FACIAL RECOGNITION ATTENDANCE SYSTEM

A Multi Disciplinary Design project report submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE

by

BIKRANT SEN (RA1811027010045)

&

SHASHANK VERMA (RA1811027010050)

under the guidance of

MS. BRISKILAL J.

Assistant Professor

Department of Computer Science Technology

SRM Institute of Science & Technology

At

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Kattankulathur, Chennai - 603203

(October 2020)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

KATTANKULATHUR

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Register No.: **RA1811027010045**

RA1811027010050

BONAFIDE CERTIFICATE

Certified to be the bonafide record of the work done by **BIKRANT SEN** & **SHASHANK VERMA** of B.Tech Computer Science Engineering with specialization in Big Data Analytics, Third year, V Semester for the award of B.Tech degree course in the Department of Computer Science Engineering in **18CSP103L- SEMINAR** 1 project during the Academic year 2020-21.

PROJECT IN-CHARGE **HEAD OF THE DEPARTMENT**

ABSTRACT

This report will show how we can implement algorithms for face detection and recognition in image processing to build a system that will detect and recognize faces of students in a classroom. In human interactions, the face is the most important factor as it contains important information about a person or individual. All humans have the ability to recognize individuals from their faces. The proposed solution is to develop a prototype of a system that will facilitate class control/attendance for students in a classroom by detecting the frontal faces of students.

In recent years, research has been carried out and face recognition and detection systems have been developed. Some of which are used on social media platforms, banking apps, government offices e.g. the Metropolitan Police, Facebook etc.

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my guide, Ms. Briskilal J. (Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology) her valuable guidance, consistent encouragement, personal caring, timely help and providing us with an excellent atmosphere for doing research related to our project. All through the work, in spite of her busy schedule, she has extended cheerful and cordial support to us for completing this project.

Alongside would also like to thank both of our parents to be a constant support during the course of the project.

BIKRANT SEN

(name of student 1)

SHASHANK VERMA

(name of student 2)

TABLE OF CONTENTS

	page no.
1. List of snapshots	6
2. Introduction	7
3. Requirement analysis	9
4. Design	12
5. Implementation	17
6. Result	19
7. Conclusion	22
8. Bibliography	23

LIST OF SNAPSHOTS

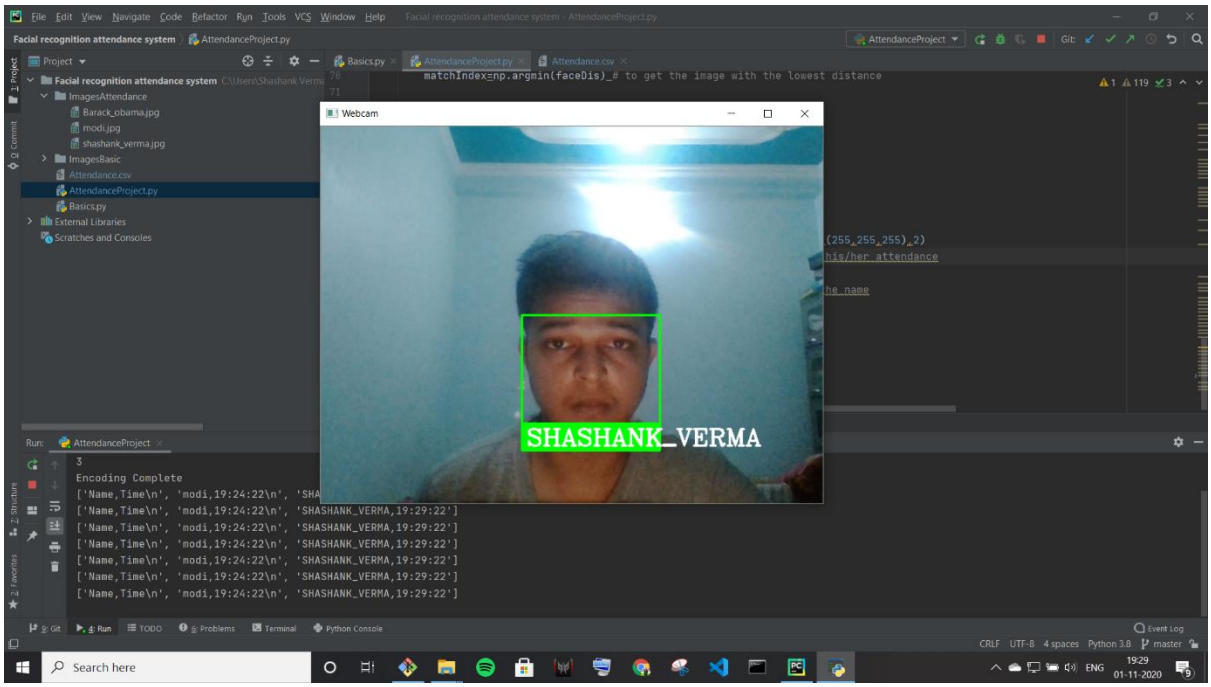
The following are the list of snapshots used in the report.

- Introduction screenshot
- Design flowchart
- Facial attendance workflow
- Train image
- Test image
- Test image-FALSE
- Test image-TRUE
- Result images
- .csv excel file

1.

INTRODUCTION

In Face Detection and Recognition systems, the flow process starts by being able to detect and recognize frontal faces from an input device. Face detection and recognition is not new in our society we live in. The capacity of the human mind to recognize particular individuals is remarkable. It is amazing how the human mind can still persist in identification of certain individuals even through the passage of time, despite slight changes in appearance.



PROBLEM DEFINITION:

This project is being carried out due to the concerns that have been highlighted on the methods which lectures use to take attendance during lectures. The use of clickers, ID cards swiping and manually writing down names on a sheet of paper as a method to track student attendants has prompted this project to be carried out. This is not in any way to criticize the various methods used for student attendance, but to build a system that will detect the faces present in a classroom as well as recognizing them. Also, a teacher will be able to tell if a student was honest as these methods mentioned can be used by anyone for

attendance records, but with the face detection and recognition system in place, it will be easy to tell if a student is actually present in the classroom or not.

This system will not only improve classroom control during lectures, it will also possibly detect faces for student attendance purposes.

AIM:

To develop a prototype that will facilitate classroom control and attendance by face detection and recognition of students face in a digital image.

OBJECTIVES:

- To detect the face of the student at some accuracy and match it with the faces present in the database.
- To ensure the update in the database resulting in automatic facial attendance.
- Minimize the time and effort spent in oral or written attendance.
- Reduce the human errors and enhance efficiency in attendance records.

2.

REQUIREMENT ANALYSIS

The project is developed on Python and has been implemented using PyCharm.

The following are the few dependencies used:

- OpenCV
- NumPy
- face-recognition
- os module in Python

OpenCV

OpenCV is an open-source library that includes several hundreds of computer vision algorithms.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

OpenCV handles all the memory automatically. OpenCV deallocates the memory automatically, as well as automatically allocates the memory for output function parameters most of the time. As a computer vision library, OpenCV deals a lot with image pixels that are often encoded in a compact, 8- or 16-bit per channel, form and thus have a limited value range. Furthermore, certain operations on images, like color space conversions, brightness/contrast adjustments, sharpening, complex interpolation (bi-cubic, Lanczos) can produce values out of the available range. If you just store the lowest 8 (16) bits of the result, this results in visual artifacts and may affect a further image analysis. To solve this problem, the so-called saturation arithmetics is used.

For documentation click here.

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image.

The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

For documentation click here.

face-recognition

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

For documentation click here.

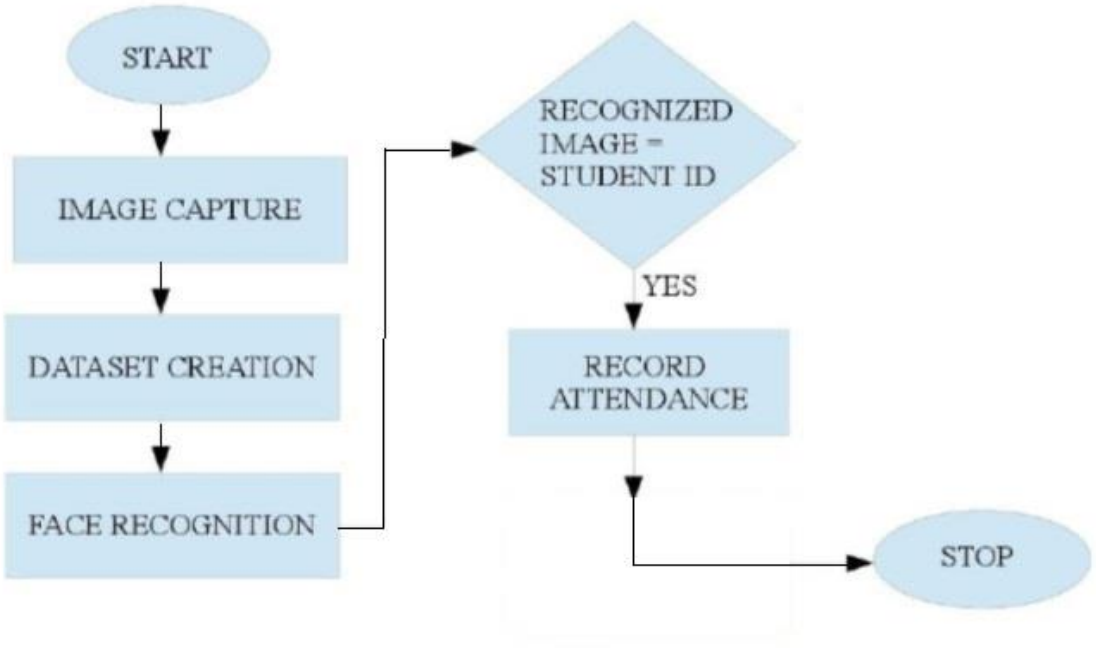
os module in python

The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

For documentation click here.

3.

DESIGN



Database Creation – Database of student is created before the recognition process, which includes one image of each individual. This is done by the respected teacher through the admin panel. During this process admin will simultaneously enter the student's name.

Face Recognition – This is the most important part of the proposed system. As students enter the class, teacher will switch on the camera and it will continuously detect and recognize the face. After recognizing all the students present in the class an excel file is created giving the attendance of the class with date.

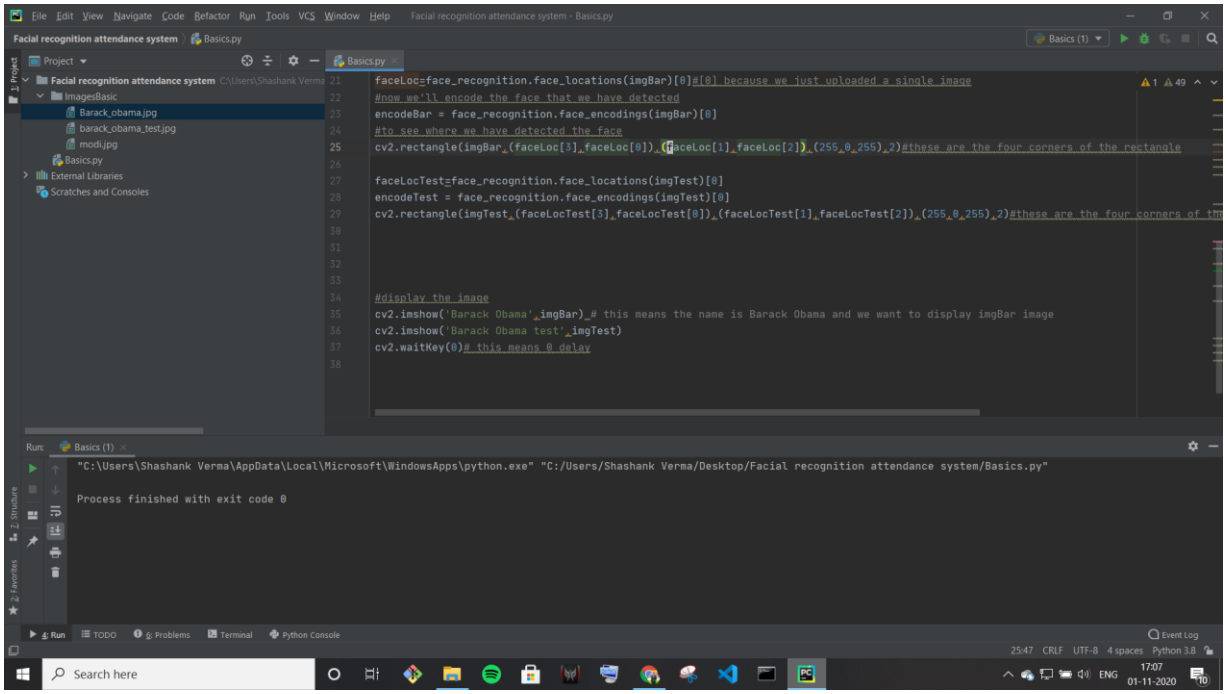
Webcam - A webcam is a video camera that feeds or streams its image in real time to or through a computer to a computer network. When "captured" by the computer, the video stream may be saved, viewed or sent on to other networks via systems such as the internet, and emailed as an attachment.

The basic design for facial recognition is as follows:

```
import cv2
import numpy as np
import face_recognition
#print('All necessary libraries imported!')

#First step is loading the images and converting them into RGB, because
we get the image as BGR but the library understands the image as RGB

#What we are gonna do is first we will test the image of barack obama
and find the encodings and then we'll test our model with the test
images whether it's of barack obama or not

#firstly, we'll need to import our image
imgBar=face_recognition.load_image_file('ImagesBasic/Barack_obama.jpg')#
then we'll convert this to RGB, since it's BGR
imgBar=cv2.cvtColor(imgBar,cv2.COLOR_BGR2RGB)



#Similarly import our test image
imgTest=face_recognition.load_image_file('ImagesBasic/barack_obama_test.
jpg')
imgTest=cv2.cvtColor(imgTest,cv2.COLOR_BGR2RGB)

# The next step is to find the faces in our images and then their
encodings
faceLoc=face_recognition.face_locations(imgBar)[0]#[0] because we just
uploaded a single image
#now we'll encode the face that we have detected
encodeBar = face_recognition.face_encodings(imgBar)[0]
#to see where we have detected the face
cv2.rectangle(imgBar,(faceLoc[3],faceLoc[0]),(faceLoc[1],faceLoc[2]),(25
5,0,255),2)#these are the four corners of the rectangle

faceLocTest=face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),(faceLocTest[1],fa
ceLocTest[2]),(255,0,255),2)#these are the four corners of the rectangle

#the third and the final step is to compare these two faces and find the
distance between the encodings
#in the backend we'll use Linear SVM to check whether they are
equivalent or not

results = face_recognition.compare_faces([encodeBar],encodeTest)
faceDis=face_recognition.face_distance([encodeBar],encodeTest)
print(results)# if true that means it has found the similar encodings
and false if the encodings are different
print(faceDis)
cv2.putText(imgTest,f'{results}{round(faceDis[0],2)}',(50,50),cv2.FONT_H
ERSHEY_COMPLEX,1,(0,0,255),2)

#display the image
cv2.imshow('Barack Obama',imgBar) # this means the name is Barack Obama
and we want to display imgBar image
cv2.imshow('Barack Obama test',imgTest)
cv2.waitKey(0)# this means 0 delay
```

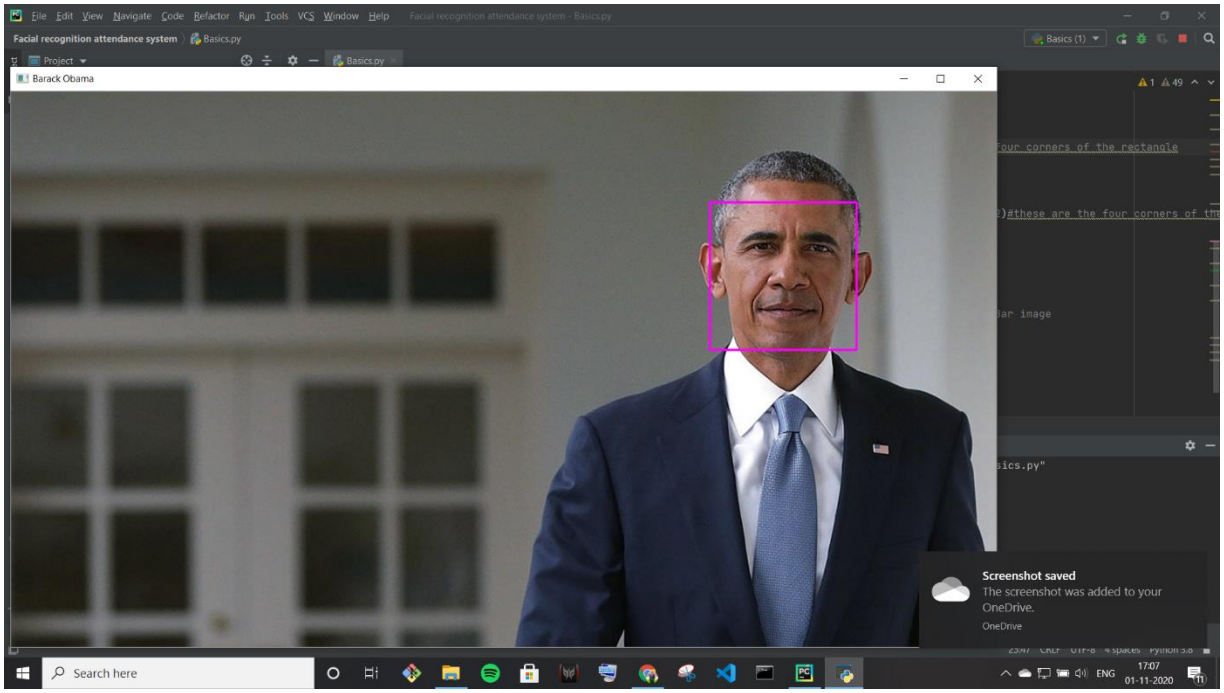
FACIAL ATTENDANCE WORKFLOW:

This depicts the basic facial recognition system.



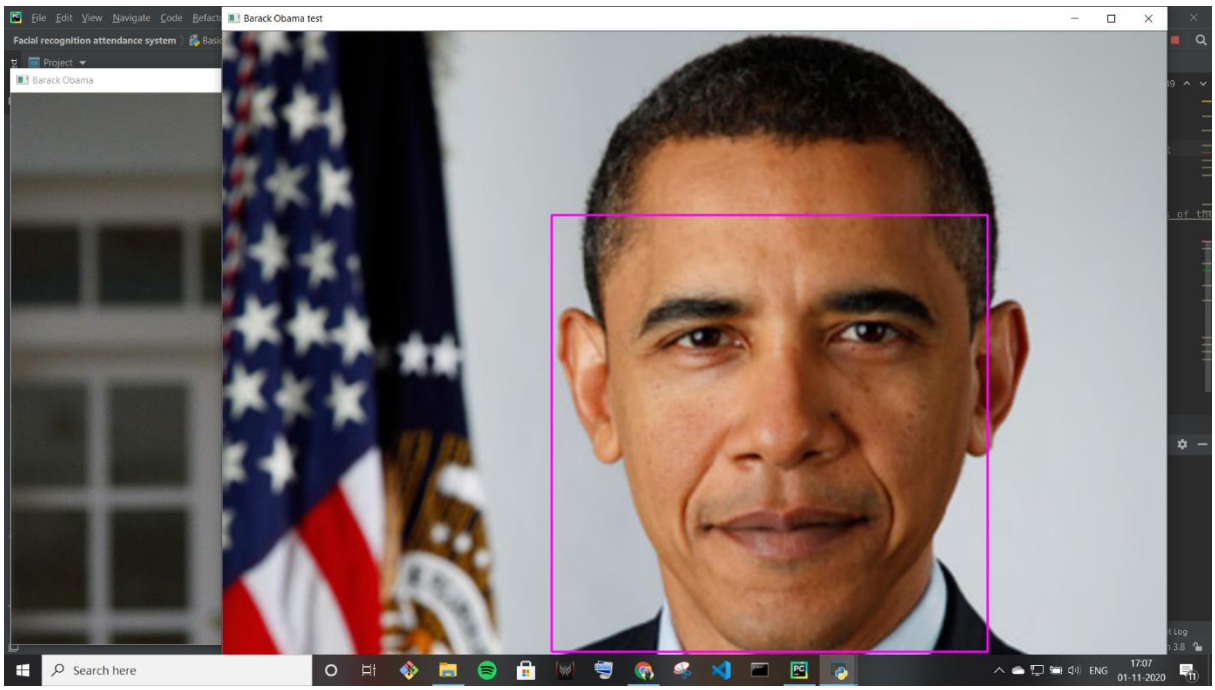
On compiling the above code we train it with the dataset available to analyse the photos present in the attendance dataset.

Thus the system trains with the data and we can see the face-recognition library detecting the faces in the image as shown below.

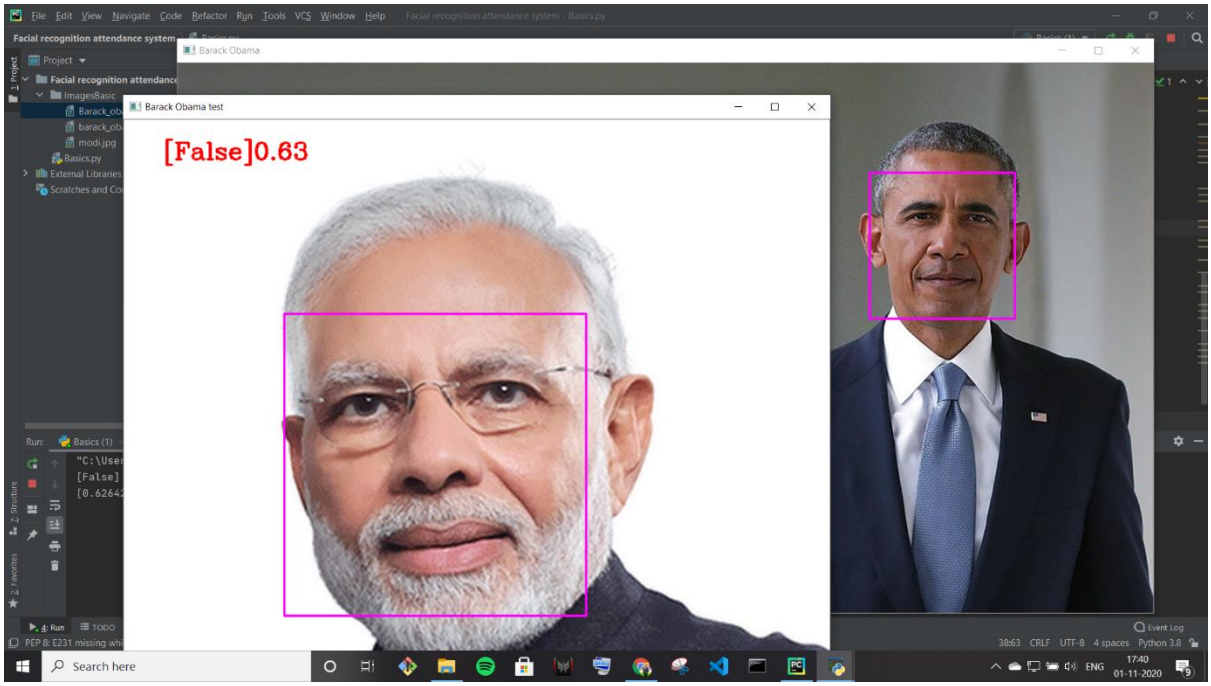


After the system is trained, we test it with the test images.

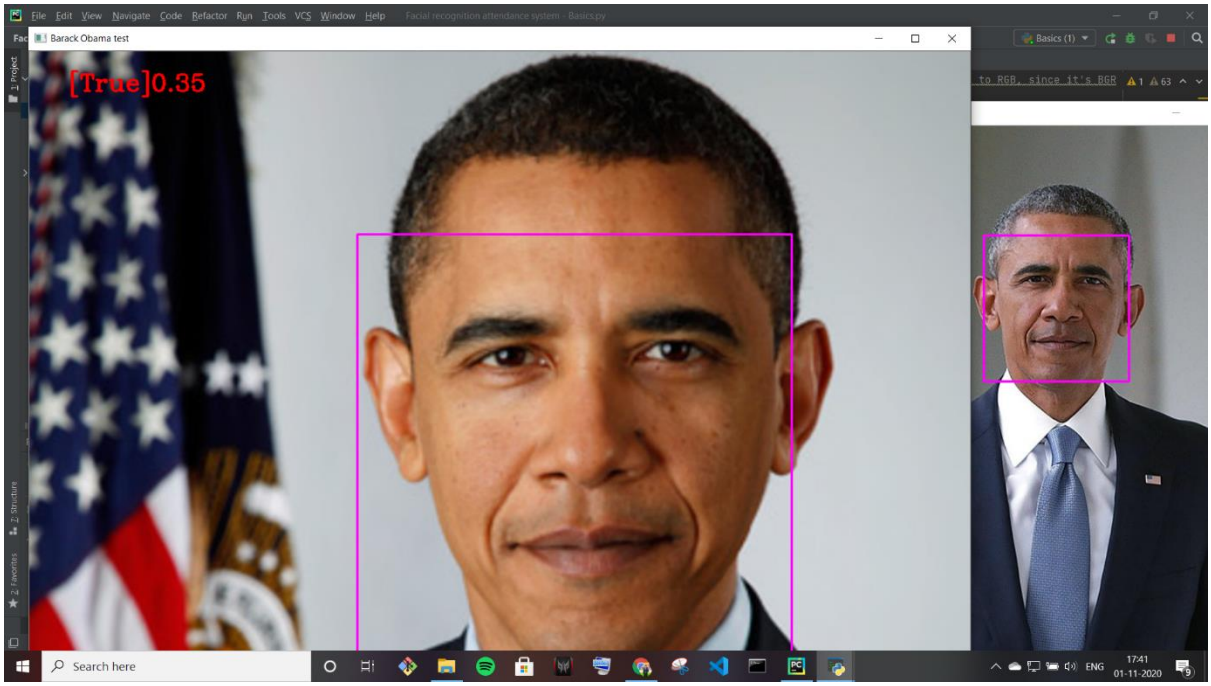
As shown below it very well detects the face and proceeds for the True and False testing.



Thus in the below images we can see that the system checks for both the photos and results out FALSE with a value which depicts distance. The lesser the distance the better the encodings and the similar the picture.



Below it shows the TRUE result.



4.

IMPLEMENTATION

We use the above idea of facial recognition to record the attendance of the classes and we have the following output recorded on few students of our class.

The Implementation goes as follows.

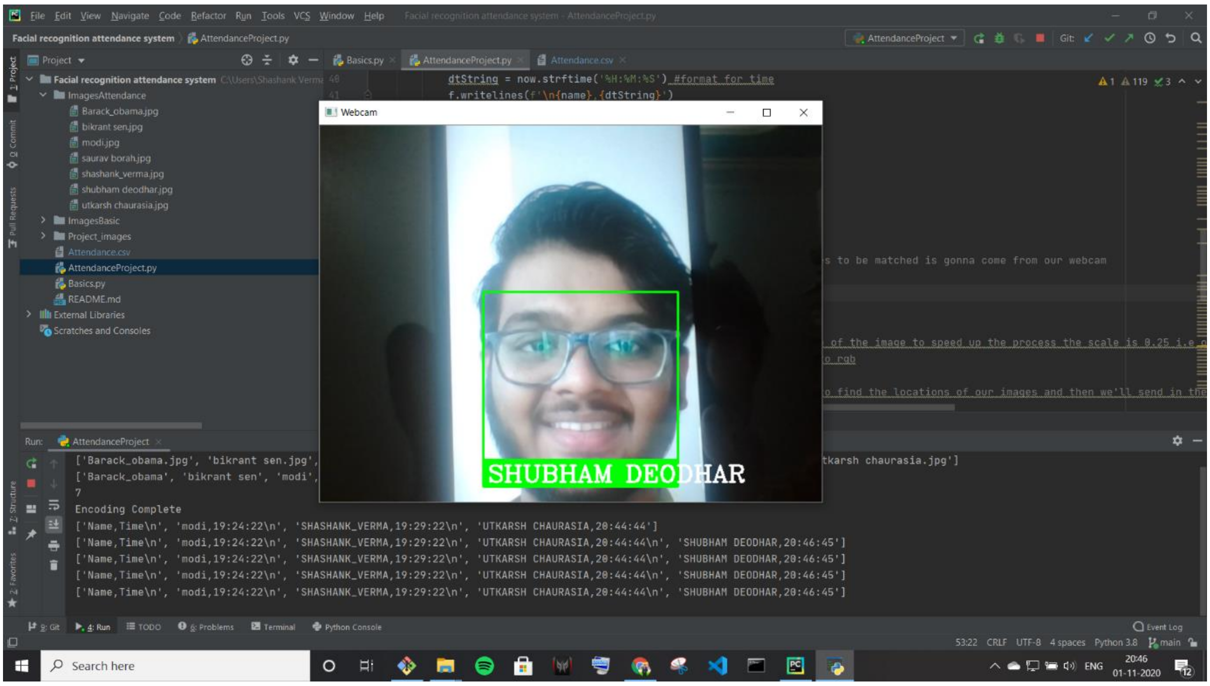
```
import cv2
import numpy as np
import face_recognition
#we create a list that will create a list of all the images in the folder
ImagesBasic and create it's encodings
import os
from datetime import datetime

path = 'ImagesAttendance'
images = []
classNames =[]
mylist=os.listdir(path)
print(mylist)
#next we'll use these names and import these images one by one
for cl in mylist:
    curImg=cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])#so that .jpg doesn't come
in the names

print(classNames)
#next we start with our encoding process, we'll find the encodings for
each and every image
def findEncodings(images):
    encodeList=[]
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)#convert the images to
rgb
        encode =face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
#We'll create the function to mark our attendance
def markAttendance(name):
    with open('Attendance.csv','r+') as f: #to open the file and we want
to read and write at the same time
        #now we'll read in all the lines currently in our data, so that
somebody arrives twice we don't want to repeat the entry
        myDataList=f.readlines()
        nameList=[]#we want to put all the names already present in this
list
```

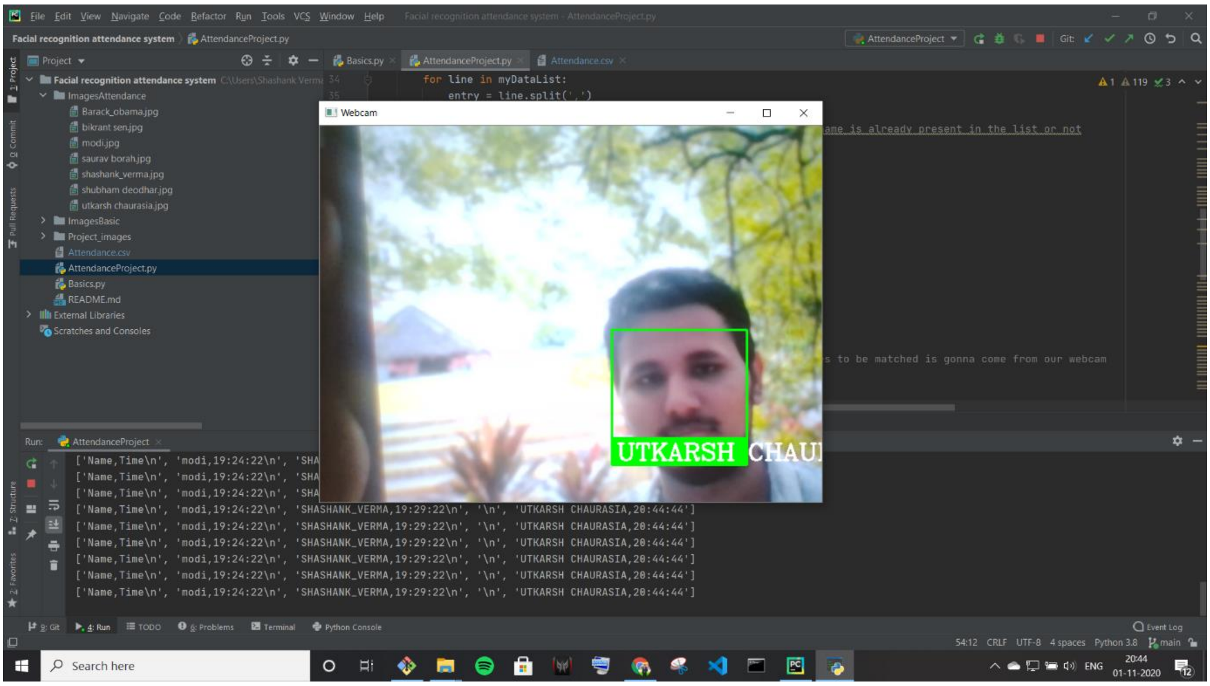
```
for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        #once we have this list complete, we'll check that the current
name is already present in the list or not
        if name not in nameList:
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S') #format for time
            f.writelines(f'\n{name},{dtString}')


        print(myDataList)


encodeListKnown = findEncodings(images)
print(len(encodeListKnown))
print("Encoding Complete")

# The third step is to find the matches between our encodings. the
images to be matched is gonna come from our webcam

cap=cv2.VideoCapture(0)
while True:
    success,img=cap.read()
    imgS=cv2.resize(img,(0,0),None,0.25,0.25)#We want to reduce the
size of the image to speed up the process the scale is 0.25 i.e one-
fourth of the original size
    imgS=cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB)# We'll convert the image
to rgb

    #Now we can find multiple images in our webcam so for that we need
to find the locations of our images and then we'll send in the
locations of our images to the encoding function
    #to find the location
    facesCurFrame= face_recognition.face_locations(imgS)
    encodesCurFrame=face_recognition.face_encodings(imgS,facesCurFrame)
    #we'll iterate through all the faces in our current frame
    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame): #one
by one it'll grab one face location and it's encoding
        #then we'll perform the matching

matches=face_recognition.compare_faces(encodeListKnown,encodeFace)
        #then we'll find the distance

faceDis=face_recognition.face_distance(encodeListKnown,encodeFace)#The
lowest distance will be our best match
        #print(faceDis)
        matchIndex=np.argmin(faceDis) # to get the image with the
lowest distance

        if matches[matchIndex]:
            name = classNames[matchIndex].upper() #to convert uppercase
            #print(name)
            #we'll create a rectangle around the face
            y1,x2,y2,x1=faceLoc
            y1, x2, y2, x1=y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
```

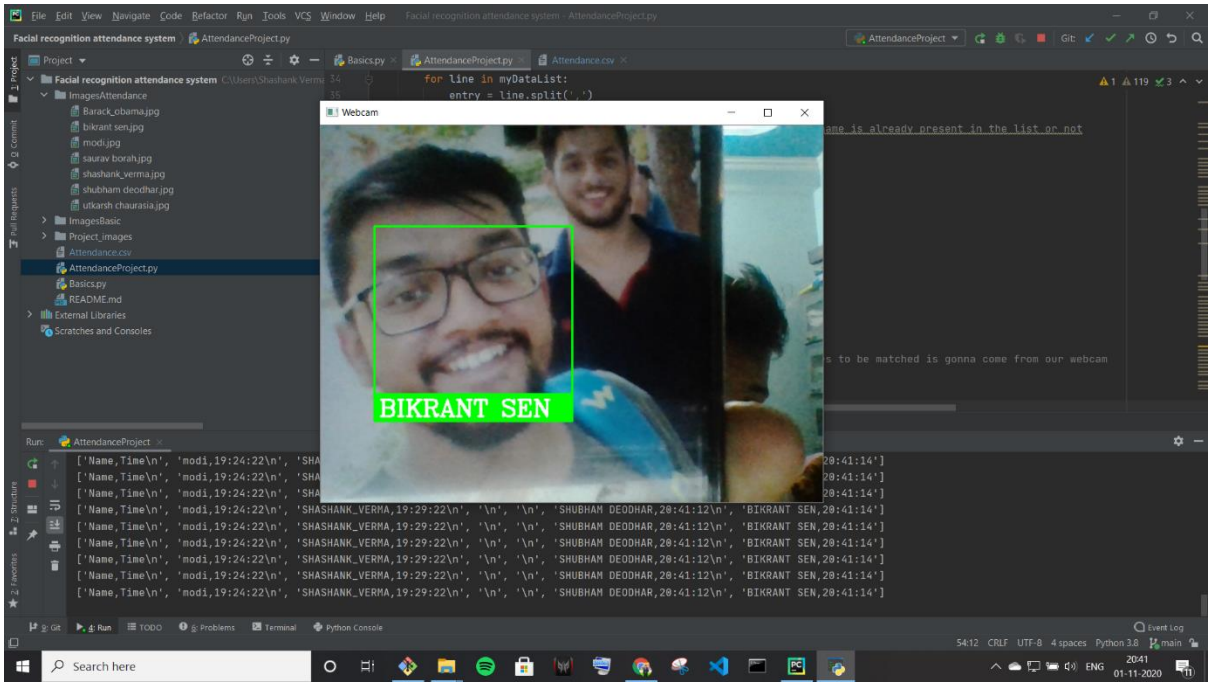
5.

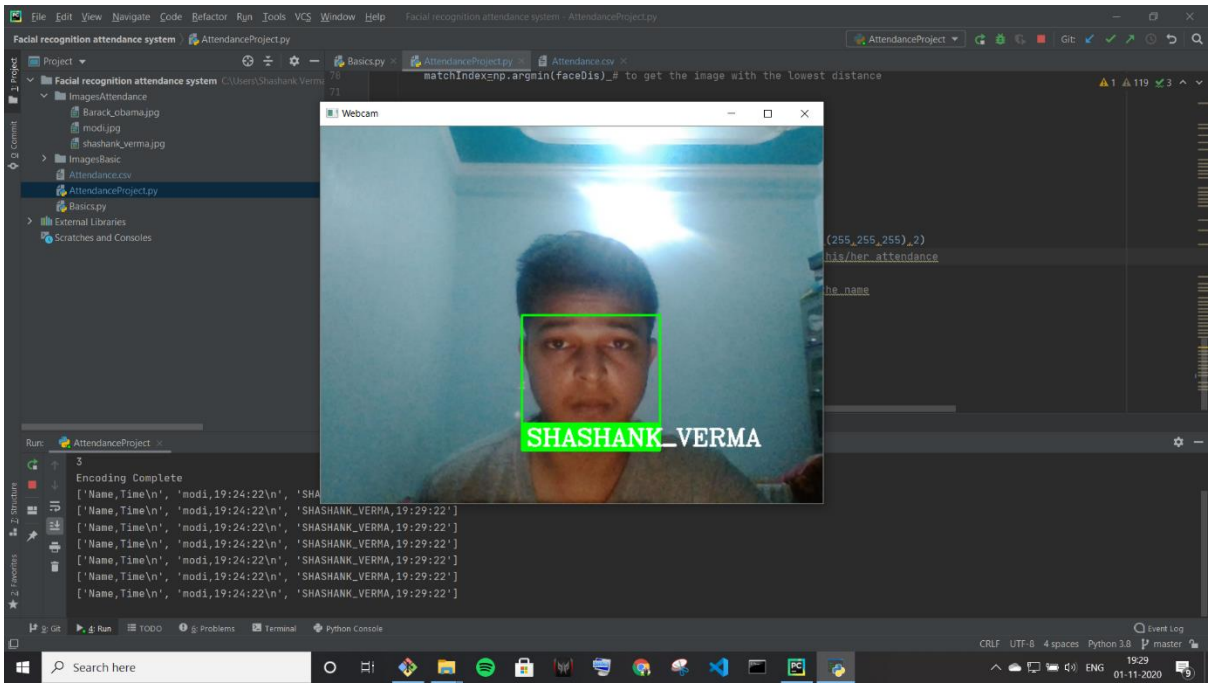
RESULT

We use the above idea of facial recognition to record the attendance of the classes and we have the following output recorded on few students of our class.



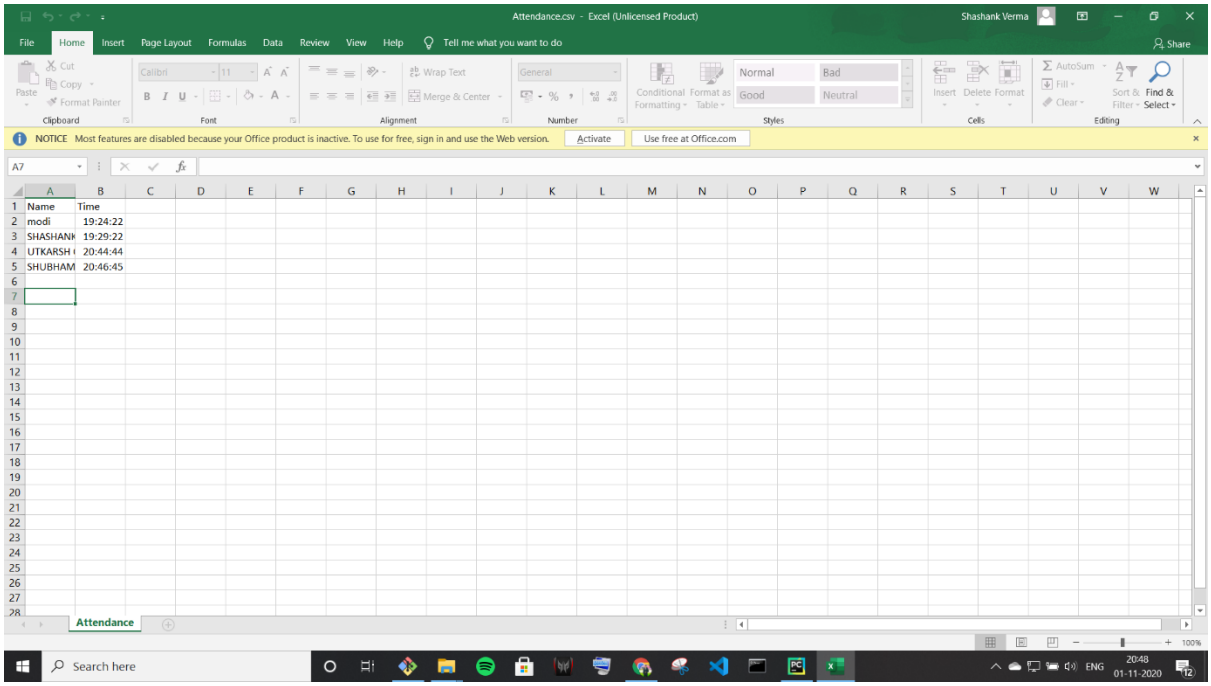






Thus we see that the system successfully detects the faces of the students and further we see that the system stores the attendance in .csv file.

Thus the Attendance.csv file when opened in Excel is as follows.



6.

CONCLUSION

Thus on completion of this project we have successfully made a FACIAL RECOGNITION ATTENDANCE SYSTEM which successfully detects faces and maps them into a .csv file. Thus this will enhance the attendance system and will ensure lesser errors and will in turn provide proper time management

With the completion of this project we are extremely happy to have learnt the documentation and implementation of OpenCV and face-recognition libraries thus boosting the confidence to work in more projects related to it.

It was indeed useful to have invested in this project as it taught us many new dependencies and also enabled us to be thorough with the Python implementation.

7.

BIBLIOGRAPHY

1. Murtaza Hassan's workshop (https://github.com/murtazahassan/Face-Recognition)
2. OpenCV documentation (https://opencv.org/)
3. Face-recognition documentation (https://face-recognition.readthedocs.io/en/latest/)
4. NumPy documentation (https://numpy.org/doc/)