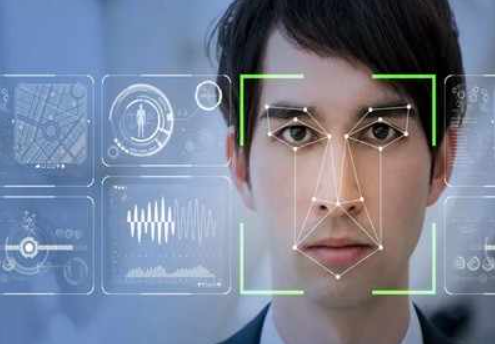
**Introduction:**

Nowadays Attendance is considered as an important factor for both the student as well as the teacher of an educational organization. With the advancement of the deep learning technology the machine automatically detects the attendance performance of the students and maintains a record of those collected data. In general, the attendance system of the student can be

* Manually
* Face recognition attendance system



Manual Student Attendance Management system is a process where a teacher concerned with the particular subject need to call the students name and mark the attendance manually. Manual attendance may be considered as a time-consuming process or sometimes it happens for the teacher to miss someone or students may answer multiple times on the absence of their friends. So, the problem arises when we think about the traditional process of taking attendance in the classroom. To solve all these issues we go with face recognition attendance system.

1. **Scope and objectives:**

The main objective of this project is to offer system that simplify and automate the process of recording attendance through face recognition technology. It is biometric technology to identify or verify a person from a digital image and to provide contact less attendance.

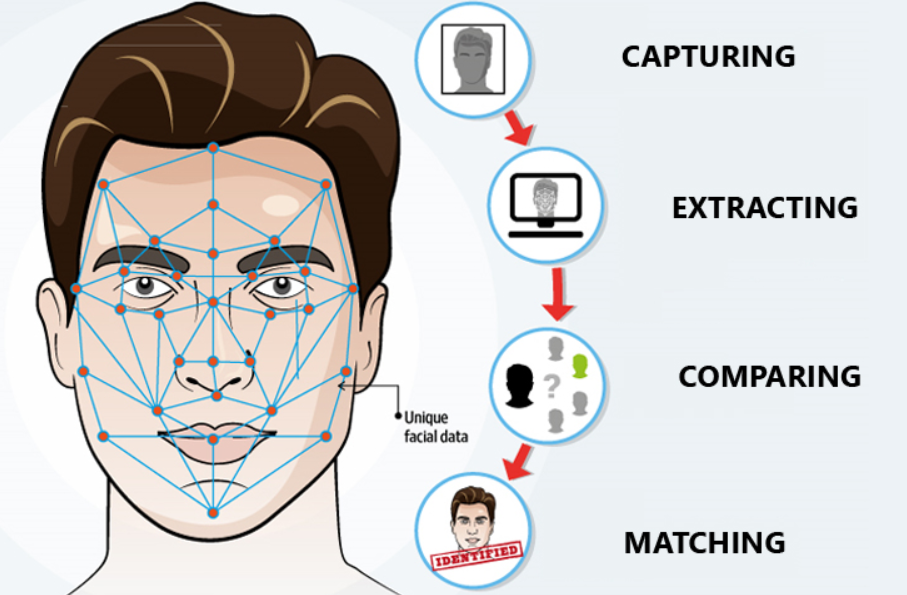
As we know that in this time of pandemic nobody wants to touch anything out there with their hands.

So to come up with this problem we came up with the solution that we can use face recognition technology that help us to take attendance without touching anything.

 Project **s**cope Provides facility for the automated attendance of students. Uses live face recognition to recognize each individual and mark their attendance automatically. Utilizes image processing to provide inputs to the system. Facility of marking manual attendance

2.**Features:**

* Face detection
* Face recognition
* Matching index
* Connection with employee
* Real time

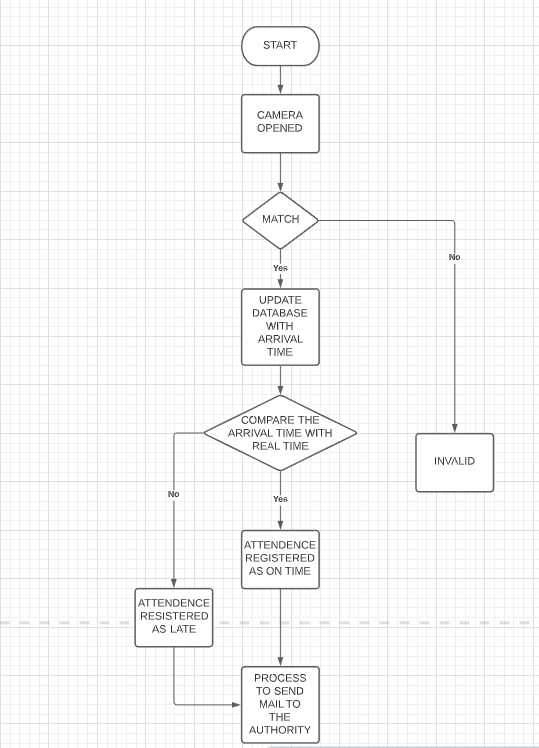


1. **PROBLEM STATEMENT :**

We conducted a survey in which we found many school offices uses finger biometric for taking attendance of their employees . And they think that it is a problem in this time of pandemic. As we know that in this time of pandemic nobody wants to touch anything out there with their hands.

According to reports “**The biometrics market will be significantly impacted by the COVID-19 pandemic”.**

1. **PROJECT DESIGN /ALGORITHM:**



1. **Libraries are imported to the program :**

* **Open cv2**

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.

* **numpy as np**

numpy is a python library and is written partially in python.Numpy stands for numerical python it is a python library used for working with arrays it also has functions for working in domain of linear algebra and matrices.we have lists that serve serve the purpose of arrays ,but they are slow to process.Numpy arrays are stored at one continuous place in memory unlike lists,so processes can acess and manpiluuate them very efficiently .

* **face\_recognition**

Recognize and manipulate faces from Python or from the command line with

the world’s simplest face recognition library.

* **Os**

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.

* **Cmake**

CMake supports in-place and out-of-place builds, and can therefore support multiple builds from a single source tree. CMake also supports static and dynamic library builds. ... The build process is controlled by creating one or more CMakeLists. txt files in each directory (including subdirectories) that make up a project.

* **Dlib**

Dlib is a general purpose [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) software [library](https://en.wikipedia.org/wiki/Library_(computing)) written in the programming language . Its design is heavily influenced by ideas from [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [component-based software engineering](https://en.wikipedia.org/wiki/Component-based_software_engineering). Thus it is, first and foremost, a set of independent software components. It is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) released under a [Boost Software License](https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries)#License).

* **Smtplib**

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. ... An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations.

* **Datetime**

The[datetime](https://docs.python.org/3/library/datetime.html#module-datetime) module supplies classes for manipulating dates and times.

While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

* **Mime types**

The [mimetypes](https://docs.python.org/3/library/mimetypes.html" \l "module-mimetypes" \o "mimetypes: Mapping of filename extensions to MIME types.) module converts between a filename or URL and the MIME type associated with the filename extension. Conversions are provided from filename to MIME type and from MIME type to filename extension; encodings are not supported for the latter conversion.

The module provides one class and a number of convenience functions. The functions are the normal interface to this module, but some applications may be interested in the class as wel

* **MIMEMultipart**

This module is part of the legacy (Compat32) email API. Its functionality is partially replaced by the [contentmanager](https://docs.python.org/3/library/email.contentmanager.html" \l "module-email.contentmanager" \o "email.contentmanager: Storing and Retrieving Content from MIME Parts) in the new API, but in certain applications these classes may still be useful, even in non-legacy code.

Ordinarily, you get a message object structure by passing a file or some text to a parser, which parses the text and returns the root message object. However you can also build a complete message structure from scratch, or even individual [Message](https://docs.python.org/3/library/email.compat32-message.html#email.message.Message) objects by hand. In fact, you can also take an existing structure and add new [Message](https://docs.python.org/3/library/email.compat32-message.html#email.message.Message) objects, move them around, etc. This makes a very convenient interface for slicing-and-dicing MIME messages.

* **Encoders**

The [email](https://docs.python.org/3/library/email.html#module-email) package provides some convenient encoders in its encoders module. These encoders are actually used by the [MIMEAudio](https://docs.python.org/3/library/email.mime.html" \l "email.mime.audio.MIMEAudio" \o "email.mime.audio.MIMEAudio) and [MIMEImage](https://docs.python.org/3/library/email.mime.html" \l "email.mime.image.MIMEImage" \o "email.mime.image.MIMEImage) class constructors to provide default encodings. All encoder functions take exactly one argument, the message object to encode. They usually extract the payload, encode it, and reset the payload to this newly encoded value. They should also set the Content-Transfer-Encoding header as appropriate.

* **Message**

**Messages** is a package designed to make sending **messages** easy! **Messages** wraps various standard **library module**, third-party **module**, web app API calls, etc. all in one package. **Messages** can send **messages** asynchronously.

* **MIMEBase**

This is the base class for all the MIME-specific subclasses of [Message](https://docs.python.org/3/library/email.compat32-message.html#email.message.Message). Ordinarily you won’t create instances specifically of [MIMEBase](https://docs.python.org/3/library/email.mime.html" \l "email.mime.base.MIMEBase" \o "email.mime.base.MIMEBase), although you could. [MIMEBase](https://docs.python.org/3/library/email.mime.html" \l "email.mime.base.MIMEBase" \o "email.mime.base.MIMEBase) is provided primarily as a convenient base class for more specific MIME-aware subclasses.

* **MIMEText**

the [MIMEText](https://docs.python.org/3/library/email.mime.html" \l "email.mime.text.MIMEText" \o "email.mime.text.MIMEText) class is used to create MIME objects of major type text. \_text is the string for the payload. \_subtype is the minor type and defaults to plain. \_charset is the character set of the text and is passed as an argument to the [MIMENonMultipart](https://docs.python.org/3/library/email.mime.html" \l "email.mime.nonmultipart.MIMENonMultipart" \o "email.mime.nonmultipart.MIMENonMultipart) constructor; it defaults to us-ascii if the string contains only ascii code points, and utf-8 otherwise. The \_charset parameter accepts either a string or a [Charset](https://docs.python.org/3/library/email.charset.html#email.charset.Charset) instance.

1. **How to install libraries:**

Install the libraries in the order from command prompt (administration mode)

* pip install face-recognition
* pip install cmake
* pip install dlib
* pip install numpy
* pip install opencv
* pip install smtplib
* pip install datetime
* pip install  [mimetypes](https://docs.python.org/3/library/mimetypes.html" \l "module-mimetypes" \o "mimetypes: Mapping of filename extensions to MIME types.)
* pip install MIMEMultipart
* pip install encoding
* pip install message
* pip install MIMEBase
* pip install MIMEText

1. **PROJECT CODING:**

*import cv2*

*import numpy as np*

*import face\_recognition*

*import os*

*from datetime import datetime*

*import smtplib*

*path = 'Employee Images' #images folder*

*images = [] #import images*

*classNames = []*

*myList = os.listdir(path)*

*print(myList)#printing names of images*

*for cl in myList:*

*curImg = cv2.imread (f'{path}/{cl}')# read the images in the path*

*images.append(curImg)*

*classNames.append(os.path.splitext(cl)[0])# to show the name of the image without the extension*

*print(classNames)*

*def findEncodings(images):*

*#images are converted into a code*

*encodeList = []*

*for img in images:*

*img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)*

*encode = face\_recognition.face\_encodings(img)[0]*

*encodeList.append(encode)*

*return encodeList*

*def markAttendance(name):*

*with open('Attendance.csv','r+') as f:*

*myDataList = f.readlines()*

*nameList = []*

*for line in myDataList:*

*entry = line.split(',')*

*nameList.append(entry[0])*

*if name not in nameList:*

*now = datetime.now()*

*dtString = now.strftime('%H:%M:%S')*

*today10am = now.replace(hour=10, minute=00, second=0, microsecond=0)*

*if now < today10am :*

*f.writelines(f'\n{name},{dtString},on time')*

*if now == today10am :*

*f.writelines(f'\n{name},{dtString},on time')*

*if now > today10am :*

*f.writelines(f'\n{name},{dtString},late ')*

*encodeListKnown = findEncodings(images)*

*print('Encoding Complete')*

*cap = cv2.VideoCapture(0)*

*while True:*

*success, img = cap.read()*

*imgS = cv2.resize(img,(0,0),None,0.25,0.25)*

*imgS = cv2.cvtColor(imgS, cv2.COLOR\_BGR2RGB)*

*facesCurFrame = face\_recognition.face\_locations(imgS)*

*encodesCurFrame = face\_recognition.face\_encodings(imgS,facesCurFrame)*

*for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):*

*matches = face\_recognition.compare\_faces(encodeListKnown,encodeFace)*

*faceDis = face\_recognition.face\_distance(encodeListKnown,encodeFace)*

*#print(faceDis)*

*matchIndex = np.argmin(faceDis)*

*if matches[matchIndex]:*

*name = classNames[matchIndex].upper()*

*#print (name)*

*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)*

*markAttendance(name)*

*cv2.imshow('Webcam',img)*

*cv2.waitKey(1)*

*key = cv2.waitKey(1)*

*if key == 27:*

*break*

*#for sending mail to the manager(HR)*

*import smtplib*

*import mimetypes*

*from email.mime.multipart import MIMEMultipart*

*from email import encoders*

*from email.message import Message*

*from email.mime.base import MIMEBase*

*from email.mime.text import MIMEText*

*emailfrom = ""*

*emailto = " mayank2000joshi@gmail.com "*

*fileToSend = "Attendance.csv"*

*username = "aiproject6923@gmail.com"*

*password = "Aiproject6923"*

*msg = MIMEMultipart()*

*msg["From"] = emailfrom*

*msg["To"] = emailto*

*msg["Subject"] = "file"*

*ctype, encoding = mimetypes.guess\_type(fileToSend)*

*if ctype is None or encoding is not None:*

*ctype = "application/octet-stream"*

*maintype, subtype = ctype.split("/", 1)*

*if maintype == "text":*

*fp = open(fileToSend)*

*# Note: we should handle calculating the charset*

*attachment = MIMEText(fp.read(), \_subtype=subtype)*

*fp.close()*

*else:*

*fp = open(fileToSend, "rb")*

*attachment = MIMEBase(maintype, subtype)*

*attachment.set\_payload(fp.read())*

*fp.close()*

*encoders.encode\_base64(attachment)*

*attachment.add\_header("Content-Disposition", "attachment", filename=fileToSend)*

*msg.attach(attachment)*

*server = smtplib.SMTP("smtp.gmail.com:587")*

*server.starttls()*

*server.login(username,password)*

*server.sendmail(emailfrom, emailto, msg.as\_string())*

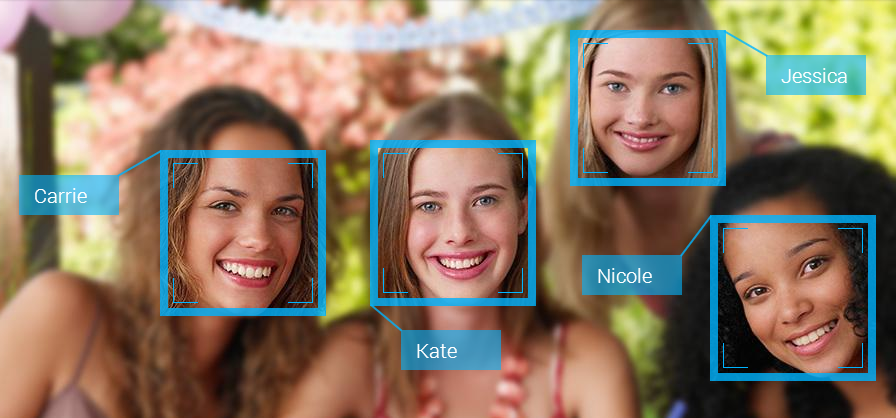
*server.quit()*

*cv2.destroyAllWindows()*

**Execution:**

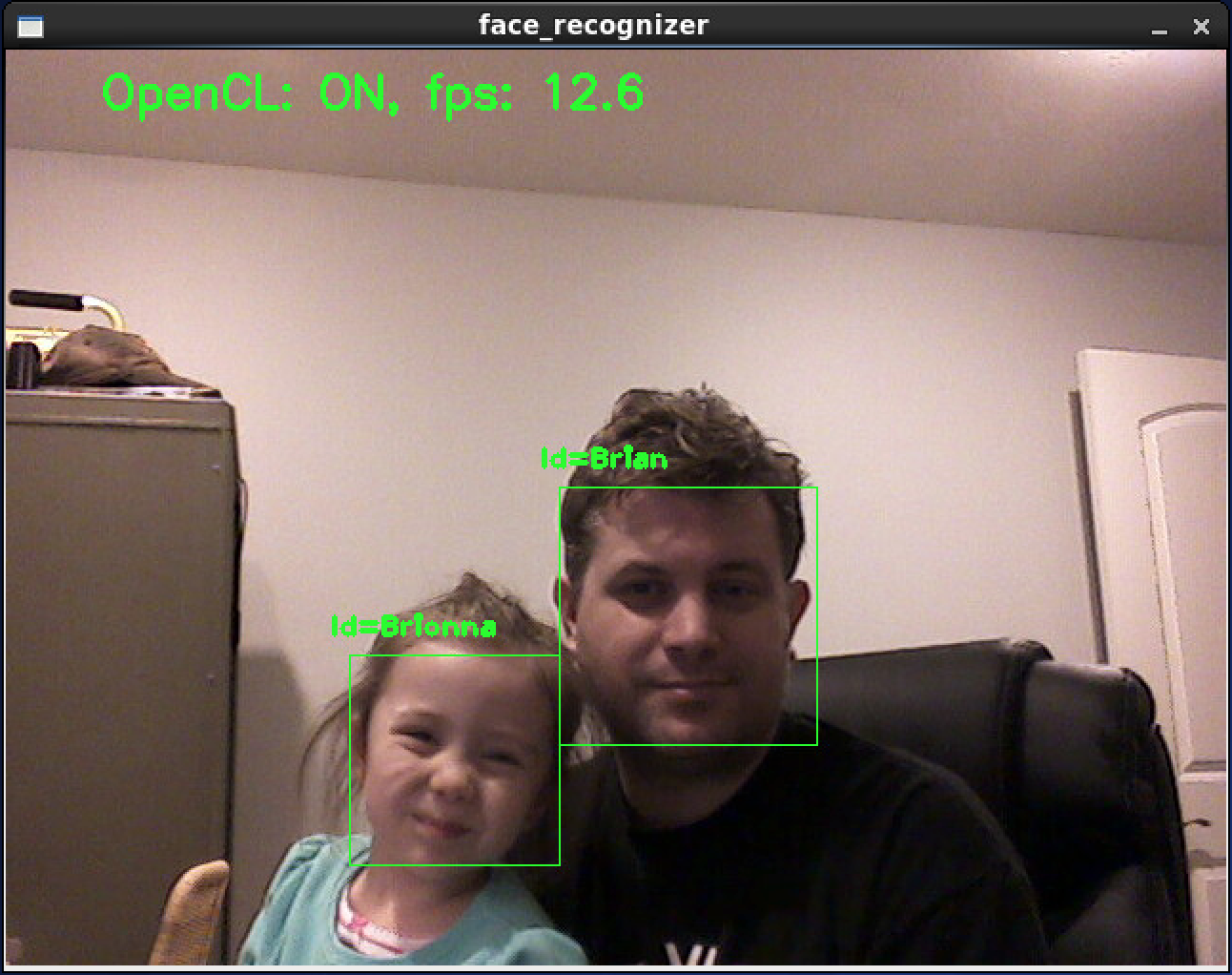
1.After execution the web

cam will open and detect face Infront of it.

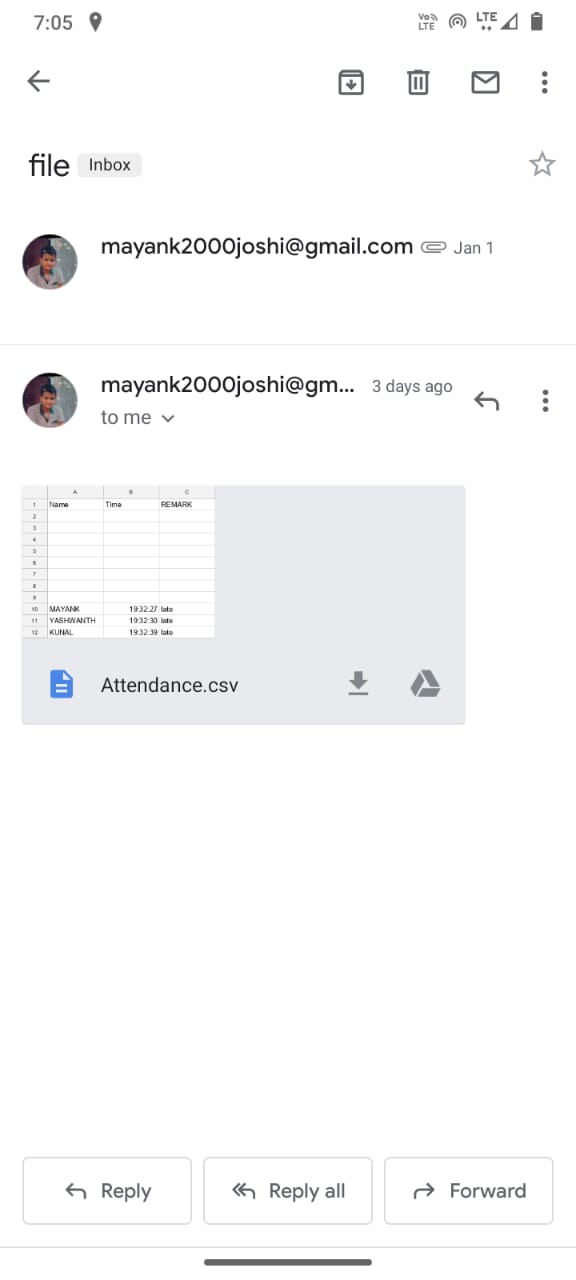


2.And after detecting individual face it will mark the attendance with time in the file itself.





3.After the program is terminated the csv file is sent to the provided mail id.



**9.Application Areas:**

* Access Control
* Airport Security
* Financial Services
* Law Enforcement
* Driver's Licenses & Passport

**11.ADVANTAGES:**

* **Automated time tracking system**
* **Cost-effective**
* **Increased security**
* **Time saving**
* **Easy to manage**

**12.DISADVANTAGES:**

* **If in the wrong hands, it will be a disaster**
* **Data privacy breach**
* **Low reliability**
* **Lack of regulations in the AI in Face recognition systems**

**13.CONCLUSION:**

From this we conclude that to detect real time human face are used and a simple fast Principal Component Analysis has used to recognize the faces detected with a high accuracy rate. The matched face is used to mark attendance of the employee. Our system maintains the attendance records of employees automatically and send mail to the authority after end of the session.

 THANK YOU