FACE RECOGNITION ATTENDANCE SYSTEM

## A report submitted for the partial fulfillment in requirement for the Degree of Bachelor of Science

*In Computer Science*

at

# West Bengal State University



*Submitted by*

|  |  |  |
| --- | --- | --- |
| **Labani Das** | **Sahim Khan** | **Rahul Kumar Shaw** |
| **Roll No:** *623213419578* | **Roll No:** *623110706582* | **Roll No:** *623110706593* |
| **Reg No :** *1032021100062* | **Reg No :** *1032012600053* | **Reg No :** *1032011400054* |

*Under the supervision of*

**Dr. Anirban Chakraborty**

*Department of Computer Science Barrackpore Rastraguru Surendranath College affiliated to West Bengal State University, Barasat*

### July. 2023

**Declaration**

*We solemnly declare that the contents of this report, titled "Face recognition attendance system" represent our project work (Project Code: CMSADSE06P) conducted under the guidance*

*and supervision of Dr. Anirban Chakraborty from the Department of Computer Science, Barrackpore Rastraguru Surendranath College. This report is submitted as a partial fulfillment of the requirements for the degree of B.Sc in Computer Science (CBCS System) for the academic year 2020-2023, at Barrackpore Rastraguru Surendranath College, affiliated with West Bengal State University, Barasat.*

*We also declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct, and the results presented in this report or parts of it have not been presented for the award of any other degree.*

|  |  |  |
| --- | --- | --- |
| **Name** | **Registration** | **Signature** |
| **Labani Das** | *1032021100062* |  |
| **Sahim Khan** | *1032012600053* |  |
| **Rahul Kumar Shaw** | *1032011400054* |  |

### CERTIFICATE

*Depart of Computer Science*

*This is to certify that the project titled "****Face recognition attendance system****” : A Comprehensive Study" is completed by* ***Labani Das*** *(Roll No : 623213419578, Reg No: 1032021100062),* ***Sahim Khan*** *(Roll No: 623110706582, Reg No : 1032012600053) and* ***Rahul Kumar Shaw*** *(Roll No : 623110706593, Reg No : 1032011400054 ) during the academic year 2020-2023. This work is a genuine and conducted as part of the requirements for the degree of* ***B.Sc in Computer Science*** *at* ***Barrackpore Rastraguru Surendranath College****, affiliated with* ***West Bengal State University****. This project is submitted in fulfillment of the partial requirements for the aforementioned degree and reflects the student’s dedication and knowledge in the field of Computer Science.*

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

(***Sri. Phulen Mahato***) (***Dr. Anirban Chakraborty***)

HEAD SUPERVISOR

Department of Computer Science Department of Computer Science Barrackpore Rastraguru Surendranath Barrackpore Rastraguru Surendranath

College. College.

- - - - - - - - - - - - - - - - - - - - - - - - - - (EXTERNAL EXAMINER)

# ACKNOWLEDGEMENT

Undertaking this dissertation work as an essential component of our academic curriculum has been an enriching and scientifically progressive experience. This research effort has not only widened our knowledge horizons but also propelled us towards a greater understanding of our chosen field.

In the course of preparing this dissertation and shaping its form, we acknowledge with profound gratitude the invaluable contributions of several individuals and institutions. Our heartfelt appreciation goes to Dr. Monojit Roy (Principal, Barrackpore Rastraguru Surendranath College), Sri. Phulen Mahato (Head of Department, Computer Science), and Dr. Anirban Chakraborty (Department of Computer Science) for their valuable guidance, constant support, and continuous motivation throughout this effort.

Special thanks are extended to our dedicated supervisor, Dr. Anirban Chakraborty, whose expertise and encouragement have been instrumental in shaping the direction of our project.

We also express our sincere gratitude to all the faculty members, lab assistants, and administrative staff of the university for their generous assistance and timely help whenever needed.

In conclusion, we would like to acknowledge the contributions of all those who played a part, big or small, in making this dissertation work a reality.

Table of Contents

[INTRODUCTION 6](#_Toc142041223)

[1.1 Introduction 7](#_Toc142041224)

[1.2. History 7](#_Toc142041225)

[1.3. Problem Statement 8](#_Toc142041226)

[1.4. Aims And Objective 9](#_Toc142041227)

[PRELIMINARIES 10](#_Toc142041228)

[2.1. Digital Image Processing 11](#_Toc142041229)

[2.2. Image Representation In A Digital Computer 11](#_Toc142041230)

[**2.3.** Steps In Digital Image Processing 11](#_Toc142041231)

[2.4. Definition Of Terms And History 12](#_Toc142041232)

[2.5. Face Recognition Algorithms 13](#_Toc142041233)

[CODE METHODOLOGY 17](#_Toc142041234)

[3.1. Terminilogies Used 18](#_Toc142041235)

[3.2. Algorithm Used 19](#_Toc142041236)

[3.2.1. Hog (Histogram Of Oriented Gradients) Algorithm 19](#_Toc142041237)

[3.3. Steps In Face Recognition 21](#_Toc142041238)

[3.4. Model Implementation And Requirements 25](#_Toc142041239)

[3.4.1. Implementation 25](#_Toc142041240)

[3.4.2. Software Requirement 26](#_Toc142041241)

[3.4.3. Hardware Requirement 26](#_Toc142041242)

[3.5. System Flow Chart 27](#_Toc142041243)

[3.6. Screenshots Of Working System 29](#_Toc142041244)

[3.6.1. Registering Students/ Employees 29](#_Toc142041245)

[3.6.2. Taking Attendance 29](#_Toc142041246)

[END USER STORY 33](#_Toc142041247)

[4.1. Definition 34](#_Toc142041248)

[4.2. User Story 34](#_Toc142041249)

[BENEFITS AND LIMITATIONS 35](#_Toc142041250)

[CONCLUSION AND FUTURE SCOPE 37](#_Toc142041251)

[REFERENCES 39](#_Toc142041252)

[APPENDIX 41](#_Toc142041253)

[8.1. Code For For Registering Student 42](#_Toc142041254)

[8.2. Code For Face Attendance System Code 46](#_Toc142041255)

Chapter 1

# INTRODUCTION

## 1.1 INTRODUCTION

Attendance is prime important for both the teacher and student of an educational organization. So it is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room. Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. So an automatic attendance system can solve all above problems.

There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique and RFID system. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking. This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure.

The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions. 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. HISTORY

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.

Robinson-Riegler, Face Recognition Attendance System Page 7 of 54 G., & Robinson-Riegler, B. (2008) mentioned that after visual processing done by the human visual system, we actually classify shape, size, contour and the texture of the object in order to analyze the information. The analyzed information will be compared to other representations of objects or face that exist in our memory to recognize. In fact, it is a hard challenge to build an automated system to have the same capability as a human to recognize faces. However, we need large memory to recognize different faces, for example, in the Universities, there are a lot of students with different race and gender, it is impossible to remember every face of the individual without making mistakes.

In order to overcome human limitations, computers with almost limitless memory, high processing speed and power are used in face recognition systems. 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.

Nowadays, face recognition system is prevalent due to its simplicity and awesome performance. For instance, airport protection systems and FBI use face recognition for criminal investigations by tracking suspects, missing children and drug activities . Apart from that, Facebook which is a popular social networking website implement face recognition to allow the users to tag their friends in the photo for entertainment purposes.

## 1.3. PROBLEM STATEMENT

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as calling student names or checking respective identification cards. There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names, attendance sheet is passed around the classroom during the lecture sessions.

The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class. Thus, face recognition attendance system is proposed in order to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance.

Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

## 1.4. AIMS AND OBJECTIVE

The objective of this project is to develop face recognition 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.

Chapter 2

# PRELIMINARIES

## 2.1. DIGITAL IMAGE PROCESSING

Digital Image Processing is the processing of images which are digital in nature by a digital computer[2]. Digital image processing techniques are motivated by three major applications mainly:

• Improvement of pictorial information for human perception

• Image processing for autonomous machine application

• Efficient storage and transmission.

## 2.2. Image Representation in a Digital Computer

An image is a 2-Dimensional light intensity function 𝐟 (𝐱,𝐲) = 𝐫 (𝐱,𝐲) × 𝐢 (𝐱,𝐲) - (2.0) Where, r (x, y) is the reflectivity of the surface of the corresponding image point. i (x,y) Represents the intensity of the incident light. A digital image f(x, y) is discretized both in spatial co-ordinates by grids and in brightness by quantization.

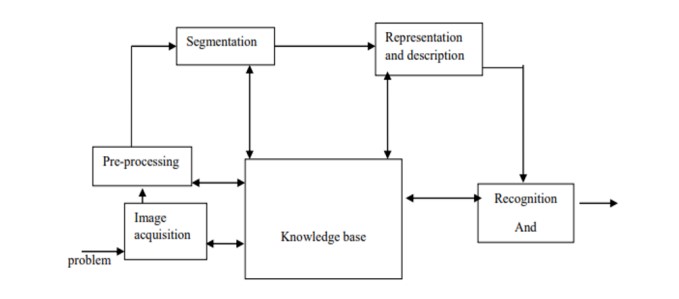
Effectively, the image can be represented as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point. These elements are referred to as pixels or pels. Typically following image processing applications, the image size which is used is𝟐𝟓𝟔 × 𝟐𝟓𝟔, elements, 𝟔𝟒𝟎 × 𝟒𝟖𝟎 pels or 𝟏𝟎𝟐𝟒 × 𝟏𝟎𝟐𝟒 pixels.

Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for colored images (because of the three color planes Red, Green and Blue each at 8 bits)

## **2.3.** STEPS IN DIGITAL IMAGE PROCESSING

Digital image processing involves the following basic tasks:

* Image Acquisition - An imaging sensor and the capability to digitize the signal produced by the sensor.
* Preprocessing – Enhances the image quality, filtering, contrast enhancement etc.
* Segmentation – Partitions an input image into constituent parts of objects.
* Description/feature Selection – extracts the description of image objects suitable for further computer processing.
* Recognition and Interpretation – Assigning a label to the object based on the information provided by its descriptor. Interpretation assigns meaning to a set of labelled objects.
* Knowledge Base – This helps for efficient processing as well as inter module cooperation.



A diagram showing the steps in digital image processing

## 2.4. DEFINITION OF TERMS AND HISTORY

Face Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content

Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a three dimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images.

Face Recognition is therefore simply the task of identifying an already detected face as a known or unknown face and in more advanced cases telling exactly whose face it is.

Difference between Face Detection and Face Recognition

Face detection answers the question, Where is the face? It identifies an object as a “face” and locates it in the input image. Face Recognition on the other hand answers the question who is this? Or whose face is it? It decides if the detected face is someone known or unknown based on the database of faces it uses to validate this input image.It can therefore be seen that face detections output (the detected face) is the input to the face recognizer and the face Recognition’s output is the final decision i.e. face known or face unknown.

Face Detection

A face Detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. Face detection can be performed based on several cues: skin color (for faces in color images and videos, motion (for faces in videos), facial/head shape, facial appearance or a combination of these parameters. Most face detection algorithms are appearance based without using other cues. An input image is scanned at all possible locations and scales by a sub window. Face detection is posed as classifying the pattern in the sub window either as a face or a non-face. The face/nonface classifier is learned from face and non-face training examples using statistical learning methods. Most modern algorithms are based on the Viola Jones object detection framework, which is based on Haar Cascades.

## 2.5. FACE RECOGNITION ALGORITHMS

FACE LANDMARK ESTIMATION ALGORITHM

Facial landmark detection algorithms help to automatically identify the locations of the facial key landmark points on a facial image or from a video. The key landmark points normally includes the facial regions like nose tip, eye corner, eye brows and chin tip. Some applications of facial landmark detection are face swap, head pose detection, detecting facial gestures, gaze direction etc.

1. Face detection

Face detection locate a human face from the image and return a value in terms of the co-ordinates of the bounding rectangle of the detected face.

2. Landmarks detection

After detecting the face, we have go through points inside the bounding rectangle for key points.

It returns different 68 feature points from a face.

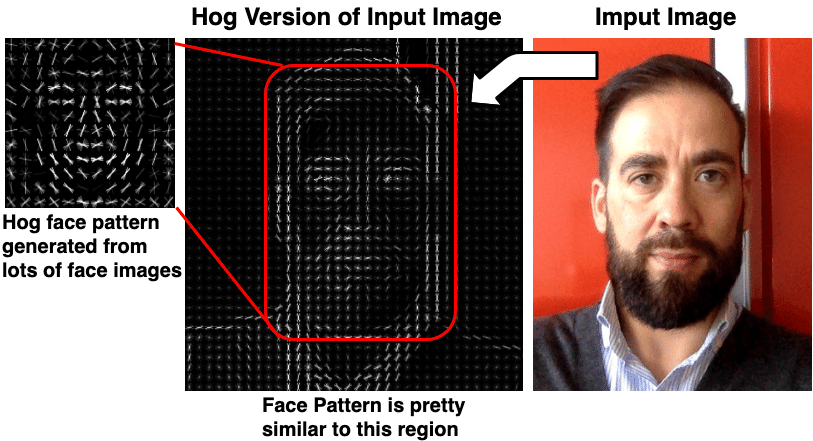
## {3E7ACB67-889B-4782-A70A-B9351ADFBAD7}.png.jpg

landmark points on face

HOG (HISTOGRAM OF ORIENTED GRADIENTS) ALGORITHM

HOG is a simple and powerful feature descriptor. It is not only used for face detection but also it is widely used for object detection like cars, pets, and fruits. HOG is robust for object detection because object shape is characterized using the local intensity gradient distribution and edge direction.

**The Histogram of Oriented Gradients (HOG) algorithm for face detection involves dividing the image into connected cells (Step 1). For each cell, histograms are computed (Step 2), and these histograms are then combined to form a unique feature vector for each face, representing its distinctive characteristics (Step 3). This process allows HOG to effectively detect and differentiate faces in images based on local gradients and orientations, enabling successful face recognition tasks.**

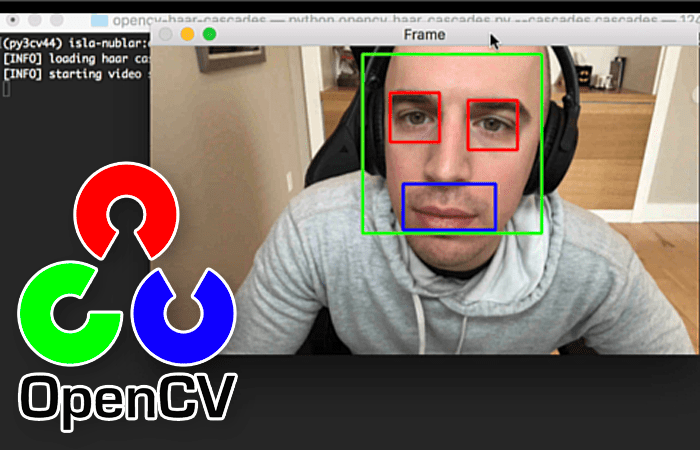
****

HAAR CSCADE

A Haar classifier, or a Haar cascade classifier, is a machine learning object detection program that identifies objects in an image and video.

The algorithm can be explained in four stages:

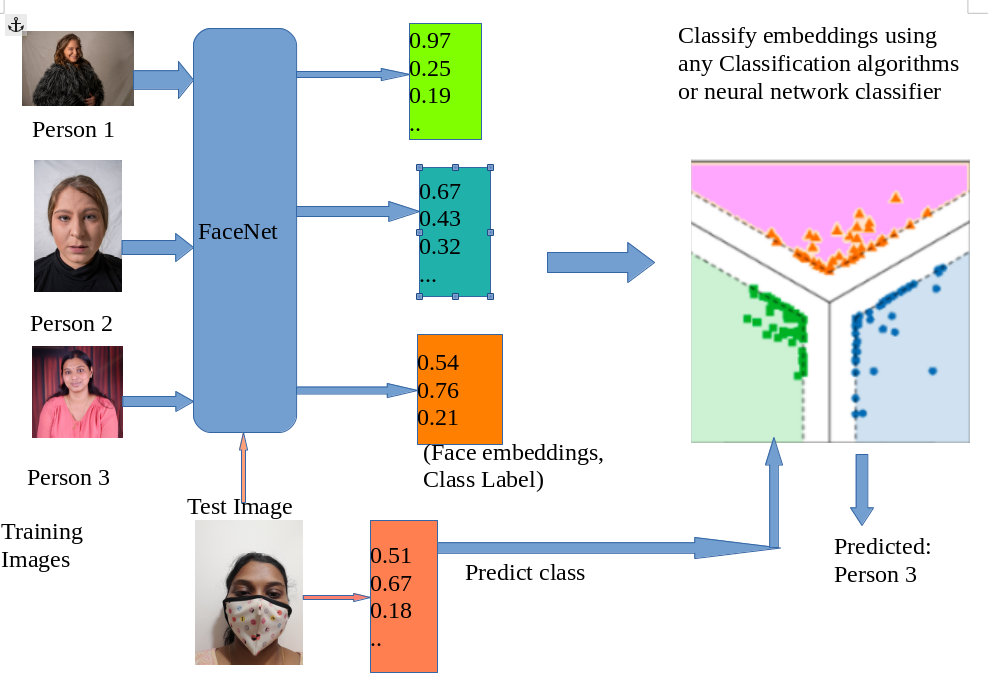
1. Calculating Haar Features
2. Creating Integral Images
3. Using Adaboost
4. Implementing Cascading Classifiers



*Facial features detected using Haar cascade Classifier*

FACENET

FaceNet is a deep learning algorithm developed by Google that learns to map faces to a 128-dimensional vector space, known as face embeddings. It uses a triplet loss function to ensure that similar faces are clustered closely together, enabling accurate face recognition and verification. FaceNet's embeddings facilitate efficient face matching, allowing it to recognize faces with high accuracy, even across large databases, making it a pivotal advancement in face recognition technology.



Face recognition walkthrough using faceNet

Chapter 3

# **METHODOLOGY**

## 3.1. TERMINILOGIES USED

Shear

Shearing in computer graphics is a transformation that skews or distorts the shape of an object along a particular axis. It involves shifting each point in a geometric object in a direction perpendicular to that axis proportional to its distance from the axis. This process alters the shape of the object while maintaining its size along the axis not being sheared. Shearing is commonly used in 2D graphics to create the illusion of perspective or to create special effects in image manipulation and animation.

Neural Network

In the context of extracting 128 measurements from a face for face recognition, a neural network refers to a deep learning model specifically designed to learn and represent facial features in a high-dimensional feature space. This type of neural network is commonly known as a "Siamese network" or "FaceNet."

The FaceNet neural network takes an input face image and processes it through several layers of convolutional and fully connected neural units. These layers learn to extract relevant facial features and transform the face into a fixed-length vector of 128 measurements, often referred to as an "embedding" or "feature representation." This embedding encodes unique characteristics of the face, making it suitable for face recognition tasks, where similar faces are mapped close together in the feature space, and dissimilar faces are mapped far apart. This way, the neural network can distinguish between different individuals based on the extracted feature representations. The network is trained on a large dataset of labeled face images, using techniques like contrastive loss or triplet loss, to optimize the embeddings' clustering and separation properties, ensuring accurate face recognition.

Histogram

In digital image processing, a histogram is a graphical representation of the distribution of pixel intensity values in an image. It displays the frequency of occurrence of each pixel intensity level, providing valuable insights into the image's overall brightness and contrast, aiding in various image enhancement and analysis techniques.

## 3.2. ALGORITHM USED

### 3.2.1. HOG (HISTOGRAM OF ORIENTED GRADIENTS) ALGORITHM

HOG is a simple and powerful feature descriptor. It is not only used for face detection but also it is widely used for object detection like cars, pets, and fruits. HOG is robust for object detection because object shape is characterized using the local intensity gradient distribution and edge direction.

**Step1:** The basic idea of HOG is dividing the image into small connected cells

## {841AEAFD-88A0-4CF4-B4BE-32358C73A3DB}.png.jpg

**Step2:**Computes histogram for each cell.

## {BD13B483-8AEE-4B5D-8F0F-470142F39C90}.png.jpg

**Step3:**Bring all histograms together to form feature vector i.e., it forms one histogram from all small histograms which is unique for each face

## {19B3A3C7-EE60-4FD6-B001-1DFA61885FE4}.png.jpg

Below is the **HOG** algorithm or pseudocode is written:

1. Initialize parameters:

   - Cell size (size of local regions for computing gradients)

   - Block size (size of blocks for normalization)

   - Number of orientations (number of bins for histogram)

2. Load the **input** image.

3. Convert the image to grayscale.

4. Divide the image into cells of the specified size.

5. For each cell:

   5.1 Compute the gradients (direction and magnitude) of pixel intensity changes.

   5.2 Create a histogram of gradient orientations in the cell using the specified number of bins.

   5.3 Normalize the histogram to account for changes in lighting and contrast.

6. Divide the image into blocks of the specified size, overlapping by a certain stride.

7. For each block:

   7.1 Concatenate the histograms of **all** cells within the block.

   7.2 Normalize the concatenated histogram.

8. Combine the normalized block histograms to form the final feature vector.

9. Use a sliding window approach to move the detection window across the image.

10. For each position of the window:

    10.1 Extract the HOG features from the region covered by the window.

    10.2 Use a classifier (e.g., support vector machine) to determine if a face is present in the region.

11. If a face is detected, mark the position of the window as a potential face region.

12. Repeat steps 9 to 11 until the entire image has been scanned.

13. Return the positions of **all** potential face regions.

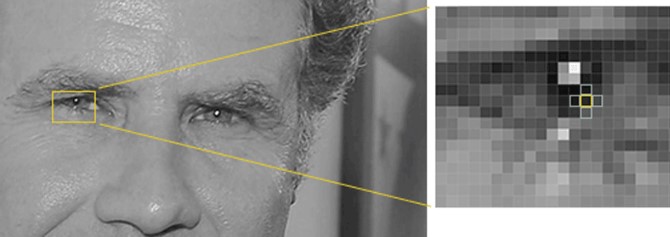
## 3.3. STEPS IN FACE RECOGNITION

There are four steps for revognising face:- finding a face first of all, then get the face as straight as pisible from posing faces, encoding faces and finding the person’s identity from the encoding



**Step 1:** Finding all the Faces

The first step in our pipeline is face detection. To find faces in an image, we’ll start by making our image black and white because we don’t need color data to find faces:

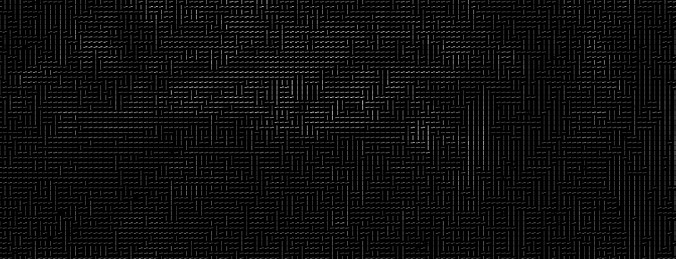


Then we’ll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it

## **{BFD66575-B5B4-44CF-93D0-9B8954209260}.png.jpg**

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:

If you repeat that process for **every single pixel** in the image, you end up with every pixel being replaced by an arrow. These arrows are called **gradients** and they show the flow from light to dark across the entire image:



gradients showing the flow from light to dark across the entire image

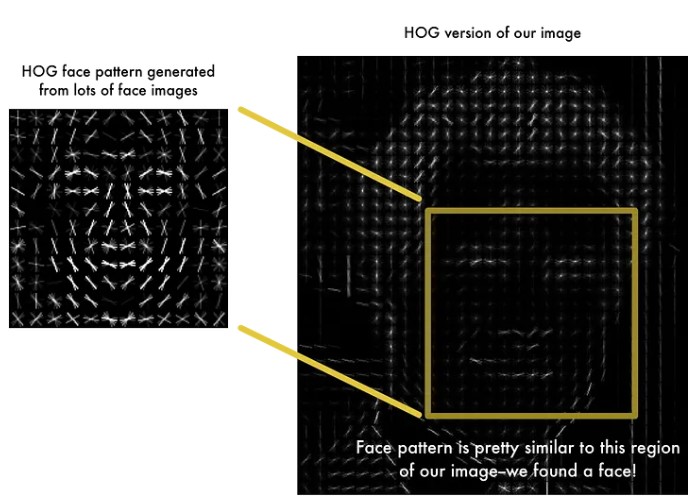
There’s a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the *direction* that brightness changes, both really dark images and really bright images will end up with the same exact representation.

But saving the gradient for every single pixel gives us way too much detail. We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

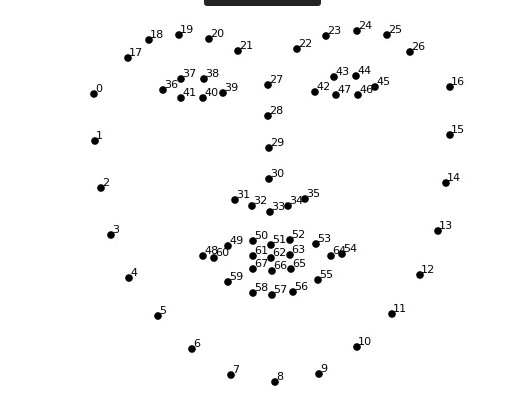
To do this, we’ll break up the image into small squares of 16x16 pixels each. In each square, we’ll count up how many gradients point in each major direction (how many point up, point up-right, point right, etc…). Then we’ll replace that square in the image with the arrow directions that were the strongest. The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way:



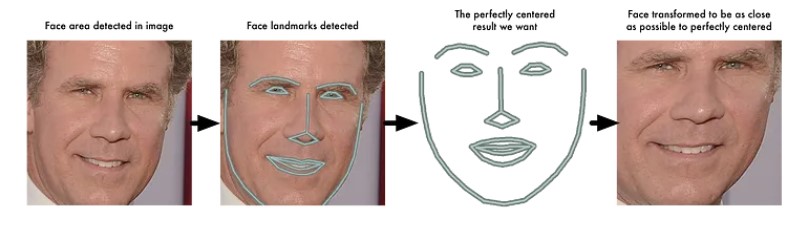
The original image is turned into a HOG representation that captures the major features of the image regardless of image brightnesss.

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces:

Step 2: Posing and Projecting Faces

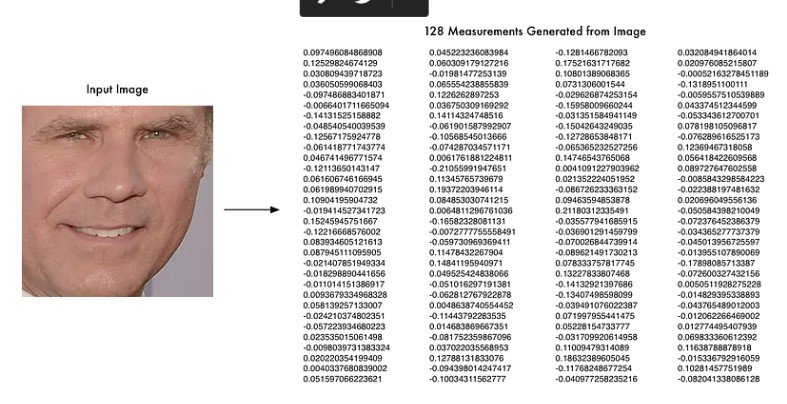
Whew, we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer. To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps. To do this, we are going to use an algorithm called **face landmark estimation**.

The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these **68 specific points** on any face:

Now that we know were the eyes and mouth are, we’ll simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible. 

Step 3: Encoding Faces

We need to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc. So all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. Here’s the measurements for our test image.



The network generates nearly the same numbers when looking at two different pictures of the same person.

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

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image.

## 3.4. MODEL IMPLEMENTATION AND REQUIREMENTS

### 3.4.1. IMPLEMENTATION

**1. Window Setup:**

- The code creates a resizable window named "Face Attendance" using OpenCV.

- The window is resized to fit the desired screen resolution.

**2. Loading Data:**

* The code imports student images from the "Database/Images" folder and stores them in a dictionary, where the keys are student IDs and the values are their respective images.
* It loads pre-registered face encodings and student IDs from the "EncodeFile.p" file.

**3. Face Recognition and Attendance:**

* The code uses the face\_recognition library to detect faces in the webcam feed and encode their facial features.
* For each detected face, it compares the face encodings with the pre-registered encodings to recognize known faces.
* If a known face is detected, it displays a bounding box around the face and loads relevant information from the CSV attendance file.
* If it's the first time detecting a student on a particular day, the code updates the attendance count and last attendance date for that student in the CSV file.
* It displays relevant student information and their image on the screen during the attendance update process.

**4. Mode Switching:**

* The code uses different modes (displayed images) to guide users through the attendance process.
* The modes include showing loading text, displaying student information during attendance, and showing a message after completing attendance for a student.

**5. User Interaction:**

* The code constantly loops through the webcam feed, updating the screen with detected faces, attendance status, and relevant student information.
* The user can press the 'q' key to quit the attendance system.

The implementation ensures that the system smoothly handles face recognition, attendance updating, and mode transitions, providing a user-friendly experience for recording student attendance.

### 3.4.2. SOFTWARE REQUIREMENT

* **PycharmIDE**

PyCharm is a powerful Python Integrated Development Environment (IDE) by JetBrains. It offers advanced coding assistance, intelligent code analysis, debugging, and a customizable interface. With a vast range of productivity features, PyCharm is a top choice for Python developers, enhancing efficiency and streamlining the development process.

* **Python ( version 3.7.6 )**

Python is a high-level, versatile programming language known for its simplicity and readability. With a vast standard library and a supportive community, Python is widely used in web development, data analysis, machine learning, automation, and more. Its ease of learning makes it an excellent choice for beginners and experienced developers alike.

### 3.4.3. HARDWARE REQUIREMENT

1. **Computer with Operating System**

Computer with either operating system can be given below

Windows: Widely used OS with user-friendly interface.

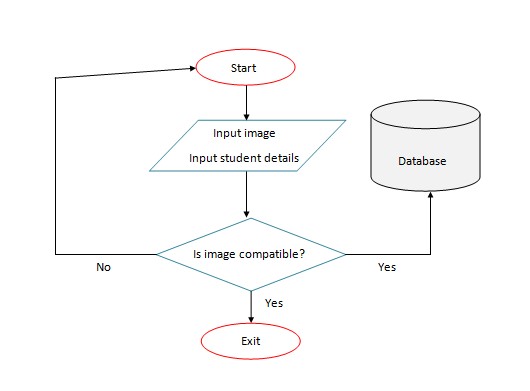
Linux: Open-source OS with customization options.

Mac: Apple's OS known for sleek design and performance.

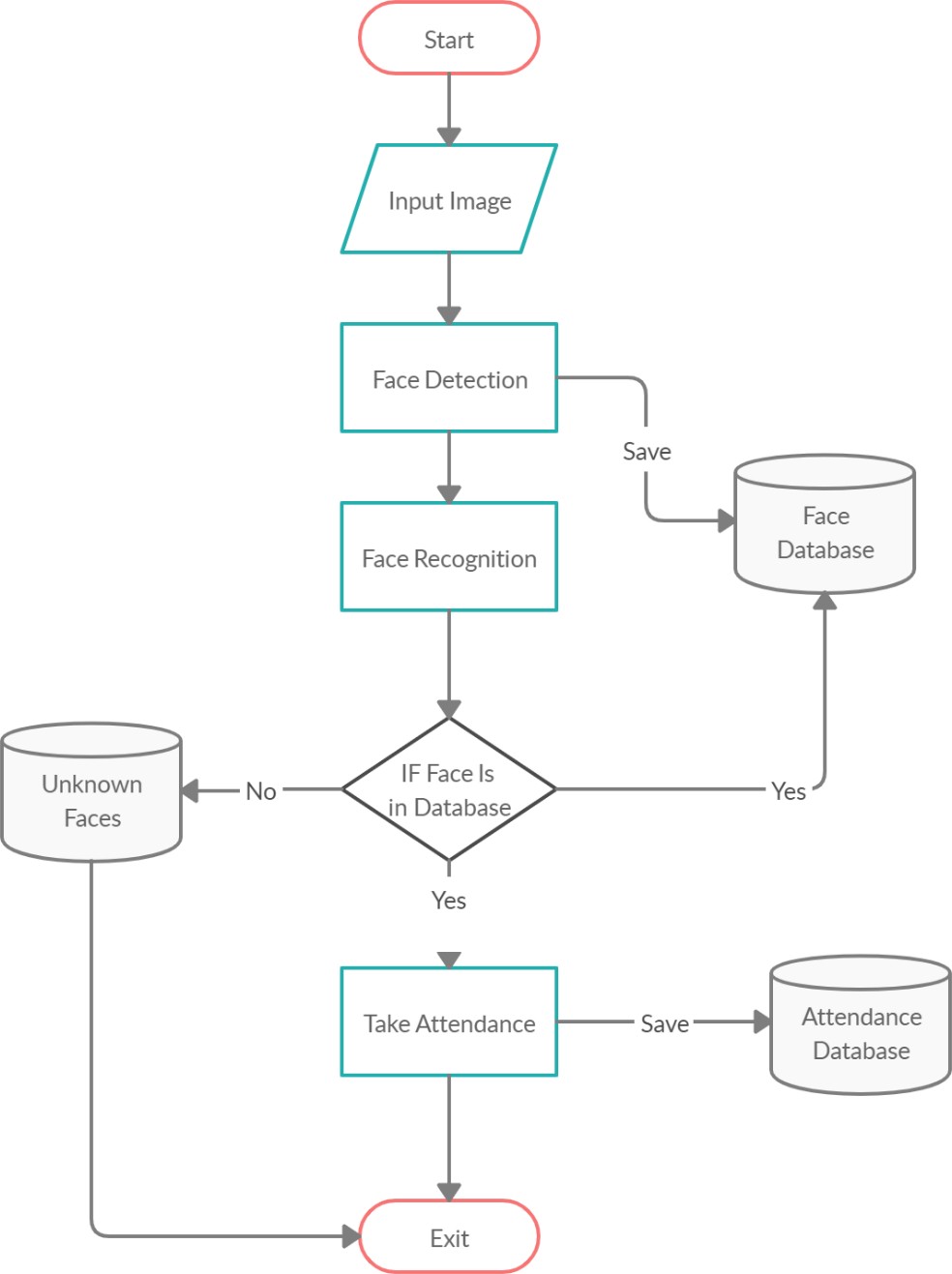
1. **A webcam**

A webcam is a small camera connected to a computer or device, capable of capturing video and images in real-time. It enables video conferencing, online streaming, and video recording for various purposes, such as communication, content creation, and virtual meetings. Webcams have become an essential tool in the digital age, facilitating remote interactions and collaborations.

## 3.5. SYSTEM FLOW CHART

For registering student

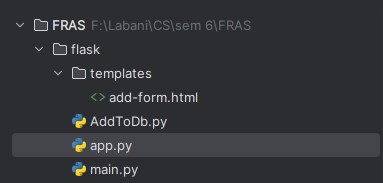
For taking attendance



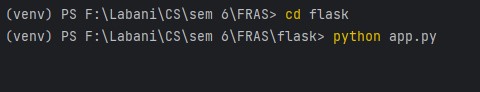
## 3.6. SCREENSHOTS OF WORKING SYSTEM

### 3.6.1. REGISTERING STUDENTS/ EMPLOYEES

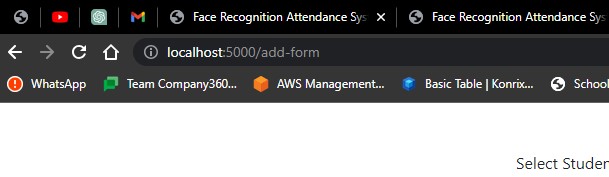
1. Go to the directory where app.py exists



1. Run command python app.py in terminal



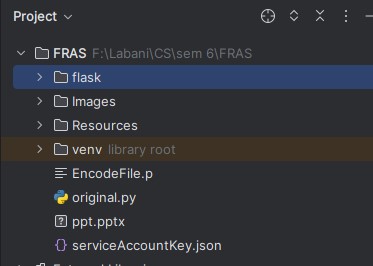
1. Go to http://localhost:5000/add-form in the browser



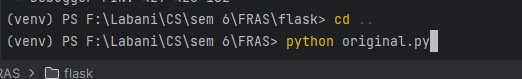
1. Fill the form and submit to register student for the attendance system

### 3.6.2. TAKING ATTENDANCE

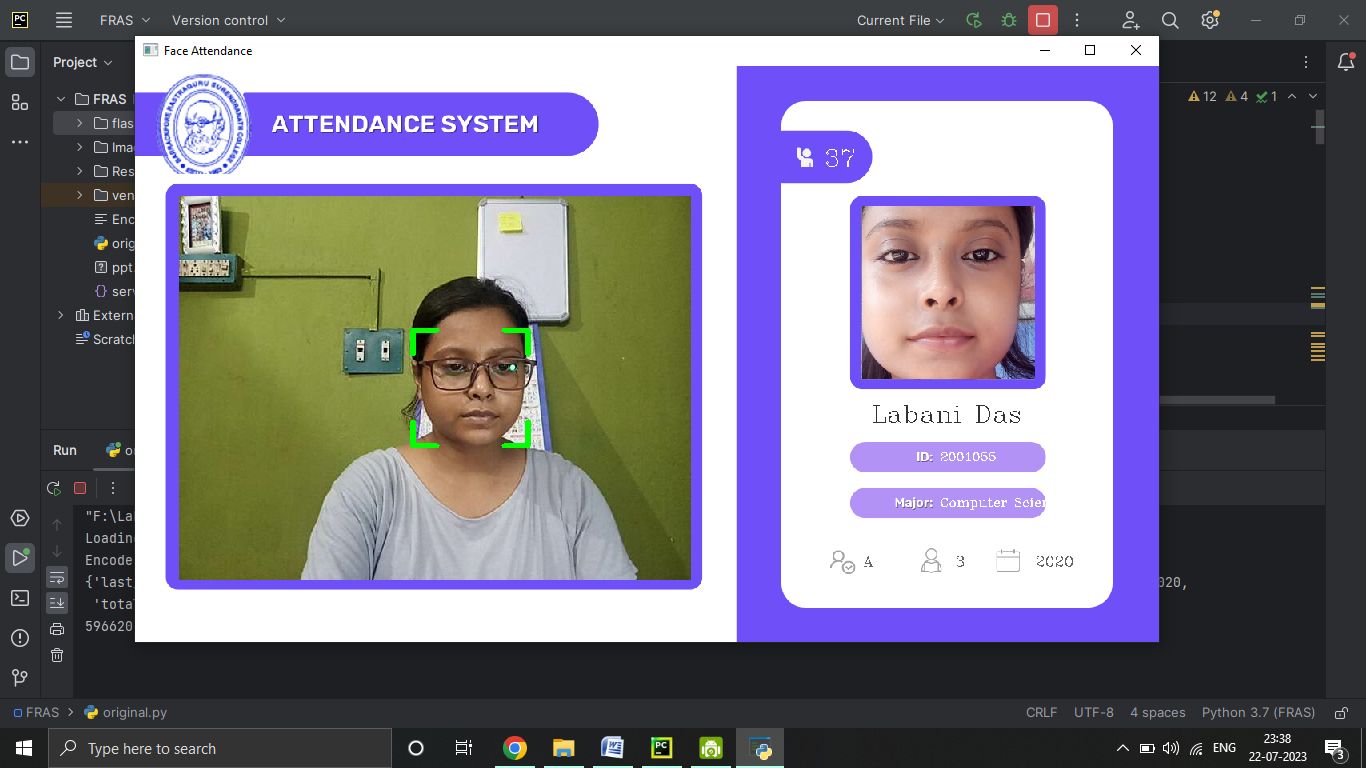
1. Go to FRAS folder

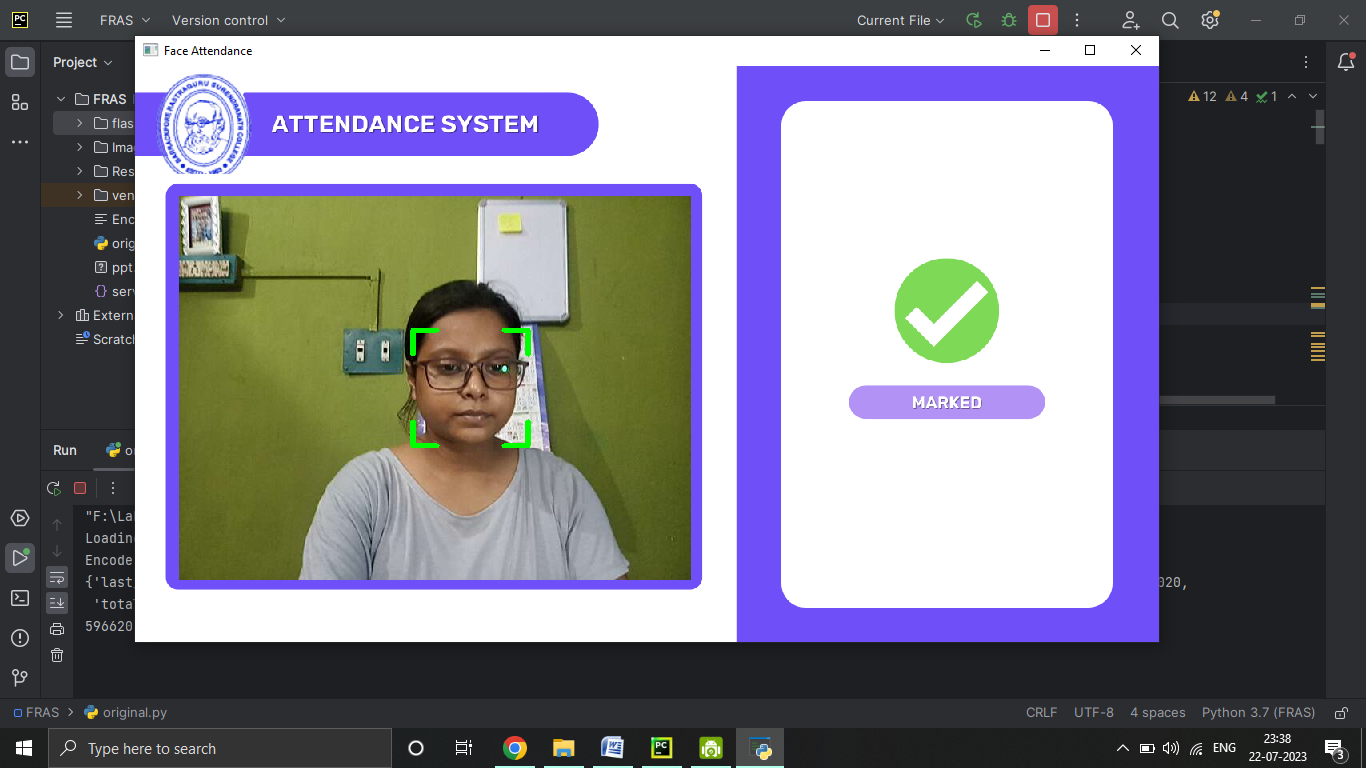


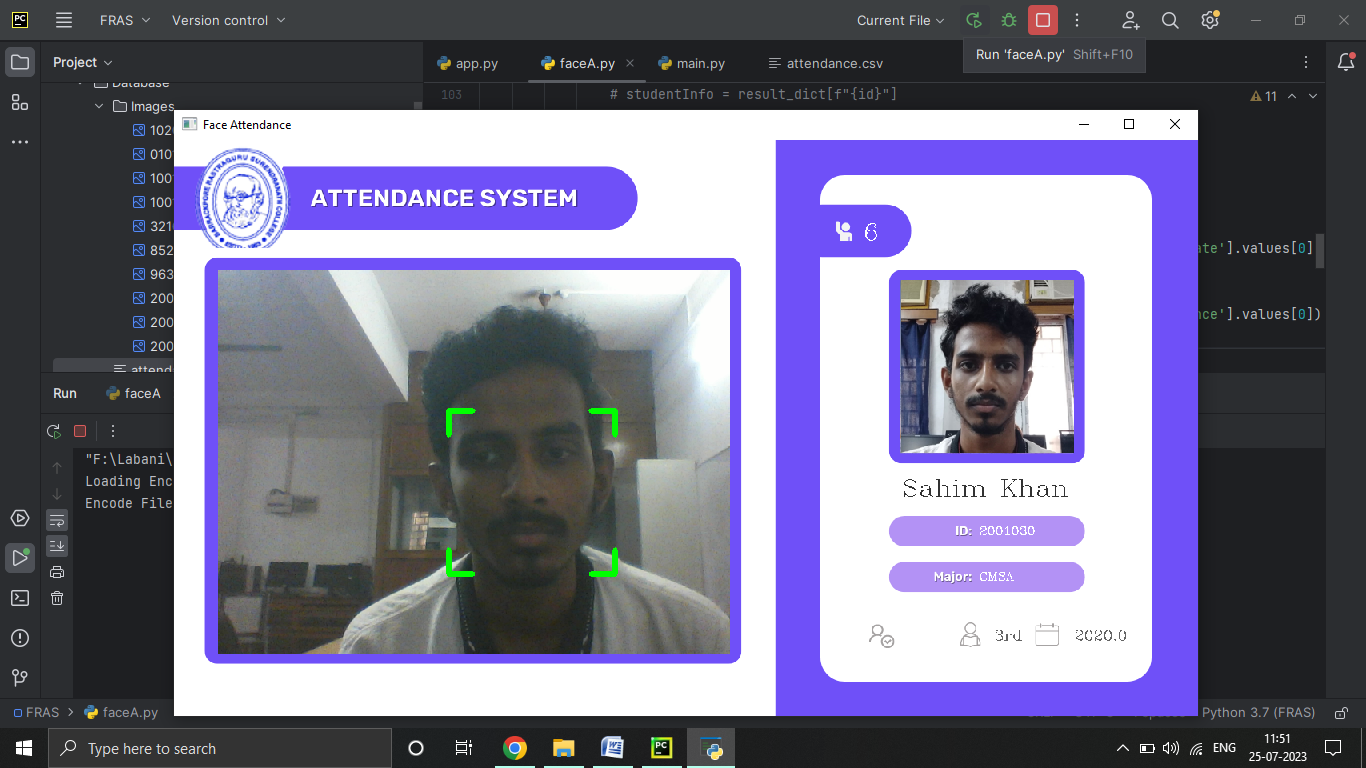
1. Run command python original.py in the terminal

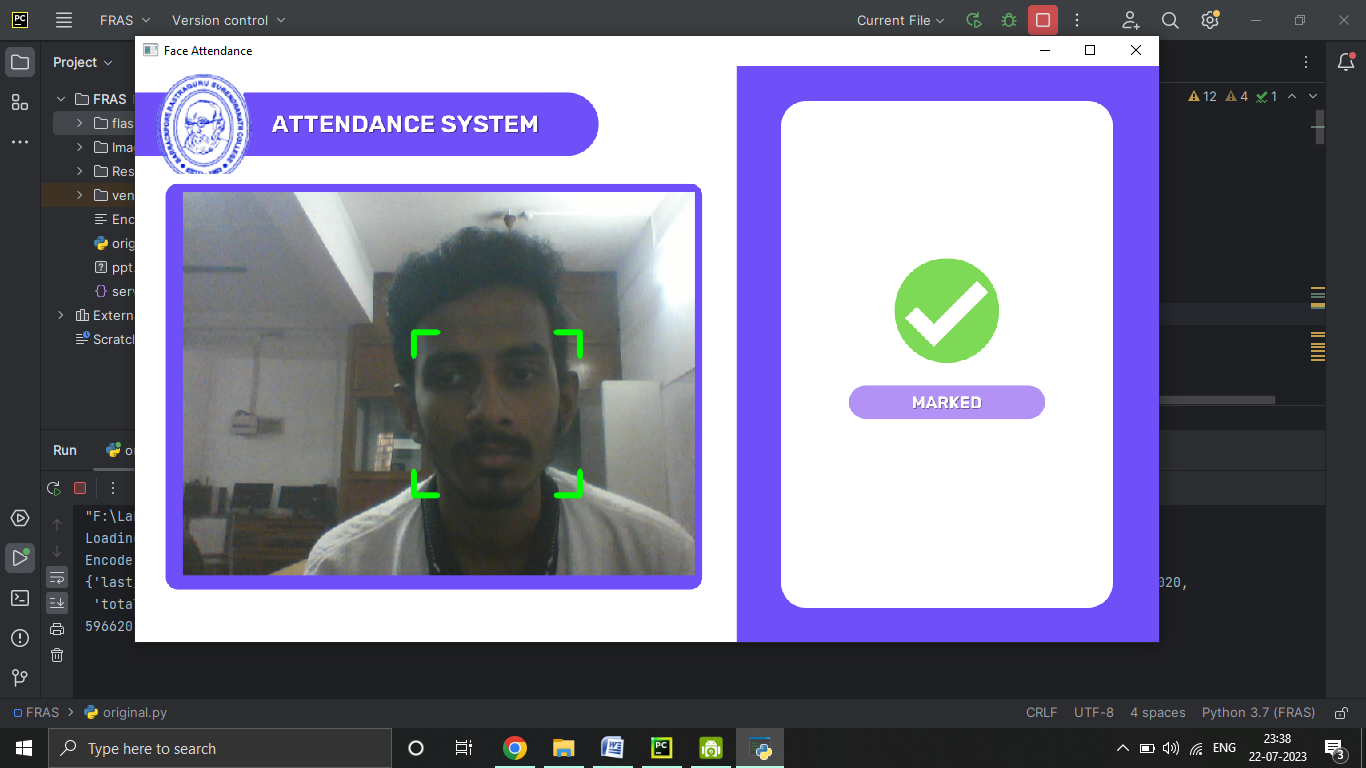


1. Stand in front of camera for 2-3 seconds to get your attendance marked

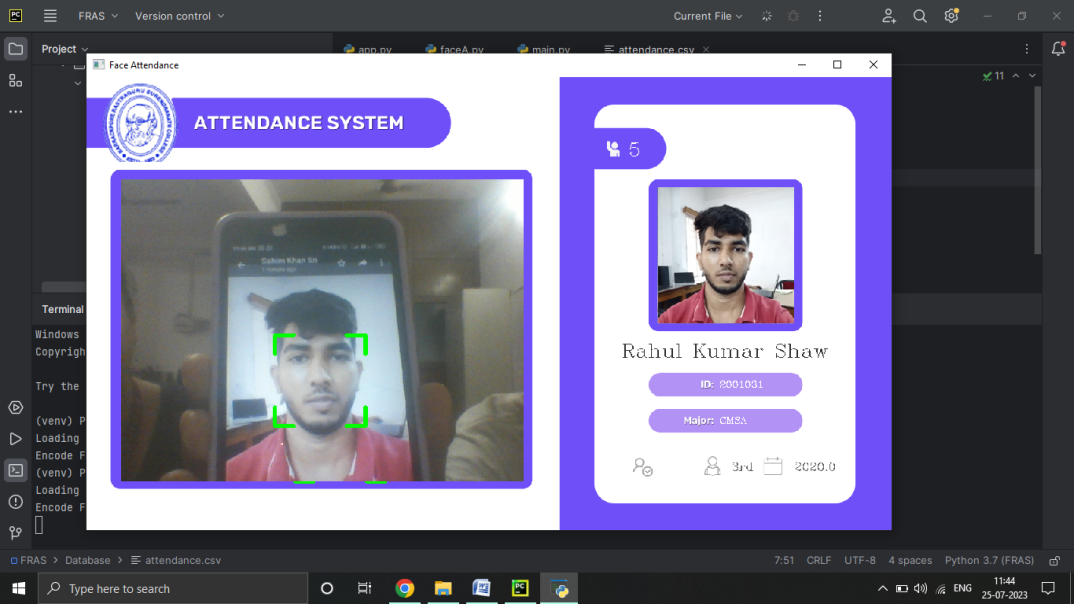


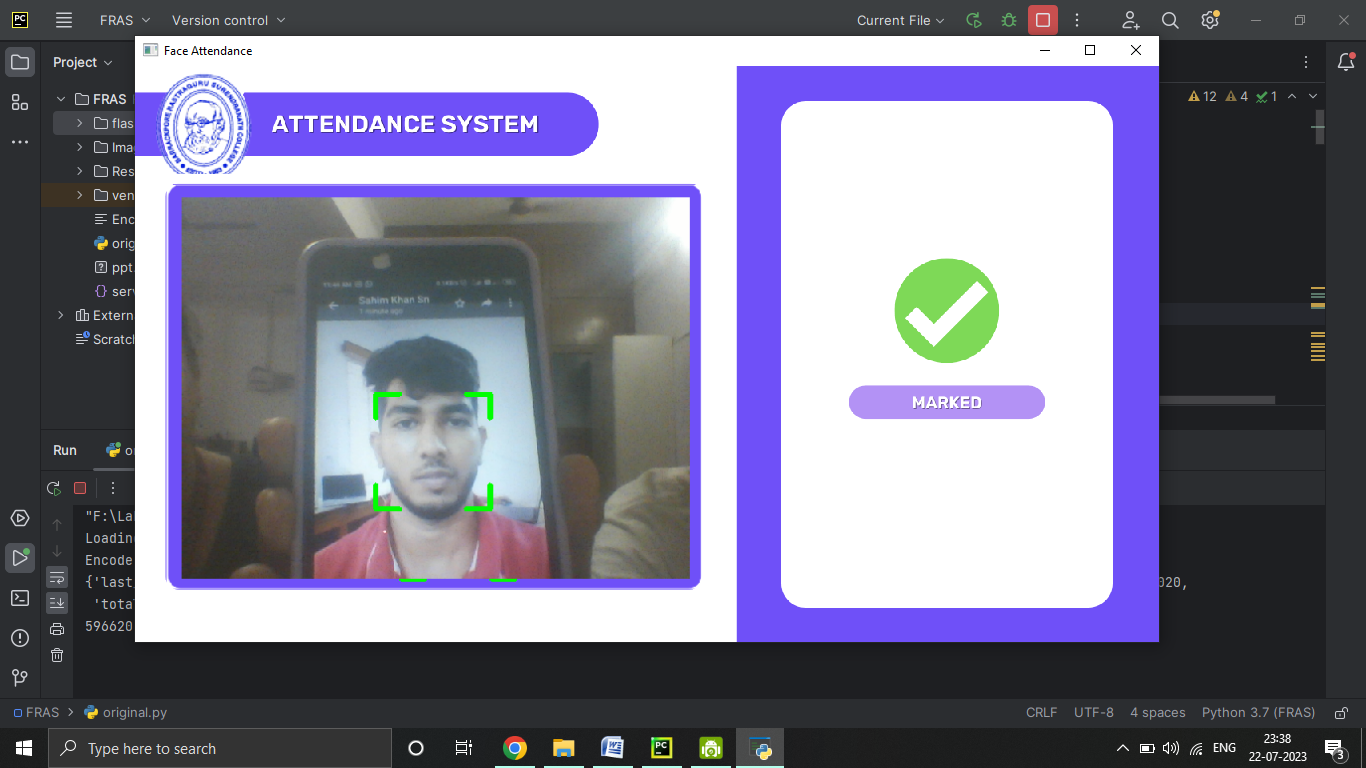






After waiting for 5-6 seconds the status of is shown as marked to the users





*See the Face Attendance System in action! Our innovative system uses advanced face recognition technology to accurately track and record attendance. Say goodbye to traditional methods and experience a seamless and efficient way to manage attendance effortlessly*

Chapter 4

# END USER STORY

## 4.1. DEFINITION

An End User Story in software development is a brief, user-focused description of a specific feature or functionality. It communicates the user's needs, goals, and the value they expect from the software. These stories help guide development teams in delivering user-centric solutions and driving project success.

## 4.2. USER STORY

**ADMIN USER STORY:**

As an ADMIN,

I want to register a new user,

So I can add their information and picture to the system easily.

Acceptance Criteria:

1. The register page allows me to fill in the required user details.
2. . I can upload a picture of the user during registration.
3. After filling the form, I can submit it to save the user's information.

**CANDIDATES USER STORY:**

As a CANDIDATE,

I want to have my attendance taken,

So I can participate in the event or session.

Acceptance Criteria:

1. I stand in front of the camera for 2 to 3 seconds to capture my image.
2. My candidate ID card is shown to verify my identity.
3. Once my identity is confirmed and status shows marked, I can proceed.

Chapter 5

# BENEFITS AND LIMITATIONS

BENEFITS AND LIMITATIONS

BENEFITS

* Automate attendance registering
* Recognize and register attendance in seconds
* Easy to use

LIMITATIONS

* The accuracy of the system is not 100%. It can only detect face from a limited distance
* Sometimes slow as it data is constantly processing and being accessed from database in every iteration
* Cannot be used offline

Chapter 6

# CONCLUSION AND FUTURE SCOPE

CONCLUSION

The face recognition system aims to streamline student attendance tracking. Users can register students by submitting a form, including an uploaded student image. Data is stored in Firebase Realtime Database, and images are saved in Firebase Storage.

During face recognition, the system captures faces through the camera, cross-references them with the database, and if recognized, displays relevant student details.

This seamless process ensures efficient attendance management by automating registration and verification. By leveraging Firebase's robust cloud infrastructure, the system offers scalability, real-time updates, and secure storage. The user-friendly interface simplifies interactions, making it a valuable tool for educational institutions seeking a reliable, modern attendance solution.

FUTURE SCORE

* Performance Optimization: Optimize the face recognition algorithms and techniques to make the system faster and more efficient
* Multiple Face Detection: Extend the system to handle the detection and recognition of multiple faces in an image or video stream.
* Security and Privacy: Enhance the security of our system by implementing measures to prevent unauthorized access or tampering with the data
* User Interface and Experience: Improve the user interface and experience of the system to make it more user-friendly and intuitive.

Chapter 7

# REFERENCES

REFERENCES

* Digital Image Processing - Third Edition , Rafael C. Gonzalez (University of Tennessee) and Richard E. Woods (NledData Interactive), Pearson Education Publication
* Computer Graphics, Donald Hearn and M. Pauline baker, Pearson Education Publication
* Machine Learning For Absolute Beginners, Oliver Theobald, Independently Published, 2017
* Modern Face Recognition with Deep Learning <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
* Face landmark detection <http://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf>
* <https://medium.com/codex/facial-landmark-detection-algorithms-5b2d2a12adaf>
* <http://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf>
* <https://en.wikipedia.org/wiki/Support_vector_machine>
* Image Recognition using Histogram of Oriented Gradients (HOG) Descriptor <https://debuggercafe.com/image-recognition-using-histogram-of-oriented-gradients-hog-descriptor/>
* Face detection HOG *-* <https://medium.com/mlcrunch/face-detection-using-dlib-hog-198414837945>

Chapter 8

# APPENDIX

## 8.1. CODE FOR FOR REGISTERING STUDENT

Html file for taking student/ employee details through html form

Filename: add-form.html

<!DOCTYPE *html*>

<html *lang*="en">

  <head>

    <meta *charset*="UTF-8" />

    <title>Add Student</title>

    <link

*href*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"

*rel*="stylesheet"

*integrity*="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM"

*crossorigin*="anonymous"

    />

  </head>

  <body *class*="d-flex justify-content-center mt-5">

    <form

*action*="http://localhost:5000/save-student"

*method*="post"

*enctype*="multipart/form-data"

    >

      <div *class*="mb-3">

        <label *class*="form-label">Select Student Image</label>

        <input *type*="file" *name*="image" *class*="form-control" />

        <div *class*="form-text">Must upload image of square shape</div>

      </div>

      <div *class*="mb-3">

        <label *class*="form-label">Student ID</label>

        <input *type*="text" *name*="key" *class*="form-control" />

      </div>

      <div *class*="mb-3">

        <label *class*="form-label">Student Name</label>

        <input *type*="text" *name*="name" *class*="form-control" />

      </div>

      <div *class*="mb-3">

        <label *class*="form-label">Student Starting year</label>

        <input *type*="text" *name*="starting\_year" *class*="form-control" />

      </div>

      <div *class*="mb-3">

        <label *class*="form-label">Student Current Year </label>

        <input *type*="number" *name*="year" *class*="form-control" />

      </div>

      <input *type*="submit" *value*="submit" *class*="btn btn-primary" />

    </form>

    <script

*src*="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"

*integrity*="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"

*crossorigin*="anonymous"

    ></script>

    <script

*src*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"

*integrity*="sha384-fbbOQedDUMZZ5KreZpsbe1LCZPVmfTnH7ois6mU1QK+m14rQ1l2bGBq41eYeM/fS"

*crossorigin*="anonymous"

    ></script>

  </body>

</html>

Python Flask file for rendering html form

Filename: app.py

from flask import Flask, redirect, url\_for, request, render\_template

from AddToDb import saveStudentDetails, saveStudentImages

from main import compatible\_image, resize\_image

import os

app = Flask(\_\_name\_\_)

current\_directory = os.path.dirname(os.path.abspath(\_\_file\_\_))

*# root path*

**@app.route**('/')

def **root**():

   return "Hello World"

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*# get student detail form*

**@app.route**('/add-form', methods = ['GET'])

def **save\_student**():

      return render\_template('add-form.html')

*# post student detail form*

**@app.route**('/save-student', methods = ['POST'])

def **save**():

    if 'image' not in request.files:

        return "No image part in the request", 400

    image = request.files['image']

    key = request.form['key']

    name = request.form['name']

    starting\_year = request.form['starting\_year']

    year = request.form['year']

**print**(image, key, starting\_year, year)

*# Save the uploaded image to a file*

    folder = "Database/Images"

    path = f"{folder}/{key}.png"

    image.save(path)

**print**("Image saved")

*# check image compatibility*

    if compatible\_image(path):

        resize\_image(path)

        saveStudentDetails(key, name, starting\_year, year)  *#save student details*

        saveStudentImages()  *# save image*

    else:

**print**("Not compatible")

    return redirect(url\_for('save\_student'))

if \_\_name\_\_ == '\_\_main\_\_':

   app.run(debug = True)

code file that adds form details to firebase database

filename: addToDb.py

import os

import pickle

import cv2

import face\_recognition

import pandas as pd

from datetime import datetime, timedelta

def **saveStudentDetails**(key, name, starting\_year, year):

    previous\_day = datetime.now() - timedelta(days=1)

    previous\_day\_str = previous\_day.strftime('%Y-%m-%d')

    data = {

        "id": [str(key)],

        "name": [name],

        "starting\_year": [str(starting\_year)],

        "year": [str(year)],

        "major": ["CMSA"],

        "total\_attendance": [str(0)],

        "last\_attendance\_date": [previous\_day\_str]

    }

    df = pd.DataFrame(data)

    df.to\_csv("Database/attendance.csv", mode='a', index=False, header=not os.path.exists("Database/attendance.csv"))

**print**("Data saved")

def **findEncodings**(imagesList):

    encodeList = []

    for img in imagesList:

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

        encodeList.append(encode)

    return encodeList

def **saveStudentImages**():

    folderPath = 'Database/Images'

    pathList = os.listdir(folderPath)

**print**(pathList)

    imgList = []

    studentIds = []

    for path in pathList:

        imgList.append(cv2.imread(os.path.join(folderPath, path)))

        studentIds.append(os.path.splitext(path)[0])

**print**("Encoding Started ...")

    encodeListKnown = findEncodings(imgList)

    encodeListKnownWithIds = [encodeListKnown, studentIds]

**print**("Encoding Complete")

    file = **open**("EncodeFile.p", 'wb')

    pickle.dump(encodeListKnownWithIds, file)

    file.close()

**print**("File Saved")

file that checks image compatibility and resizes student profile image

Filename: main.py

from PIL import Image

import face\_recognition

import cv2

def **compatible\_image**(path):

*# check if square in shape*

    image = Image.open(path)

*# check if single face detected in the image*

    image = face\_recognition.load\_image\_file(path)

    face\_locations = face\_recognition.face\_locations(image)

*# Check if only a single face is detected*

    if **len**(face\_locations) != 1:

**print**("No face detected")

        return False

    return True

def **resize\_image**(path):

    image\_path = path

    square\_crop\_around\_face(path)

    image = Image.open(image\_path)

*# Resize the image to 216x216 dimensions*

    resized\_image = image.resize((216, 216))

    resized\_image.save(path)

def **square\_crop\_around\_face**(path):

*# Load the image*

    image = cv2.imread(path)

*# Find face locations in the image*

    face\_locations = face\_recognition.face\_locations(image)

    if **len**(face\_locations) == 0:

**print**("No face found in the image.")

        return

*# Assume the first face is the one we want to crop around*

    top, right, bottom, left = face\_locations[0]

*# Calculate the width and height of the face*

    face\_width = right - left

    face\_height = bottom - top

*# Calculate the size of the square crop*

    crop\_size = **max**(face\_width, face\_height)

*# Calculate the center of the face*

    center\_x = (left + right) // 2

    center\_y = (top + bottom) // 2

*# Calculate the coordinates for the square crop*

    crop\_left = **max**(center\_x - crop\_size // 2, 0)

    crop\_top = **max**(center\_y - crop\_size // 2, 0)

    crop\_right = **min**(center\_x + crop\_size // 2, image.shape[1])

    crop\_bottom = **min**(center\_y + crop\_size // 2, image.shape[0])

*# Crop the image*

    cropped\_image = image[crop\_top:crop\_bottom, crop\_left:crop\_right]

*# Save the cropped image*

    cv2.imwrite(path, cropped\_image)

**print**("Square crop around face complete. Cropped image saved to:", path)

## 8.2. CODE FOR FACE ATTENDANCE SYSTEM CODE

the main file that takes the attendance of students or employees

Filename: original.py

import os

import pickle

import numpy as np

import cv2

import face\_recognition

import cvzone

import pandas

from datetime import datetime

*# Create a window*

cv2.namedWindow("Face Attendance", cv2.WINDOW\_NORMAL)

*# Get the screen resolution*

screen\_width, screen\_height = 1024, 576  *# Replace with your desired screen resolution*

*# Resize the window to fit the screen*

cv2.resizeWindow("Face Attendance", screen\_width, screen\_height)

*# imgBackground = cv2.imread('Resources/background.png')*

imgBackground = cv2.imread('Resources/img.png')

cap = cv2.VideoCapture(0)

cap.set(3, 640)

cap.set(4, 480)

*# Importing the mode images into a list*

folderModePath = 'Resources/Modes'

modePathList = os.listdir(folderModePath)

imgModeList = []

for path in modePathList:

    imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))

*# # Importing the student images into a list*

folderModePath = 'Database/Images'

modePathList = os.listdir(folderModePath)

imagesStudent = {}

for path in modePathList:

    img = cv2.imread(os.path.join(folderModePath, path))

    imagesStudent[path] = img

*# importing attendance sheet*

data = pandas.read\_csv("Database/attendance.csv")

**print**(data)

*# Load the encoding file*

**print**("Loading Encode File ...")

file = **open**('EncodeFile.p', 'rb')

encodeListKnownWithIds = pickle.load(file)

file.close()

encodeListKnown, studentIds = encodeListKnownWithIds

**print**(studentIds)

**print**("Encode File Loaded")

modeType = 0

counter = 0

**id** = -1

imgStudent = []

while True:

    success, img = cap.read()

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

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

    faceCurFrame = face\_recognition.face\_locations(imgS)

    encodeCurFrame = face\_recognition.face\_encodings(imgS, faceCurFrame)

    imgBackground[162:162 + 480, 55:55 + 640] = img

    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    if faceCurFrame:

        for encodeFace, faceLoc in **zip**(encodeCurFrame, faceCurFrame):

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

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

            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:

**print**("Known Face Detected")

                y1, x2, y2, x1 = faceLoc

                y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4

                bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1

                imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)

**id** = studentIds[matchIndex]

                if counter == 0:

                    cvzone.putTextRect(imgBackground, "Loading", (275, 400))

                    cv2.imshow("Face Attendance", imgBackground)

                    cv2.waitKey(1)

                    counter = 1

                    modeType = 1

        if counter != 0:

            if counter == 1:

                imgStudent = imagesStudent[f"{**id**}.png"]

                id\_str = int(**id**)

                last\_attendance\_date = data.loc[data['id'] == id\_str]['last\_attendance\_date'].values[0]

                today = datetime.now().strftime('%Y-%m-%d')

                if last\_attendance\_date.split()[0] != (today).split()[0]:

                    attendance\_count = int(data.loc[data['id'] == id\_str]['total\_attendance'].values[0]) + 1

                    attendance\_count\_str = str(attendance\_count)

*# Update the 'Total Attendance' and 'Last Attendance Date' columns for the row*

                    data.loc[data['id'] == id\_str, 'total\_attendance'] = attendance\_count\_str

                    data.loc[data['id'] == id\_str, 'last\_attendance\_date'] = datetime.now().strftime(

                        "%Y-%m-%d %H:%M:%S")

*# Save the changes back to the CSV file*

                    data.to\_csv("Database/attendance.csv", index=False)

                else:

                    modeType = 3

                    counter = 0

                    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

            if modeType != 3:

                if 10 < counter < 20:

                    modeType = 2

                imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

                if counter <= 10:

*# id = int(id)*

                    id\_str = int(**id**)

                    cv2.putText(imgBackground, str(data.loc[data['id'] == id\_str]['total\_attendance'].values[0]),

                                (861, 125),

                                cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 1)

                    cv2.putText(imgBackground, str(data.loc[data['id'] == id\_str]['major'].values[0]), (1006, 550),

                                cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (255, 255, 255), 1)

                    cv2.putText(imgBackground, str(**id**), (1006, 493),

                                cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (255, 255, 255), 1)

                    cv2.putText(imgBackground, str(data.loc[data['id'] == id\_str]['year'].values[0]), (1025, 625),

                                cv2.FONT\_HERSHEY\_COMPLEX, 0.6, (100, 100, 100), 1)

                    cv2.putText(imgBackground, str(data.loc[data['id'] == id\_str]['starting\_year'].values[0]),

                                (1125, 625),

                                cv2.FONT\_HERSHEY\_COMPLEX, 0.6, (100, 100, 100), 1)

                    (w, h), \_ = cv2.getTextSize(data.loc[data['id'] == id\_str]['name'].values[0],

                                                cv2.FONT\_HERSHEY\_COMPLEX, 1, 1)

                    offset = (414 - w) // 2

                    cv2.putText(imgBackground, str(data.loc[data['id'] == id\_str]['name'].values[0]),

                                (808 + offset, 445),

                                cv2.FONT\_HERSHEY\_COMPLEX, 1, (50, 50, 50), 1)

                    imgBackground[175:175 + 216, 909:909 + 216] = imgStudent

                counter += 1

                if counter >= 20:

                    counter = 0

                    modeType = 0

*# studentInfo = {}*

                    imgStudent = []

                    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    else:

        modeType = 0

        counter = 0

    cv2.imshow("Face Attendance", imgBackground)

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

        break