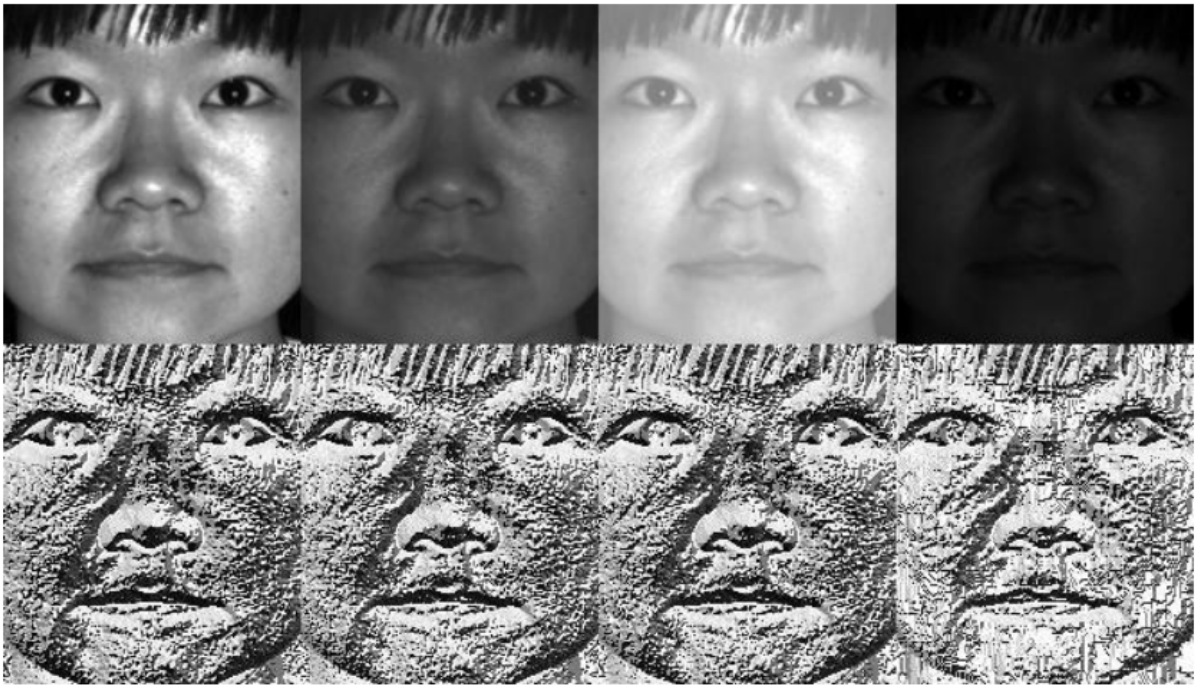

- The idea here is to align an arbitrary number of neighbors on a circle with a variable radius. This way the following neighborhoods are captured:



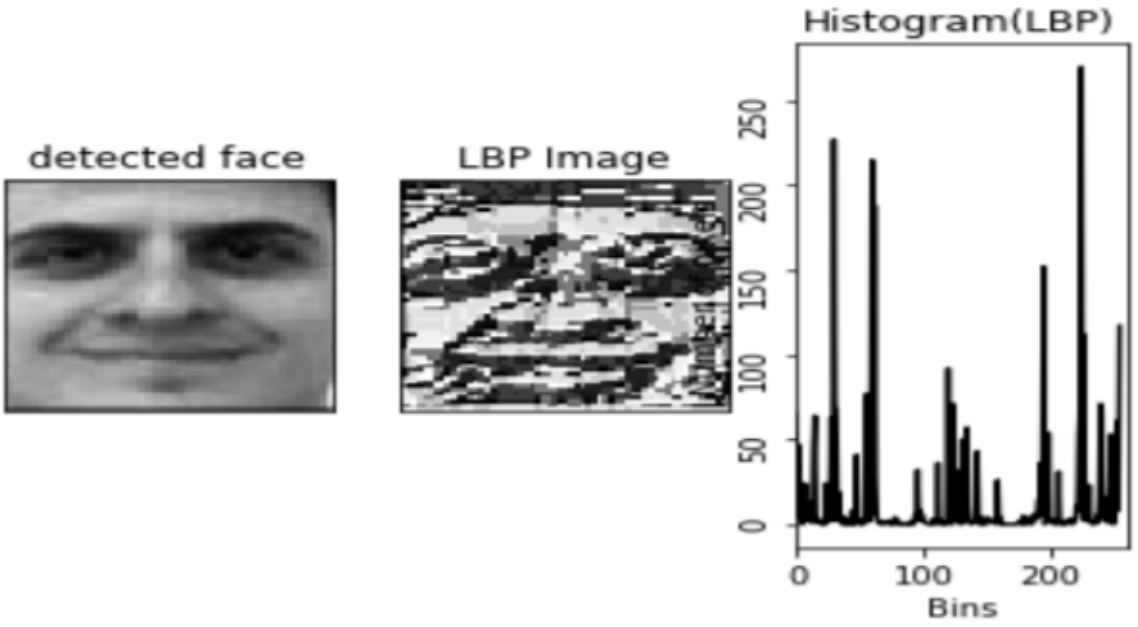

- For a given point (Xc,Yc) the position of the neighbor (Xp,Yp), p belonging to P can be calculated by:

here R is radius of the circle and P is the number of sample points.

- If a points coordinate on the circle doesn't correspond to image coordinates, it get's interpolated generally by bilinear interpolation
- The LBP operator is robust against monotonic gray scale transformations.



- After the generation of LBP value histogram of the region is created by counting the number of similar LBP values in the region.
- After creation of histogram for each region all the histograms are merged to form a single histogram and this is known as feature vector of the image.



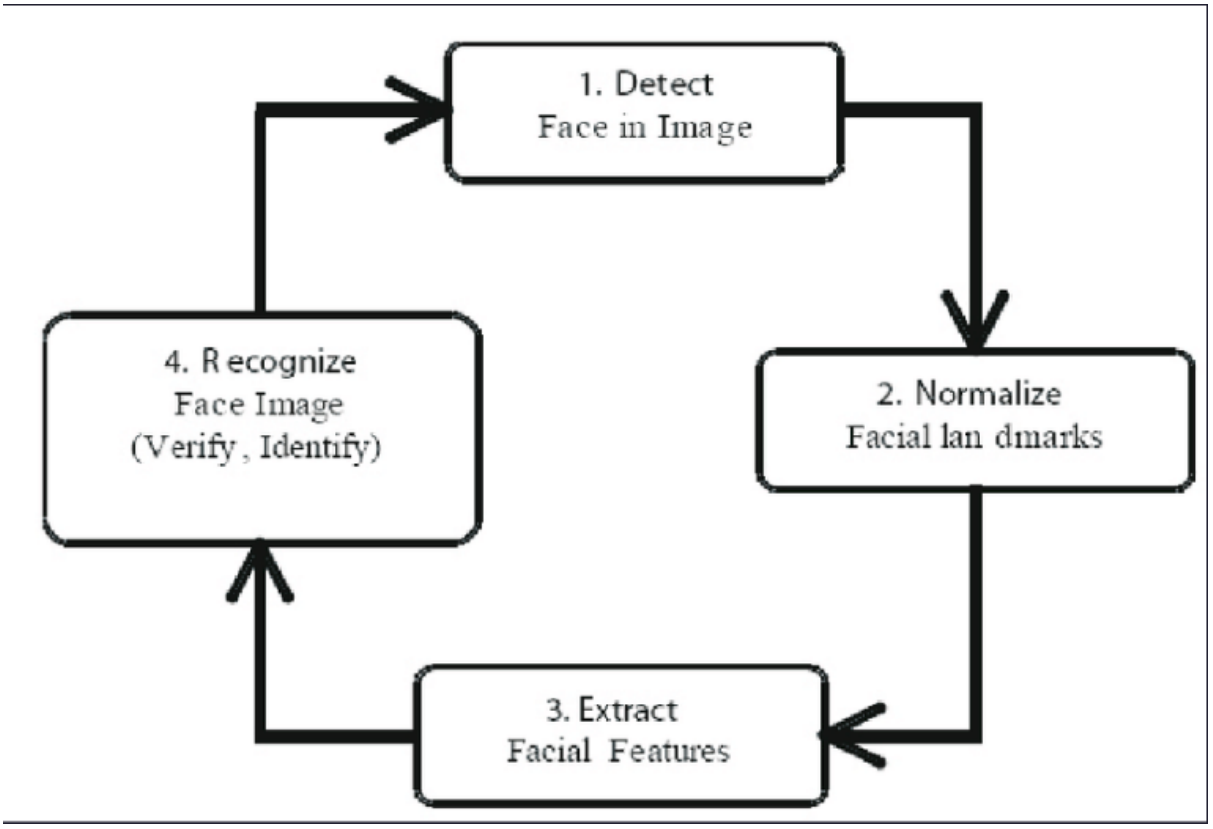
- Now we compare the histograms of the test image and the images in the database and then we return the image with the closest hitogram.
  ( This can be done using many techniques like euclidean distance, chi-square, absolute value etc )
- The Euclidean distance is calculated by comparing the test image features with features stored in the dataset.The minimum distance between test and original image gives the matching rate.

$$d(a,b) = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2}$$

- As an output we get an ID of the image from the database if the test image is recognised.


LBPH can recognise both side and front faces and it is not affected by illumination variations which means that it is more flexible.

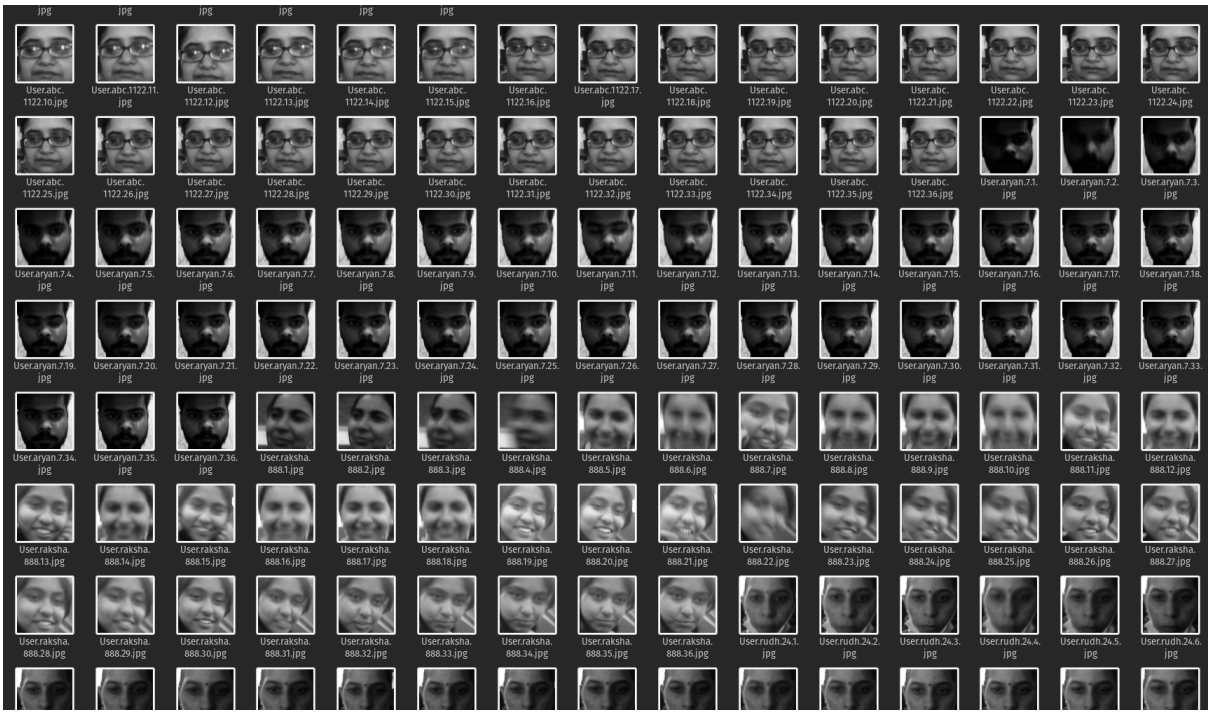
## 2.2 Data Set Storage and Record Storage (DATABASE)




## 3.1 Data Set Storage and Record Storage (DATABASE)

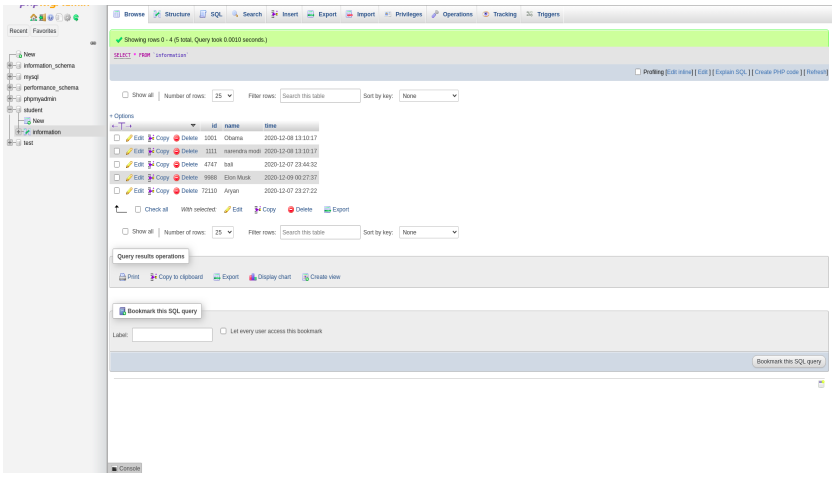
### 3.1.1 Data Set

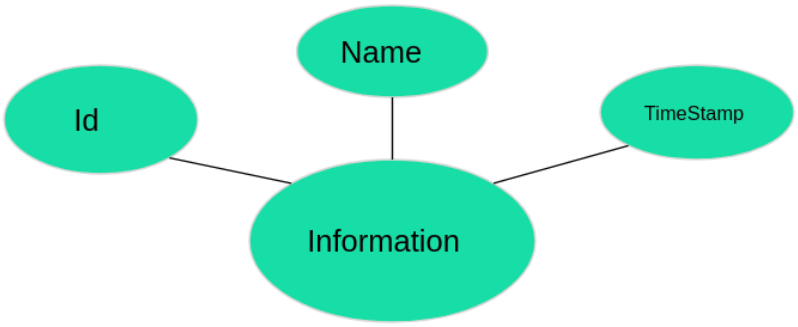
- All the data (images of users/students) are stored in the folders with the name given as their credentials.
- A data set contains pictures of the user which is generated either at the time of adding a new user or we can simply store the image in the folder.
- Then the data will be collected with the help of different libraries in python.
- There will be different folders for different users.



## 3.2.2 Record Storage

- The record of the user whether he/she is present or the face has been detected or not will be stored in an excel sheet.
- There will be different columns populated by the credentials of the user and the face detection timestamp will be noted.
- We will be using python libraries Tkinter and open xl for this.

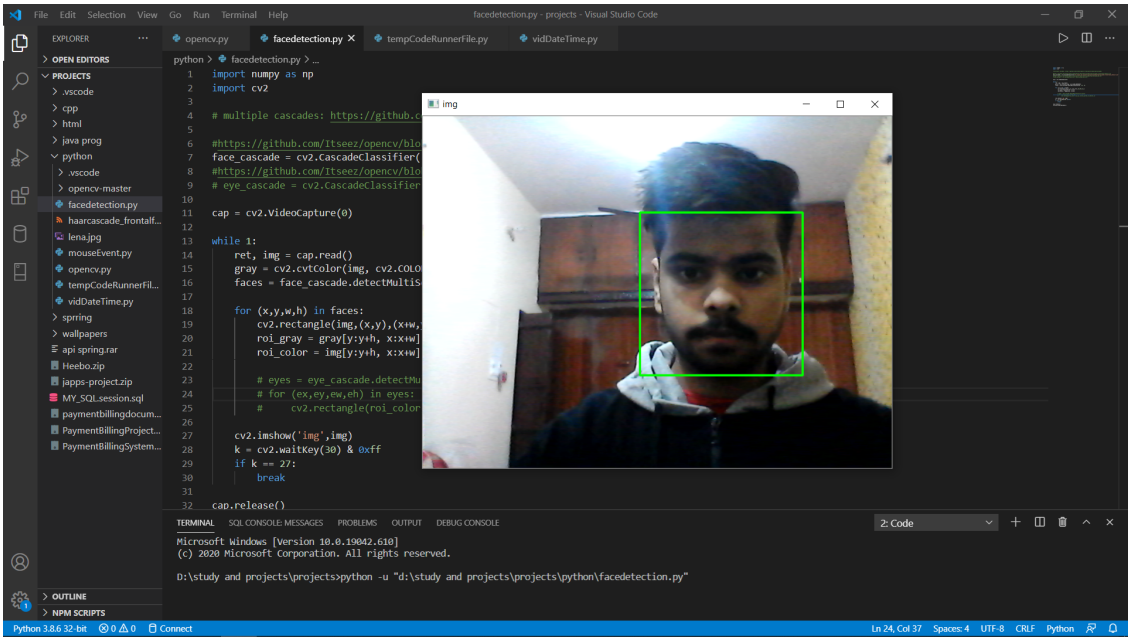


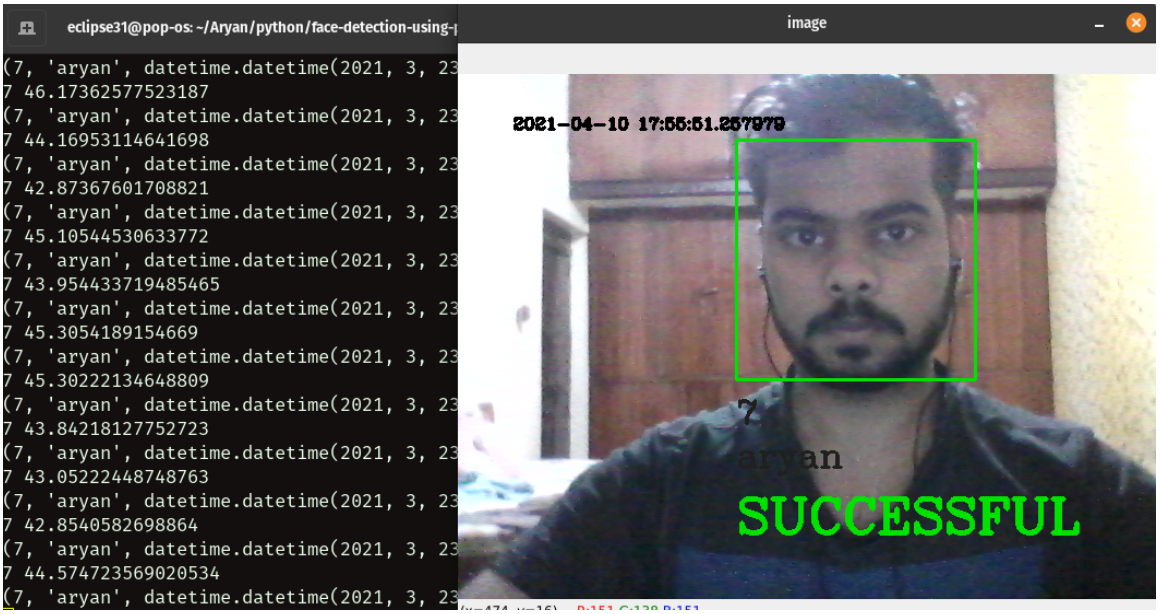



**ER DIAGRAM FOR STUDENT DATABASE**

## 3.3 Interface Design

- The interface design for input and output will be simply a window that will be monitoring the faces.
- First, it will detect faces and display a rectangular square on the outline of the face.
- If the face is recognized from the data set then it will display a green rectangle with a name hovering on it, and if not then a red rectangle.
- Then the credentials will be automatically populated in the sheet.



# Chapter-4: System Development

**DATA CREATOR**

```
import cv2
import numpy as np
import mysql.connector

print("My sql connector version  "+
format(mysql.connector.__version__))
mydb=mysql.connector.connect(host="localhost",user="root",pass
wd="",database="student")
mycursor=mydb.cursor()

mycursor.execute("select * from information ")
for row in mycursor:
    print(row)

# input name and roll no
name=input('enter user name  ')
id=input('enter rollno ')
ob=(id,name)
cmd="insert into information(id,name)values(%s,%s)"
mycursor.execute(cmd,ob)
mydb.commit()
mydb.close()

# classifier and vid capture
facedetect = cv2.CascadeClassifier("/home/eclipse/aryan/vs
code/projects/python/faceRecognition/haarcascade_frontalface_a
lt.xml")
cam=cv2.VideoCapture(0)
sampleNum=0

while 1 :
   ret,img=cam.read()
   grey=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   faces=facedetect.detectMultiScale(grey,1.3,5)

   for(x,y,h,w) in faces:
       sampleNum=sampleNum+1
```

```
cv2.imwrite("dataSet/User."+str(name)+"."+str(id)+"."+str(samp
leNum)+".jpg",grey[y:y+h,x:x+w])
        img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,225,0),2)
        font=cv2.FONT_HERSHEY_COMPLEX
        img=cv2.putText(img, name, (x,y), font, 1, (0 , 225,
0), 1, cv2.LINE_AA )
        cv2.waitKey(100)

    cv2.imshow("frame",img)
    cv2.waitKey(1)
    if(sampleNum>35):
       break
cam.release()
cv2.destroyAllWindows()

print("code successful")
```

**TRAINING DATA**

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

recognizer =  cv2.face.LBPHFaceRecognizer_create()
detector=
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

path='dataSet'

def getImagesWithID(path):
    print(" \n\n TRAINING STARTED\n\n")

    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    print(imagePaths)
    faces=[]
    IDs=[]
    for imagePath in imagePaths:
        faceImg=Image.open(imagePath).convert('L')
        faceNp=np.array(faceImg,'uint8')
        ID=int(os.path.split(imagePath)[-1].split(".")[2])
        faces.append(faceNp)
        print(faces)
        print(ID)
        IDs.append(ID)
        print(IDs)
        cv2.imshow("training in progress",faceNp)
        cv2.waitKey(100)
    return IDs,faces

Ids,faces = getImagesWithID(path)
print(recognizer)
recognizer.train(faces,np.array(Ids))
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()
```

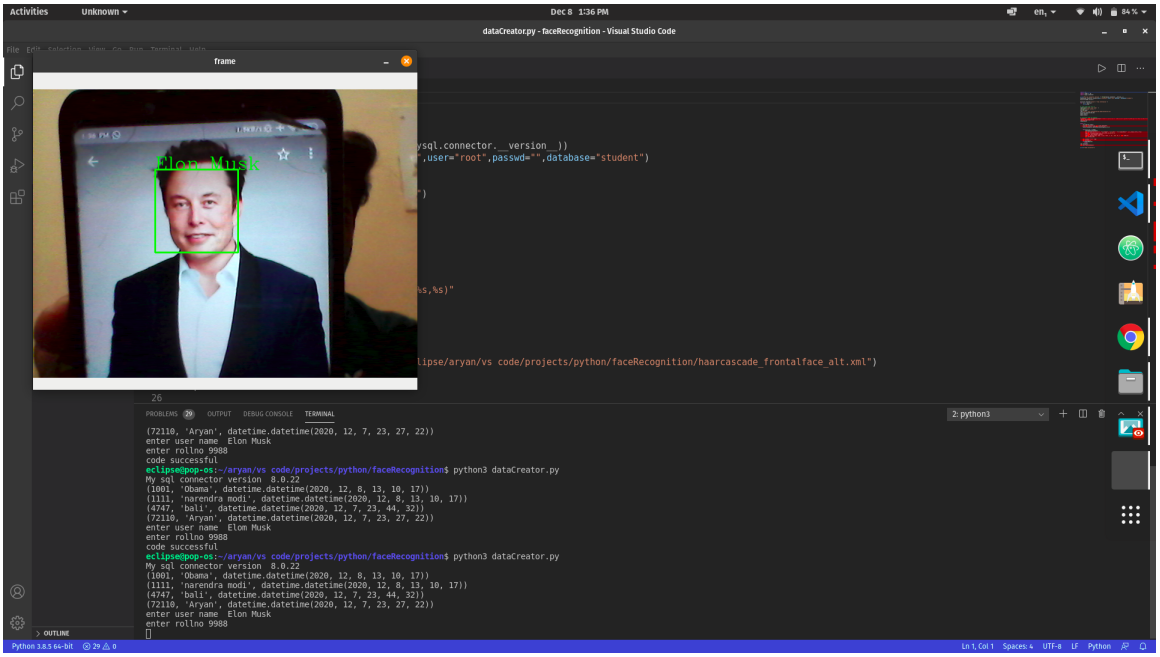
**RECOGNITION**

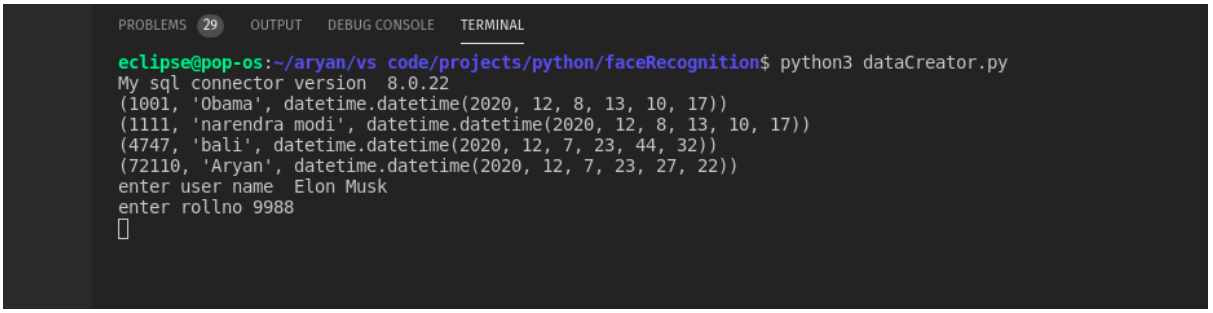
```
import cv2
import datetime
import mysql.connector

# recognizer and classifier and vid capture
facedetect =
cv2.CascadeClassifier("/home/eclipse/aryan/vs
code/projects/python/faceRecognition/haarcascade_frontal
face_alt.xml")
recognizer =  cv2.face.LBPHFaceRecognizer_create()
recognizer.read("/home/eclipse/aryan/vs
code/projects/python/faceRecognition/recognizer/training
Data.yml")
font=cv2.FONT_HERSHEY_COMPLEX
cam=cv2.VideoCapture(0)

# my sql connector
print("My sql connector version  "+
format(mysql.connector.__version__))
mydb=mysql.connector.connect(host="localhost",user="root
",passwd="",database="student")
mycursor=mydb.cursor()
# get function
def getProfile(id):
   cmd= """select * from information where id = %s"""
   mycursor.execute(cmd, (id,))
   result = mycursor.fetchall()
   profile=None
   for row in result:
       profile=row
       print(profile)
```

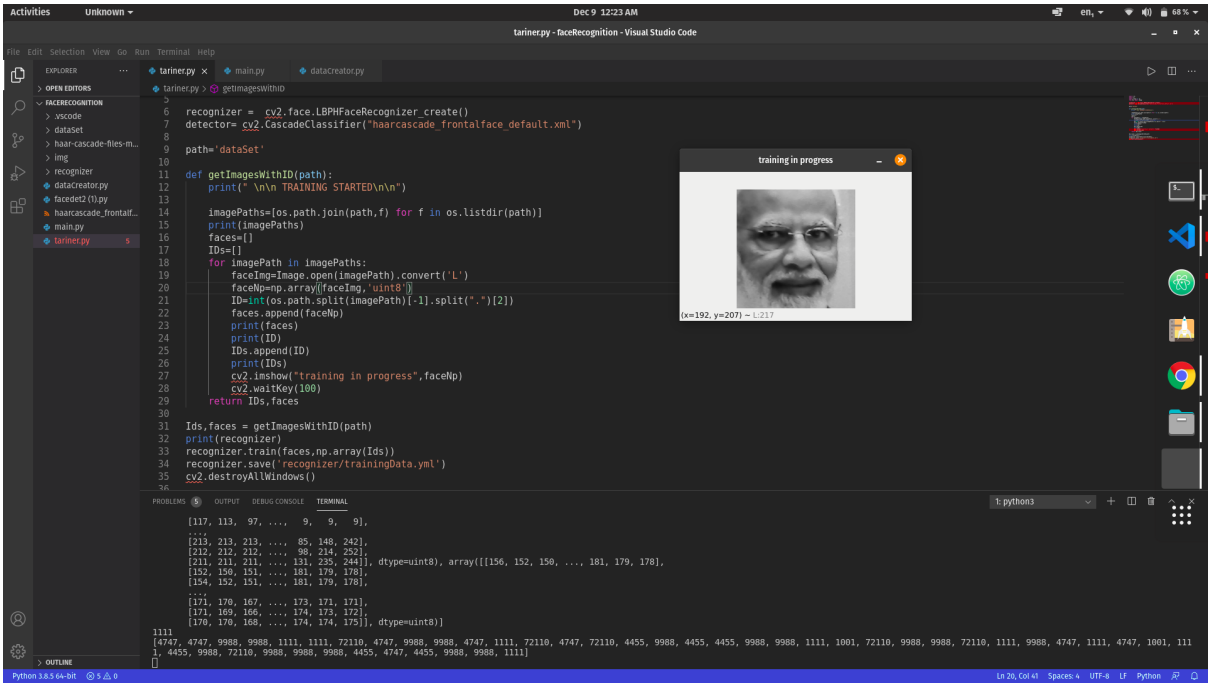
```
    return profile
text=str(datetime.datetime.now())
def present(id):
    cmmd="""update information set time = %s where id =
%s AND time IS NULL"""
    mycursor.execute(cmmd, (text,id))
    mydb.commit()

while True:
    ret, im =cam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.2,5)

    for(x,y,w,h) in faces:
        id, conf = recognizer.predict(gray[y:y+h,x:x+w])
        print(id,conf)
        profile=getProfile(id)
        if(conf<70):
            if(profile!=None ):
             present(id)

cv2.rectangle(im,(x,y),(x+w,y+h),(0,225,0),2)
            cv2.putText(im,str(profile[0]),
(x,y+h+40),font, 1, (32,32,32),2)
            cv2.putText(im,str(profile[1]),
(x,y+h+80),font, 1, ( 32,32,32),2)
            cv2.putText(im,"SUCCESSFUL",
(x,y+h+140),font, 1.5, (0, 225,0),3)
            cv2.putText(im,text, (50,50),font, 0.5, (
0,0,0),2)

        else:
            cv2.rectangle(im,(x,y),(x+w,y+h),(0,0,225),2)
            cv2.putText(im,"UNKNOWN", (x-40,y-30),font,
1.5, (0,0,225),3)
            cv2.putText(im,"NO MATCH FOUND!!!!",
(x-160,y+h+50),font, 1.5, (0,0,225),3)

    cv2.imshow("image",im)
    if(cv2.waitKey(1)& 0xff==ord("q")):
        break
```

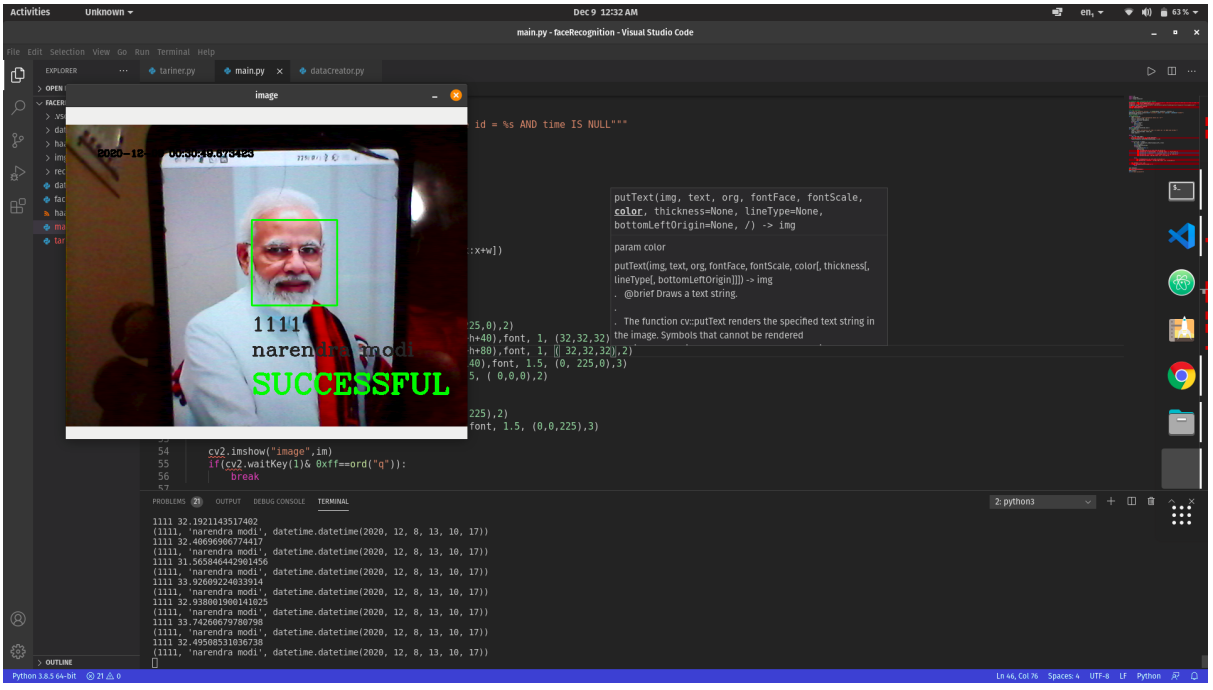
```
cam.release()
cv2.destroyAllWindows()
mydb.close()
print("code successful")
```

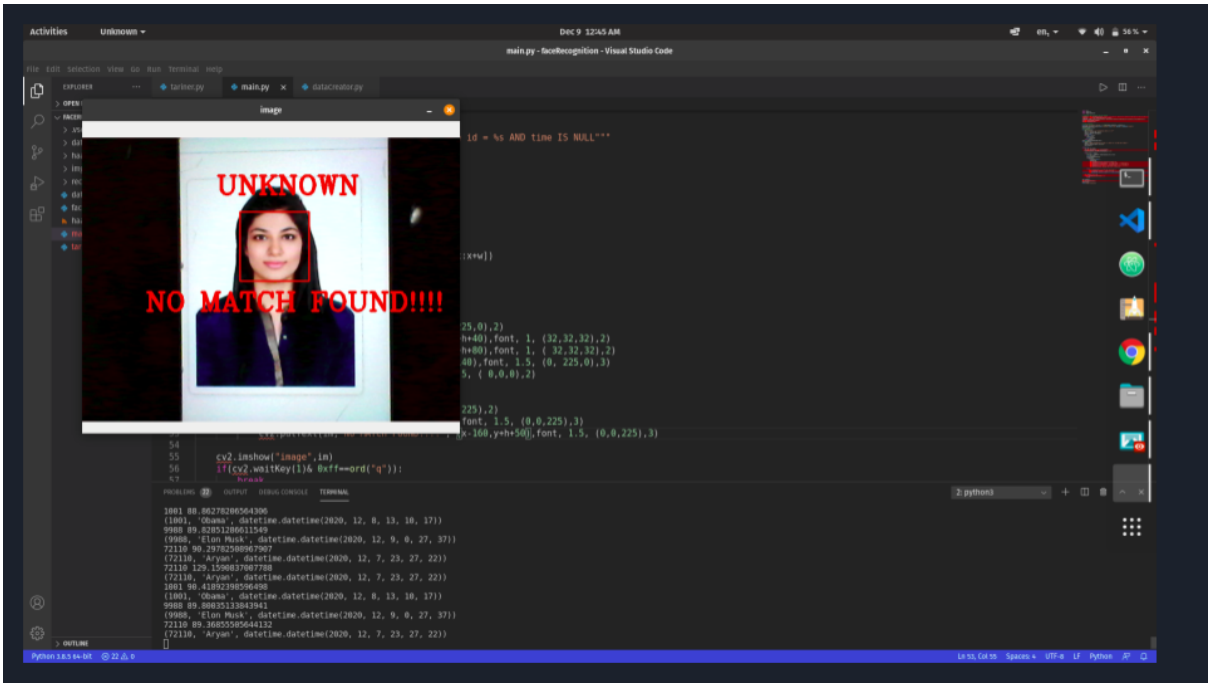
# WORKING OF THE SYSTEM

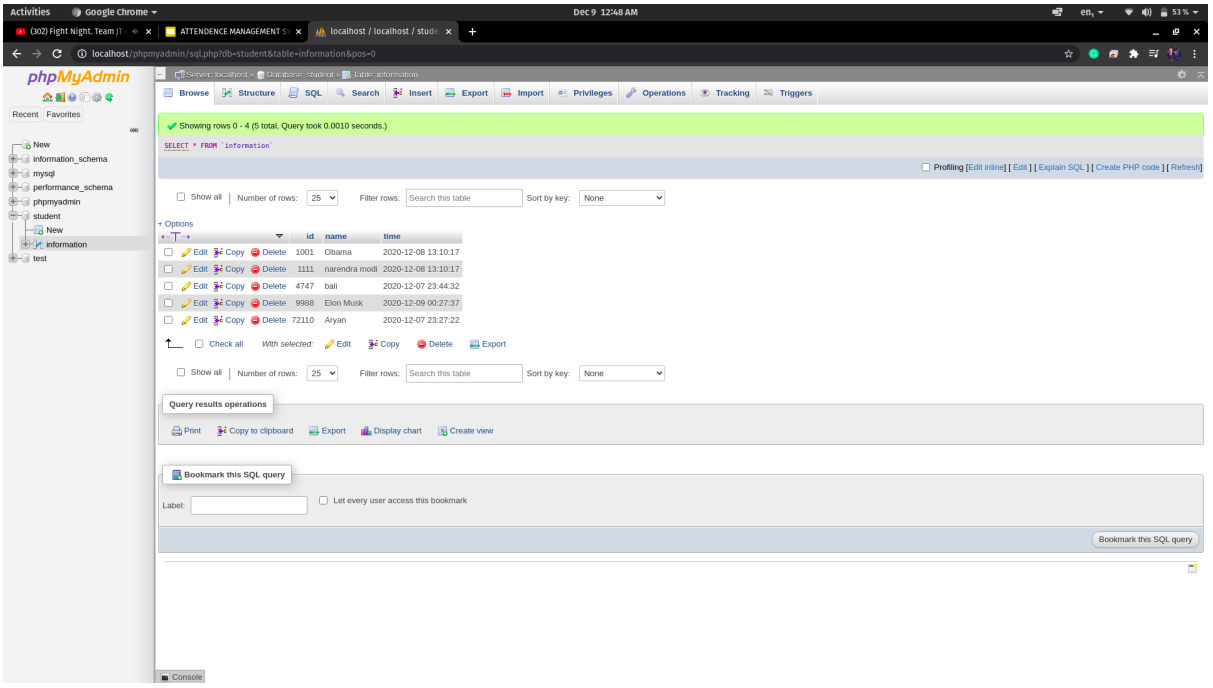

Activities
Unknown
Dec 8 1:36 PM
dataCreator.py - faceRecognition - Visual Studio Code
frame
Elon Musk
PROBLEMS 29
OUTPUT
DEBUG CONSOLE
TERMINAL
(72110, 'Aryan', datetime.datetime(2020, 12, 7, 23, 27, 22))
enter user name Elon Musk
enter rollno 9988
code successful
eclipse@pop-os:~/aryan/vs code/projects/python/faceRecognition$ python3 dataCreator.py
My sql connector version 8.0.22
(1001, 'Obama', datetime.datetime(2020, 12, 8, 13, 10, 17))
(1111, 'narendra modi', datetime.datetime(2020, 12, 8, 13, 10, 17))
(4747, 'bali', datetime.datetime(2020, 12, 7, 23, 44, 32))
(72110, 'Aryan', datetime.datetime(2020, 12, 7, 23, 27, 22))
enter user name Elon Musk
enter rollno 9988
code successful
eclipse@pop-os:~/aryan/vs code/projects/python/faceRecognition$ python3 dataCreator.py
My sql connector version 8.0.22
(1001, 'Obama', datetime.datetime(2020, 12, 8, 13, 10, 17))
(1111, 'narendra modi', datetime.datetime(2020, 12, 8, 13, 10, 17))
(4747, 'bali', datetime.datetime(2020, 12, 7, 23, 44, 32))
(72110, 'Aryan', datetime.datetime(2020, 12, 7, 23, 27, 22))
enter user name Elon Musk
enter rollno 9988


PROBLEMS 29
OUTPUT
DEBUG CONSOLE
TERMINAL
eclipse@pop-os:~/aryan/vs code/projects/python/faceRecognition$ python3 dataCreator.py
My sql connector version 8.0.22
(1001, 'Obama', datetime.datetime(2020, 12, 8, 13, 10, 17))
(1111, 'narendra modi', datetime.datetime(2020, 12, 8, 13, 10, 17))
(4747, 'bali', datetime.datetime(2020, 12, 7, 23, 44, 32))
(72110, 'Aryan', datetime.datetime(2020, 12, 7, 23, 27, 22))
enter user name Elon Musk
enter rollno 9988


tariner.py - faceRecognition - Visual Studio Code
training in progress
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
path='dataSet'
def getImagesWithID(path):
print(" \n\n TRAINING STARTED\n\n")
imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
print(imagePaths)
faces=[]
IDs=[]
for imagePath in imagePaths:
faceImg=Image.open(imagePath).convert('L')
faceNp=np.array(faceImg,'uint8')
ID=int(os.path.split(imagePath)[-1].split(".")[2])
faces.append(faceNp)
print(faces)
print(ID)
IDs.append(ID)
print(IDs)
cv2.imshow("training in progress",faceNp)
cv2.waitKey(100)
return IDs,faces
Ids,faces = getImagesWithID(path)
print(recognizer)
recognizer.train(faces,np.array(Ids))
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()


main.py - faceRecognition - Visual Studio Code
image
1111
SUCCESSFUL
putText(img, text, org, fontFace, fontScale, color, thickness=None, lineType=None, bottomLeftOrigin=None, /) -> img
param color
putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) -> img
. @brief Draws a text string.
. The function cv::putText renders the specified text string in the image. Symbols that cannot be rendered
cv2.imshow("image",im)
if(cv2.waitKey(1)& 0xff==ord("q")):
break


UNKNOWN
NO MATCH FOUND!!!!


phpMyAdmin
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.)
SELECT * FROM `information`
id
name
time
1001 Obama 2020-12-08 13:10:17
1111 narendra modi 2020-12-08 13:10:17
4747 bali 2020-12-07 23:44:32
9988 Elon Musk 2020-12-09 00:27:37
72110 Aryan 2020-12-07 23:27:22
Query results operations
Bookmark this SQL query

# Chapter-5:Summary

In the existing system for attendance making systems work manually which sometimes causes mistakes and the accuracy level wasn't able to reach 100%. Sometimes students make proxies for other students on their behalf though students are missing from the classes. Hence we will propose a system through which the attendance will be taken by detecting their faces in the dataset and then the algorithm will work in the backend to recognize the name and id of the particular face which is shown in the camera and then the timestamp will be populated in the tables.
It will collect data for entry as well as for the exit of any individual student.

---