**GUDLAVALLERU ENGINEERING COLLEGE**



**ELECTRONICS AND COMMUNICATION ENGINEERING**

**PROJECT NAME:** IOT Based Automated Attendence Based On Face Recognition

**SUBMITTED BY:**

D. Bhavani Shankhar - 18481A0445

D.Mohan Venkat - 18481A0446

D.Vennela - 18481A0447

D.Yuvaraj - 18481A0448

D.Sasidhar - 18481A0450

**INDEX**

**TITLE**

1. INTRODUCTION

1.1 Overview

1.2 Background

1.3 Current Systems

1.4 Drawbacks in existing systems

2. Literature Survey

          2.1 Existing problem

2.2  Student Attendance System

         2.3  Face detection

     3. Theoretical Analysis

           3.1 Aims and Objective

3.2 Block diagram

3.3 Tools required

     4. Experimental Investigations

     5. Flowchart

     6. Advantages & Disadvantages

     7. Applications                                                                             

8. Conclusion

    9. Appendix

          9.1 Source code

9.2 UI output Screenshot

**IOT based Automated Attendance system Based On Face Recognition**

**1 INTRODUCTION:**

**1.1 Overview**

The main objective of this project is to develop face recognition based automated student attendance system. In order to achieve better performance, the test images and training images of this proposed approach are limited to frontal and upright facial images that consist of a single face only. The test images and training images have to be captured by using the same device to ensure no quality difference. In addition, the students have to register in the database to be recognized. The enrolment can be done on the spot through the user-friendly interface.

**1.2 Background**

Face recognition is crucial in daily life in order to identify family, friends or someone we are familiar with. We might not perceive that several steps have actually taken in order to identify human faces. Human intelligence allows us to receive information and interpret the information in the recognition process. We receive information through the image projected into our eyes, by specifically retina in the form of light. Light is a form of electromagnetic waves which are radiated from a source onto an object and projected to human vision.

The human face is a unique representation of individual identity. Thus, face recognition is defined as a biometric method in which identification of an individual is performed by comparing real-time capture image with stored images in the database of that person (Margaret Rouse, 2012).

The work on face recognition began in 1960. Woody Bledsoe, Helen Chan Wolf and Charles Bisson had introduced a system which required the administrator to locate eyes, ears, nose and mouth from images. The distance and ratios between the located features and the common reference points are then calculated and compared. The studies are further enhanced by Goldstein, Harmon, and Lesk in 1970 by using other features such as hair colour and lip thickness to automate the recognition. In 1988, Kirby and Sirovich first suggested principle component analysis (PCA) to solve face recognition problem. Many studies on face recognition were then conducted continuously until today.

**1.3 Current Systems**

At present ,attendance,marking involves manual attendance on the paper sheet by professors and teachers ,but it is very time consuming process and chances of proxy and also an issue that arises in such type of attendance marking.

**1.4 Drawbacks in existing systems**

Manual systems put pressure on people to be correct in all details of their work at all times ,the problem being that people aren’t perfect,however each of us wishes we were

* There is always a forgery(one person signing the presence of the other one) Since these are manually so there is a great risk of error
* More power is required
* It is difficult to maintain a database or register in manual systems

**2.0 Literature Review**

**2.1 Existing Methods :**

A traditional way of taking attendance is the teacher or mentor will be calling the students by name or by using the respective roll numbers. It is a time taking process .

So to avoid this problems by using the traditional method we introduce this IOT based automated attendance based on face recognition.

**2.2 Student Attendance System**

Arun Katara et al. (2017) mentioned disadvantages of RFID (Radio Frequency Identification) card system, fingerprint system and iris recognition system. RFID card system is implemented due to its simplicity. However, the user tends to help their friends to check in as long as they have their friend’s ID card. The fingerprint system is indeed effective but not efficient because it takes time for the verification process so the user has to line up and perform the verification one by one. However for face recognition, the human face is always exposed and contain less information compared to iris. Iris recognition system which contains more detail might invade the privacy of the user. Voice recognition is available, but it is less accurate compared to other methods. Hence, face recognition system is suggested to be implemented in the student attendance system.

**2.3 Face Detection**

Difference between face detection and face recognition are often misunderstood. Face detection is to determine only the face segment or face region from image, whereas face recognition is to identify the owner of the facial image. S.Aanjanadevi et al. (2017) and Wei-Lun Chao (2007) presented a few factors which cause face detection and face recognition to encounter difficulties. These factors consist of background, illumination, pose, expression, occlusion, rotation, scaling.

**3.0 Theoritical Analysis**

**3.1 Aims and Objectives :**

The objective of this project is to develop face recognition based automated student attendance system. Expected achievements in order to fulfill the objectives are:

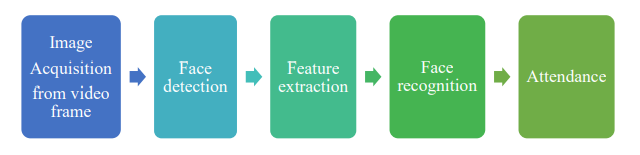
To detect the face segment from the video frame.

To extract the useful features from the face detected.

To classify the features in order to recognize the face detected.

To record the attendance of the identified student.

**3.2 Block Diagram**

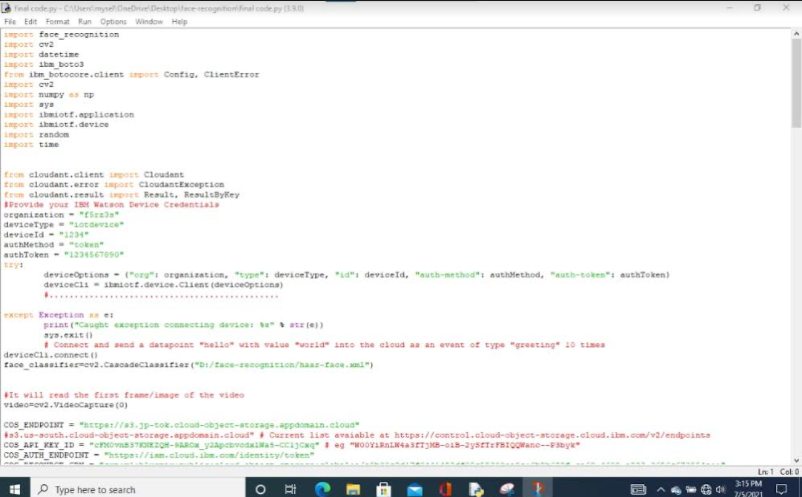
****

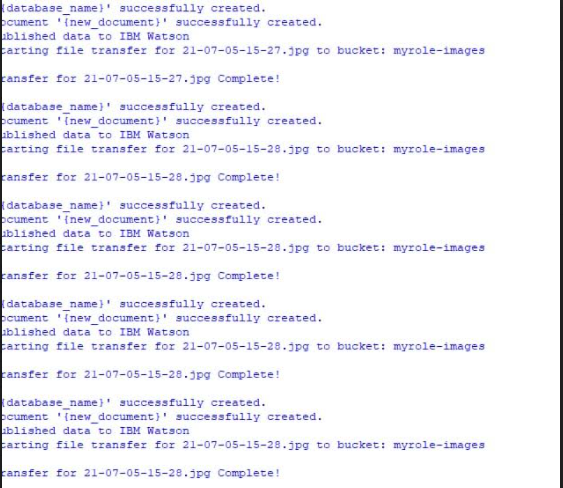
**3.3 Tools Required**

* Python IDLE(with specified packages installed)
* IBM Cloud
* Node RED Services

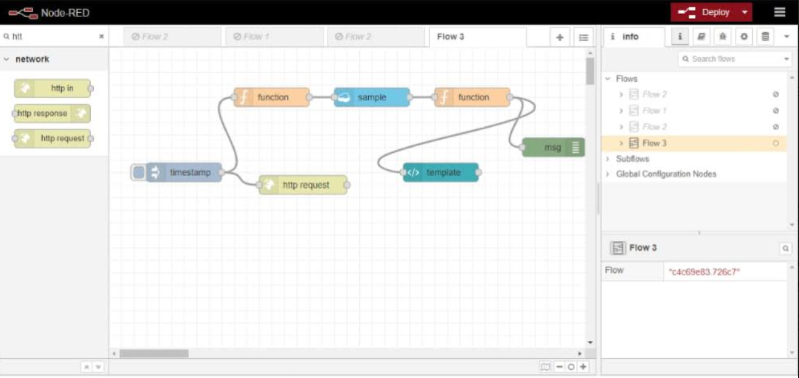
**4 .Experimental Investigations**

Initially we start our project by creating a python code for face capturing which means that when we run the code it directly capture the face whatever the image it receives first and save the image and gives the input to IBM cloud

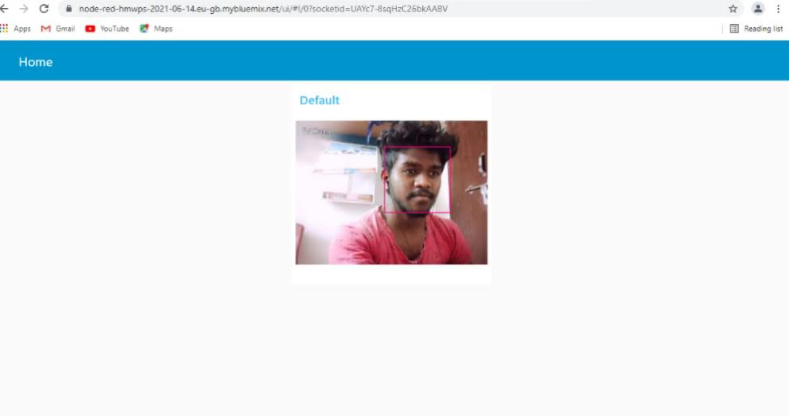




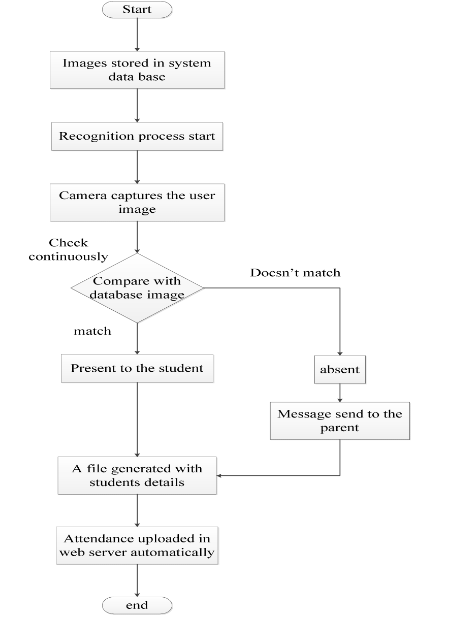
The second step is connect IBM cloud to the node red services then connect the all nodes means when captured is matched with the stored image then marked it as an attende other wise it makes as an absent .Then the message is sent to registerd mobile number



And final step is creation of WEB applications .By copying url of node-red and add ui for creation of the web page .so after deploying the node red connection the output will be appeared at the web page .



**5 . Flow Chart**

****

**6 . Advantages:**

* Increased security
* Time saving
* Cost-effective
* Easy to manage

**Disadvantage:**

* If in the wrong hands, it will be a disaster
* Low reliability
* The data can be easily tampered and used to in against a particular organization or person. Which can be very much dangerous
* Lack of regulations in the AI in Face recognition systems

**7 . Applications:**

* Used in smart phones to unlock the device
* Facilitate secure transactions
* Smart advertising
* Control access to sensitive areas(like military bases,bank vaults)
* Prevention of retail crime

**8. Conclusion:**

* The goal of our project is to make keeping track of attendance at events easier. There are many ways to evident people at events, for example using personalized cards, taking signatures at the beginning of an event, but we found that facial recognition is more practical than any other conventional way. We found that using facial recognition is quite challenging and we tackled many problems throughout the development. In the beginning there was a decision to be made, do we use a face recognition algorithm or do we use face encodings.

**9. Appendix:**

**9.1 Source code**

import face\_recognition

import cv2

# This is a demo of running face recognition on live video from your webcam. It's a little more complicated than the

# other example, but it includes some basic performance tweaks to make things run a lot faster:

# 1. Process each video frame at 1/4 resolution (though still display it at full resolution)

# 2. Only detect faces in every other frame of video.

# PLEASE NOTE: This example requires OpenCV (the `cv2` library) to be installed only to read from your webcam.

# OpenCV is \*not\* required to use the face\_recognition library. It's only required if you want to run this

# specific demo. If you have trouble installing it, try any of the other demos that don't require it instead.

# Get a reference to webcam #0 (the default one)

video\_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

obama\_image = face\_recognition.load\_image\_file(r"mohan.jpg")

obama\_face\_encoding = face\_recognition.face\_encodings(obama\_image)[0]

# Load a second sample picture and learn how to recognize it.

biden\_image = face\_recognition.load\_image\_file(r"Shankar.jpg")

biden\_face\_encoding = face\_recognition.face\_encodings(biden\_image)[0]

biden\_image1 = face\_recognition.load\_image\_file("sasidhar.jpg")

biden\_face\_encoding1 = face\_recognition.face\_encodings(biden\_image1)[0]

biden\_image2 = face\_recognition.load\_image\_file("yuvaraj.jpg")

biden\_face\_encoding2 = face\_recognition.face\_encodings(biden\_image2)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [

obama\_face\_encoding,

biden\_face\_encoding,

biden\_face\_encoding1,

biden\_face\_encoding2

]

known\_face\_names = [

"Mohan",

"Bhavanishankar",

"Sasidhar",

"Yuvaraj"

]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

i=0

process\_this\_frame = True

while True:

# Grab a single frame of video

ret, frame = video\_capture.read()

# Resize frame of video to 1/4 size for faster face recognition processing

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

# Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

rgb\_small\_frame = small\_frame[:, :, ::-1]

# Only process every other frame of video to save time

if process\_this\_frame:

# Find all the faces and face encodings in the current frame of video

face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

face\_names = []

for face\_encoding in face\_encodings:

# See if the face is a match for the known face(s)

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

# If a match was found in known\_face\_encodings, just use the first one.

if True in matches:

first\_match\_index = matches.index(True)

i=i+1

name = known\_face\_names[first\_match\_index]

print("Number of students present = %d", i)

face\_names.append(name)

process\_this\_frame = not process\_this\_frame

# Display the results

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

# Scale back up face locations since the frame we detected in was scaled to 1/4 size

top \*= 4

right \*= 4

bottom \*= 4

left \*= 4

# Draw a box around the face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image

cv2.imshow('Video', frame)

# Hit 'q' on the keyboard to quit!

if cv2.waitKey(1) & 0xFF == ord('q'):

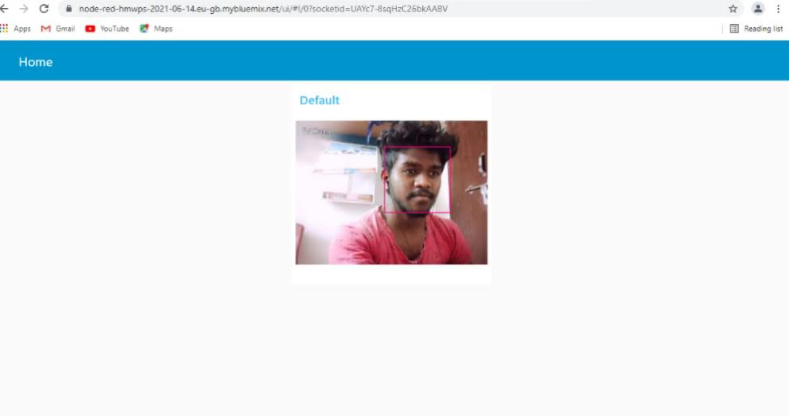
break

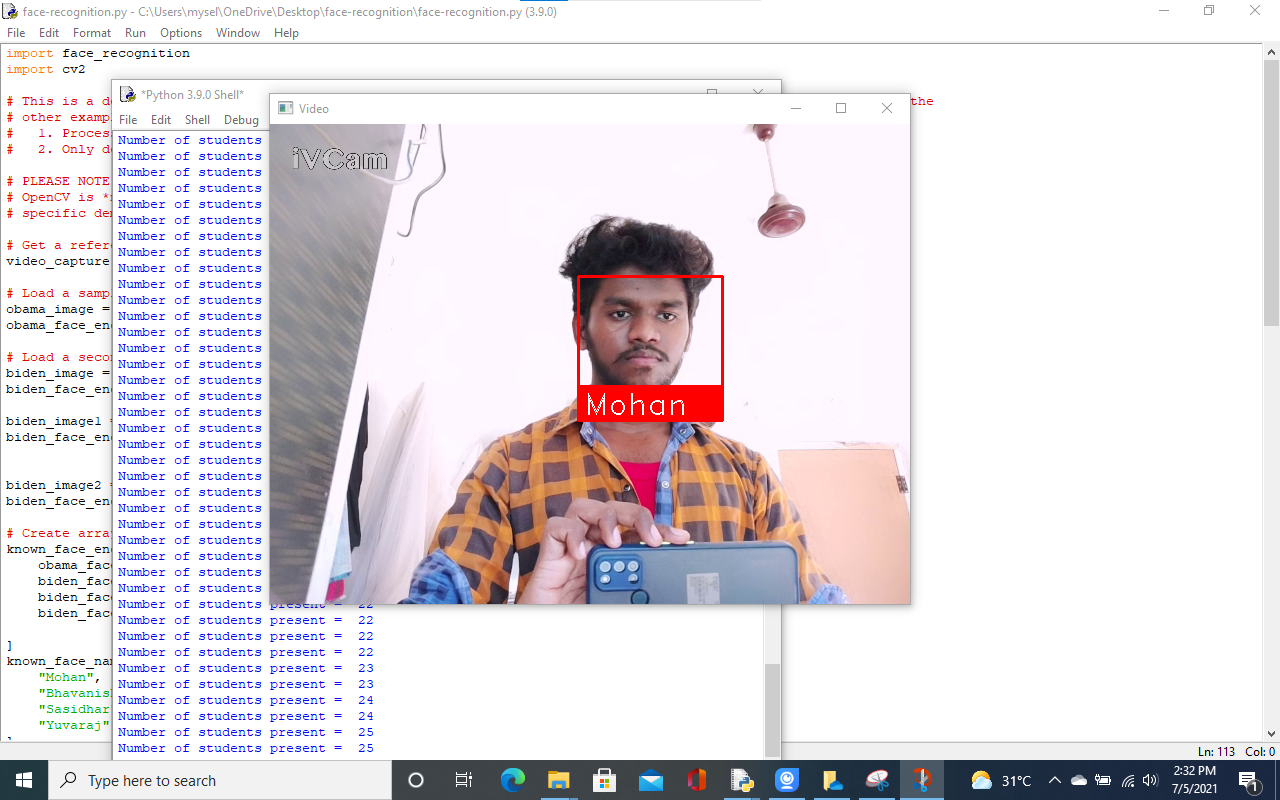
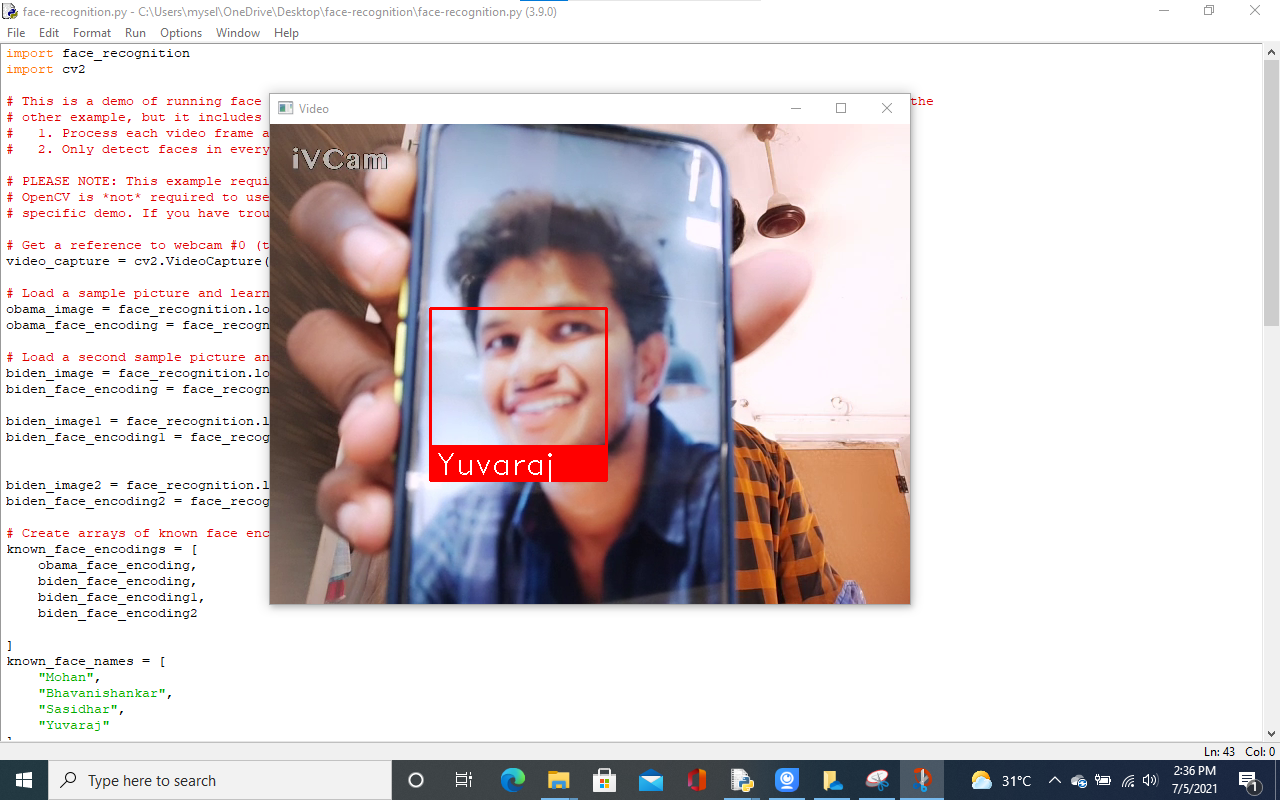
# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()

**9.2 UI Output Screenshot:**

****

**Shell o****u****put:**