# FACE RECOGNITION ATTENDANCE SYSTEM

**A PROJECT REPORT**

**Submitted By**

**YASH RAJ SINGH**
**(University Roll No- 2100290140154)**
**SHARAD GUPTA**
**(University Roll No- 2100290140122)**

**Submitted in partial fulfillment of the**
**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**Under the Supervision of**
**Dr. Akash Rajak**
**Professor**
**KIET Group of Institutions**

**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS**
**KIET Group of Institutions,**
**Delhi-NCR, Ghaziabad Uttar Pradesh-201206**

**June, 2023**

## CERTIFICATE

Certified that **Sharad Gupta (2100290140122), Yash Raj Singh (2100290140154)** have carried out the project work having "**Face Recognition Attendance System**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**DATE:-**

**Sharad Gupta (2100290140122)**
**Yash Raj Singh (2100290140154)**

This is to certify that the above statement made by the candidate is correct to the best of myknowledge.

**DATE:-**

**Dr. Akash Rajak**
**Professor**
**Department of Computer Applications**
**KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner**

**Signature of External Examiner**

**Dr. Arun K Tripathi**
**Head , Department of Computer Applications**
**Delhi -NCR, Ghaziabad, Uttar Pradesh -201206**

# DECLARATION

We, the students of Department of Computer Applications, KIET Group Of Institutions, Ghaziabad declare that the work entitled "**FACE RECOGNITION ATTENDANCE SYSTEM**" has been successfully completed under the guidance of Prof. Akash Rajak, Department of Computer Applications, KIET Group Of Institutions, Ghaziabad. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree Masters Of Computer Applications in Computer Applications during the academic year 2022-2023.

Place: **Team Members**

Date: **Sharad Gupta**
**(2100290140122)**

**Yash Raj Singh**
**(2100290140154)**

# ABSTRACT

In colleges, universities, organizations, schools, and offices, taking attendance is one of the most important tasks that must be done on a daily basis. The majority of the time, it is done manually, such as by calling by name or by roll number. The main goal of this project is to create a Face Recognition-based attendance system that will turn this manual process into an automated one. This project meets the requirements for bringing modernization to the way attendance is handled, as well as the criteria for time management. This device is installed in the classroom, where and student's information, such as name, roll number, class, sec, and photographs, is trained. The images are extracted using Open CV. Before the start of the corresponding class, the student can approach the machine, which will begin taking pictures and comparing them to the qualified dataset. This system consists of four phases- registration module, database creation, face detection, face recognition, attendance updating, attendance sending. Database consist of the image of the students in class. Face detection and recognition is performed using HOG feature extraction and SVM (Support Vector Machine) classifier. Faces will be detected and recognized from video streaming of the classroom. Attendance will be mailed to the respective faculty at the end of the lectures.

**Keywords**—Face Recognition; Face Detection; SVM classifier; HOG feature extraction; attendance system.

# ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **KIET Group Of Institutions, Ghaziabad** for providing me a platform to pursue my studies and carry out my final year project

I would like to thank **Dr. Arun K Tripathi,** Professor and Head, Department of Computer Applications, KIET Group Of Institutions, Ghaziabad, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide
**Dr. Akash Rajak, Professor,** Department of Computer Applications, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

**Yash Raj Singh**

**(2100290140154)**

**Sharad Gupta**

**(2100290140122)**

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Relevance of the Project

Traditional method of attendance marking is a hectic job in many institutions. It is also an extra task to the employee who must mark attendance by calling the names of students manually which may take minimum 5 minutes of whole session. This is time taking. There are some possibilities of proxy in attendance. That’s why, many institutes using many other techniques for taking attendance such as Radio Frequency Identification (RFID), iris scanner, fingerprint recognition, and list goes on. Face recognition has placed a very important biometric feature, which can be easily acquirable. Face recognition-based systems are relatively insensible to lots of facial expression. Face recognition system consists on two categories: verification and face identification. Face verification is a 1:1 matching process, it compares face image against the template face images and whereas is a 1:N problem that compares a single face image. The purpose of this system is to build an attendance system which is based on face recognition techniques. Here face of an individual will be used for marking attendance. Nowadays, face recognition is getting more popularity and has been widely used. In this paper, we proposed a system which finds the face of students from live streaming video and attendance will be marked if the detected face is found in the database. This new system will take less time than compared to old methods.

## 1.2 Problem Statement

The goal of this project is to build, update and maintain an Attendance recording subsystem using facial contours as the biometric key.

## EXISTING RECOGNITION SYSTEMS:

**Fingerprint Based recognition system***:* In the Fingerprint based existing attendance system, a portable fingerprint device need to be configured with the students fingerprint earlier.

**Iris Based Recognition System***:* In the Iris based student attendance system, the student needs to stand in front of a camera, so that the camera will scan the Iris of the student.

**Face Based Recognition System***:* The facial recognition technology can be used in recording the attendance through a high-resolution digital camera that detects and recognizes the faces of the students.

## PROPOSED SYSTEM AND SOLUTION:

All students must have to register in this system. And provide basic information like year, semester, branch, name, mobile number and image of their face. And image will store in database folder. In the time of class, student will have to come and see in the camera

## 1.3 Objective

In this project we aim to build an Attendance marking system with the help of facial recognition owing the difficulty of the manual as well as other traditional means of attendance systems the face will be detected automatically and compare with image stored in the database if both face and image found attendance will mark for respective student and end of the session admin have to stop the attendance marking and the attendance will automatically send to respected faculties by mail.

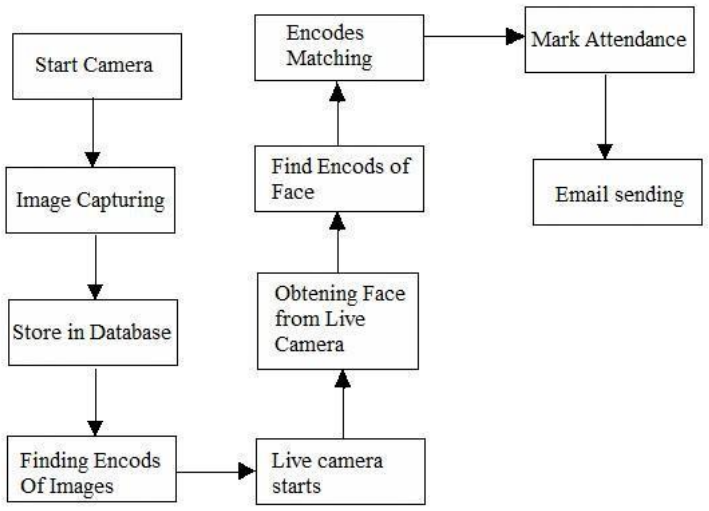


Fig 1.1 Steps in Attendance System

### 1.3.1 Database creation

Image of students is capture using our system with the help of web cam. Single image of student single student will acquire and this image will save in database without preprocessing which will be used for encoding findings. This image will be stored as the name of respective student and with roll no of student in database folder. And other information like name, roll no, phone number, branch, year, gender and semester will save in excel file name as “record.csv”

### 1.3.2 Face and Encodings Finding

The primary job is to detect the face, feature Extraction. Using landmarks of the face using Dlib and OpenCV. The state-of-the-art face recognition method build using deep learning technique 68 different points could be applied to any of the ranges from chin to eyes etc.as shown in fig 2. The transformations are rotation and scaling the preserve parallel lines known as affine transformation

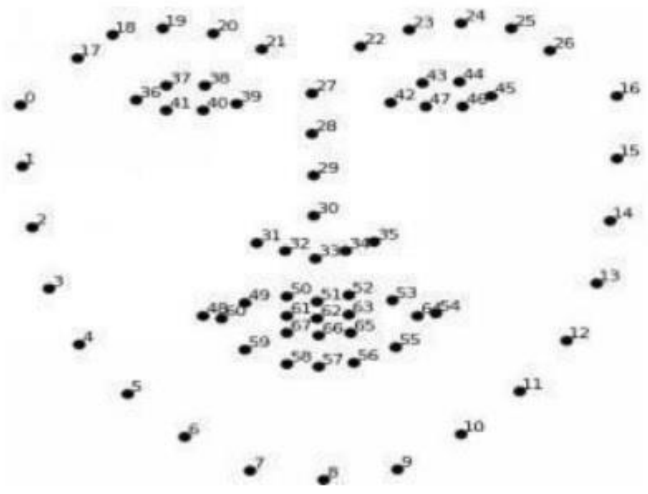


Fig 1.2. Depiction of 68 points of face

After finding face the next step is to find the encodings of that face using find encodings function present in face-recognition library of python and store it in text file by using pickle library of python, so that we can use it again and again to match the faces.

### 1.3.3 Face recognizing

This module is divided into two parts first is finding face in live streaming camera and find the encodings. Second is to matching the encodings of given face with the encodings present in text file created using pickle library. If the face is matched with database encodings the system goes to next module i.e. 4th module

### 1.3.4 Attendance marking

If given face matches with face from database then the students attendance will mark as present along with their name and roll-no. It will save in .csv file (excel file). After the lecture ends the .csv file will send to respective faculties

## 1.4 Scope of the project

- Using this system we will able to accomplish the task of marking the attendance in the classroom automatically and output is obtained in an excel sheet as desired in real-time.
- However, in order to develop a dedicated system which can be implemented in an educational institution, a very efficient algorithm which is insensitive to the lighting conditions of the classroom has to be developed.
- Also a camera of the optimum resolution has to be utilised in the system.
- Another important aspect where we can work towards is creating an online database of the attendance and automatic updating of the attendance

## 1.5 Methodology

In order to obtain the attendance, positions and face images in lecture, we proposed the attendance management system based on face detection in the classroom lecture. The system estimates the attendance and positions of each student by continuous observation and recording. Current work is based on the method to obtain the different weights of each focused seat.

The technology aims in imparting tremendous knowledge oriented technical innovations these days. Deep Learning is one among the interesting domain that enables the machine to train itself by providing some datasets as
input and provides an appropriate output during testing by applying different learning algorithms. 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. In general, the attendance system of the student can be,

**1. Manual Attendance System (MAS)**
**2. Automated Attendance System (AAS)**

### 1.5.1 Manual Attendance System (MAS)

Manual Student Attendance Management system is a process where a teacher concerned with the particular 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.

### 1.5.2 Automated Attendance System (AAS)

Automated Attendance System (AAS) is a process to student in the classroom by using face recognition technology. It is also possible to recognize whether the student is sleeping or awake during the lecture and it

can also be implemented in the exam sessions to ensure the presence of the student. Attendance of students will be taken by a real-time camera positioned at the door which senses anyone entering or exiting the classroom. The camera is trained in such a way that it differentiates shadows and photos.

The two common Human Face Recognition techniques are,

1. **Feature-Based approach**
2. **Brightness Based approach**

#### 1.5.2.1 Feature-Based approach

The Feature-based approach also known as local face recognition system, used in pointing the key features of the face.

#### 1.5.2.2 Brightness Based approach

Brightness-based approach also termed as the global face recognition system, used in recognizing all the parts of the image.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 Implementation of classroom attendance system based on face recognition in class

The system consists of a camera that captures the images of the classroom and sends it to the image enhancement module. To enhance the captured image histogram normalization, median filtering and skin classification methods are used. Face detection is done using Viola-Jones algorithm. Initially face detection algorithm was tested on variety of images with different face positions and lighting conditions and then algorithm was applied to detect faces in real time video. Algorithm is trained for the images of faces and then applied on the class room image for detection of multiple faces in the image. The next step is face recognition, where a hybrid algorithm from PCA and LDA is used. The detected faces are cropped from the image and compared with the face database using an Eigen face method. The face database consists of templates of face images of individual students that was collected and stored by an enrollment process. In this way the faces of students are verified one by one and the attendance is marked on the server. A time table module is attached to the system to obtain the subject, class, date and time. Teachers come in the class and just press a button to start the attendance process.

## 2.2 Face Recognition-based Lecture Attendance System

The system consists of two cameras; one for determining the seating positions (fixed at the ceiling) and the other for capturing the students face (Fixed in front of the seats).To determine the target seat Active Student Detection(ASD) method is used to estimate the existence of a student on a seat. One seat is targeted and camera is directed to capture the image. The face image capture is enhanced and recognized and are recoded into the database. Every seat has a vector of values that represent relationship between the student and seat. Attendance is estimated by interpreting the face recognition data obtained by continuous observation. The position and attendance of the student are recorded into the database.

## 2.3 Study of Implementing Automated Attendance System Using Face Recognition Technique

The proposed system has been implemented in three basic steps.The first step is face detection and extraction. The user stands in front of the camera and an image is captured, which is taken as input. The frontal face is captured by using the OpenCVHaarCascade method. After the face is detected, it is converted into a gray scale image of 50x50 pixels. The second step is to learn and train face images. The system needs to be initialized by feeding it a set of training images of faces. The PCA algorithm is performed on it. All the learned data is stored in an xml file. The third step is the recognition and identification. In this step the frontal face that is to be recognized, test face, is extracted from the image. The Eigen value for the test face is re-calculated and is matched with the stored data for the closest neighbor. Finding the closest neighbor is implemented as a function that computes distance from the projected test face to each projected training set. The distance basis here is "Squared Euclidean Distance." When a face is matched the corresponding information is obtained from the database. The log table is then updated with the system time to mark the attendance of that person.

## 2.4 Face Recognition Based Attendance Marking System

The system consists of a camera that must be positioned in the office room to take snap shots of the room. These images are then sent to an enhancement module where Histogram Normalization is used for the contrast enhancement of the image, Median Filter is used for removing noise from the image. To avoid false detection skin classification technique is used. This process first classifies the skin and then retains only the skin pixels and the other pixels are set to black. The enhanced image is then sent to a face detection and recognition module. This requires MATLAB software version 7.6. Two databases are maintained, the first one is the Face database to store the face images and extracted features at the time of enrolment process and the second attendance database contains the information about the employees and is also used to mark attendance.

## 2.5 Attendance Management System Using Face Recognition

In this system, the CCTV is fixed at the entry of the class room and is used to capture an image of the entering student. The detected faces are stored in a database and is compared with the existing images using Eigen faces methodology. To identify if the student image is matching, a 3D face recognition technique is used.If a match is found, that image is processed for attendance management. For attendance management, the attendance will be marked for the student image matched and the information is sent to the server which controls the overall database of the student. The software is installed in a smart phone that would help to improve the report features.When the server receives the message of student who are absent that particular day will send an SMS to the parent of that particular student.

# CHAPTER 3

# SYSTEM REQUIREMENTS SPECIFICATION

## 3.1 Hardware Requirements

- **Processor:** Intel Core i3 (6th Gen) i3-6006U Dual-core (2 Core) 2.0 GHz
- **RAM:** 4 GB (DDR3L-1600)
- **Hard Disk:** 500 GB HDD
- **Webcam**

## 3.2 Software Requirements

- **Operating System :** Windows 7 and Above
- **IDE :** PyCharm
- **Programming Language:** Python (3.8.5) 64-bit
- **Back-End Database:** Google Firebase

## 3.3 Module Description

- Python
- OpenCV
- Numpy
- CvZone
- Face Recognition
- Pickle
- OS
- Firebase

### 3.3.1 Python

Python is a popular, high-level programming language that was created by Guido van Rossum and released in 1991. It is designed to be easy to read and write, with a syntax that emphasizes code readability and simplicity. Python has gained immense popularity over the years due to its versatility, readability, and extensive standard library, which provides a wide range of functionalities for various domains.

One of the key features of Python is its simplicity. The language strives to have a clean and straightforward syntax, making it easier for both beginners and experienced programmers to understand and write code. Python emphasizes code readability through the use of indentation, rather than brackets or braces, to define blocks of code. This indentation-based syntax promotes clean and organized code and reduces the likelihood of errors caused by mismatched brackets or braces.

Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the most suitable approach for their projects. In the procedural paradigm, code is organized into functions and procedures, while in the object-oriented paradigm, code is structured around objects that encapsulate data and behavior. The functional programming paradigm focuses on writing code using pure functions without side effects.

The Python Standard Library is a vast collection of modules and packages that provide a wide range of functionalities. It includes modules for working with files, networking, databases, web development, scientific computing, and much more. This extensive library reduces the need for developers to write code from scratch, saving time and effort.

### 3.3.2 OpenCV

OpenCV (Open Source Computer Vision) is a powerful open-source library widely used for computer vision and image processing tasks. It provides a comprehensive set of functions and algorithms to handle various aspects of computer vision, including image and video manipulation, object detection and tracking, feature extraction, and more. With OpenCV, developers can build applications that analyze and interpret visual data.

OpenCV supports multiple programming languages, including Python, C++, and Java. However, Python is often the preferred choice due to its simplicity and ease of use. The Python bindings for OpenCV provide a seamless integration between the library and the language, allowing developers to leverage OpenCV's capabilities with concise and readable Python code.

Using OpenCV in Python, you can perform a wide range of tasks. Starting with basic image operations such as reading, displaying, and saving images, OpenCV provides functions for image filtering, transformation, and enhancement. You can apply various filters, such as blurring, sharpening, and edge detection, to manipulate the appearance of images.

OpenCV also includes algorithms for object detection and tracking, such as Haar cascades and deep learning-based methods. These techniques enable the identification and tracking of objects in images or videos. Additionally, OpenCV offers feature detection and extraction algorithms, such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features), which can be used for tasks like image recognition and matching.

### 3.3.3 Numpy

NumPy (Numerical Python) is a fundamental library for numerical computing in Python. It provides a powerful array object, along with a collection of functions for performing mathematical operations on arrays efficiently. NumPy is a cornerstone of the scientific Python ecosystem and is widely used for tasks such as data manipulation, numerical analysis, and scientific computing.

At the core of NumPy is the nd-array (n-dimensional array) object, which allows for efficient storage and manipulation of homogeneous multidimensional data. NumPy arrays provide a versatile and memory-efficient data structure that can handle large datasets with ease. The arrays support various data types, including integers, floats, and complex numbers, allowing for flexible and precise calculations.

NumPy provides a wide range of mathematical functions and operators that can be applied directly to arrays, eliminating the need for explicit loops and improving code performance. These functions include basic mathematical operations like addition, subtraction, multiplication, and division, as well as more advanced functions like trigonometric, exponential, and statistical operations.

One of the key advantages of NumPy is its ability to perform element-wise operations and broadcasting. Element-wise operations allow for efficient computation on entire arrays, reducing the need for explicit loops. Broadcasting enables arrays of different shapes to be combined and operated upon, making it easy to perform operations on arrays of different dimensions or sizes.

### 3.3.4 CvZone

CvZone is a Python library that provides a set of tools and utilities for computer vision and image processing tasks. It aims to simplify common computer vision operations and streamline the development process by offering pre-built functions and modules. CvZone is designed to be easy to use, making it accessible to both beginners and experienced developers.

One of the key features of CvZone is its extensive collection of ready-to-use functions for various computer vision tasks. These functions cover a wide range of operations, including image processing, object detection, facial recognition, pose estimation, and more. By leveraging these functions, developers can save time and effort in implementing common computer vision functionalities.

CvZone also includes modules that facilitate complex computer vision tasks. For example, the Pose-Module provides functionalities for pose estimation, allowing developers to detect and track human poses in images or videos. Similarly, the Face-Module offers tools for facial recognition and facial landmark detection, enabling applications such as face recognition and emotion analysis.

CvZone's documentation is well-documented, providing clear explanations, usage examples, and tutorials for each module and function. The library's GitHub repository also contains additional resources, including sample projects and demos, which can serve as starting points for developers to build their computer vision applications.

### 3.3.5 Face Recognition

The face recognition library in Python is a powerful tool that allows developers to detect and recognize faces in images and videos. It provides a range of functionalities for face detection, facial landmark detection, face recognition, and face attribute analysis. With this library, developers can build applications for various domains, including security systems, biometric identification, and augmented reality.

One of the key features of the face recognition library is its accurate face detection capability. It uses advanced algorithms to locate faces within images or video frames, even in complex and varying lighting conditions. Once faces are detected, the library can extract facial landmarks, such as the positions of the eyes, nose, and mouth, providing precise facial feature information.

Face recognition is another crucial functionality offered by the library. It can compare faces against a known database of faces to determine if a match exists, allowing for identification and verification tasks. The library employs machine learning algorithms, such as deep neural networks, to learn and represent face features, ensuring accurate and robust recognition performance.

The face recognition library also supports face attribute analysis, enabling developers to extract additional information from faces, such as age, gender, emotion, and facial expressions. This functionality opens up possibilities for applications in market research, emotion detection, and human-computer interaction.

### 3.3.6 Pickle

The pickle library in Python is a powerful module that allows developers to serialize and deserialize Python objects, enabling them to be stored persistently or transferred between different systems. Pickle provides a convenient way to convert complex data structures, such as lists, dictionaries, and class instances, into a binary representation that can be saved to a file or transmitted over a network.

One of the key features of the pickle library is its ability to preserve the complete state of an object, including its data and the relationships between different objects. This means that when an object is unpickled, it is reconstructed in its original form, including all its attributes and methods. This makes pickle useful for tasks such as saving and restoring program states or sharing objects between different Python processes.

The pickle library offers two main methods: `pickle.dump()` and `pickle.load()`. The `pickle.dump()` function serializes an object and writes it to a file, while `pickle.load()` reads the serialized data from a file and reconstructs the object. These methods provide a straightforward way to store and retrieve complex data structures without having to manually convert them to a different format.

Pickle is also extensible, allowing developers to define custom pickling and unpickling methods for their own classes. This flexibility enables fine-grained control over the serialization and deserialization process, allowing developers to handle special cases or complex object relationships.

### 3.3.7 OS

The OS library in Python provides a wide range of functions for interacting with the operating system. It offers a unified interface for accessing and manipulating various operating system functionalities, making it a powerful tool for file and directory operations, process management, environment variables, and more.

One of the key features of the OS library is its file and directory handling capabilities. It allows developers to perform operations such as creating, deleting, renaming, and moving files and directories. The library also provides functions for navigating directory structures, listing directory contents, and checking file properties, such as size, permissions, and timestamps.

The OS library facilitates environment management through functions for accessing and modifying environment variables. It enables developers to retrieve information such as the current working directory, the user's home directory, or system-specific variables. Developers can also set environment variables dynamically, providing flexibility in configuring application behavior based on the operating system environment.

Process management is another essential functionality offered by the OS library. It enables developers to launch external processes, interact with them, and obtain information about running processes. The library provides functions for spawning new processes, terminating processes, and retrieving information such as process IDs and exit statuses.

### 3.3.8 Firebase

Firebase is a comprehensive mobile and web application development platform offered by Google. It provides developers with a suite of tools and services to build, deploy, and scale applications quickly and efficiently. Firebase encompasses various features, including real-time database, authentication, hosting, cloud storage, messaging, and more.

One of the key components of Firebase is the real-time database, which allows developers to store and sync data in real-time across multiple clients. It offers a NoSQL database that can be accessed and updated from both the client and server sides. The real-time nature of the database enables seamless data synchronization and enables building collaborative applications and real-time chat functionality.

Firebase provides robust authentication services, allowing developers to easily implement user authentication and authorization in their applications. It supports authentication using email/password, social media accounts, or custom authentication methods. With Firebase authentication, developers can manage user accounts, secure application data, and control access to resources.

Firebase also offers cloud storage, enabling developers to store and serve user-generated content such as images, videos, and documents. The cloud storage feature provides a scalable and reliable solution for storing and retrieving files in the cloud, making it simple to integrate file storage capabilities into applications.

Another notable feature of Firebase is its hosting service, which allows developers to deploy their web applications effortlessly. With Firebase hosting, developers can quickly publish and update their web content, benefit from global content delivery networks (CDNs), and enjoy automatic SSL certificate provisioning.

## 3.4 Functional Requirements

### 1. Face Detection

The system should be able to accurately detect and locate human faces in images or video frames. It should handle variations in lighting conditions, facial expressions, and angles to ensure robust face detection**.**

### 2. Facial Feature Extraction

Once a face is detected, the system needs to extract facial features such as eyes, nose, and mouth landmarks. This step is crucial for identifying unique facial characteristics and creating a face representation for recognition.

### 3. Face Recognition

The system should have a face recognition algorithm capable of comparing the extracted facial features with a database of known faces. It should accurately match and identify individuals based on their facial characteristics**.**

### 4. Database Management

The attendance system requires a well-organized database to store registered individuals' face templates and associated information such as names and employee/student IDs.

5. **Enrollment and Registration**

   The system should provide a user-friendly interface to enroll and register individuals into the database. This process typically involves capturing their images, extracting facial features, and storing the corresponding templates**.**

6. **Attendance Logging**

   The system should maintain a log of attendance records, including timestamps and individuals' identification, to track attendance data accurately.

7. **User Interface**

   The attendance system should have an intuitive and user-friendly interface for administrators or users to interact with. This interface allows managing the database, enrolling new individuals, and retrieving attendance records.

8. **Security and Privacy**

   Robust security measures should be implemented to protect the attendance system and its data. This includes secure storage of face templates, encryption of sensitive information, and access controls to prevent unauthorized access.

9. **Scalability and Performance**

   The system should be capable of handling a large number of users and real-time attendance tracking. It should perform efficiently even with high volumes of data and provide quick responses for identification and logging**.**

10. **Integration**

    The attendance system may need to integrate with other systems such as access control systems, time and attendance management software, or human resource databases. Integration capabilities allow seamless data exchange and synchronization between different systems**.**

# CHAPTER 4

# SYSTEM ANALYSIS AND DESIGN

## 4.1 HOG Algorithm (Histogram of Oriented Gradient)

The Histogram of Oriented Gradients (HOG) algorithm is a popular technique in computer vision and image processing for object detection and recognition. It provides a robust representation of the shape and appearance of objects in images by analyzing the distribution of local gradient orientations.

The HOG algorithm operates by dividing an input image into small, overlapping cells. For each cell, it calculates the local gradient magnitude and orientation using techniques such as gradient operators or filters like Sobel or Scharr. The gradient magnitude represents the intensity change, while the gradient orientation indicates the direction of the intensity change.

Once the gradient magnitudes and orientations are computed, the next step is to create a histogram of gradient orientations for each cell. The histogram summarizes the distribution of gradient orientations within the cell. To enhance robustness against illumination variations and noise, the histograms often use weighted contributions from neighboring pixels using techniques like spatial or orientation normalization.

To capture more spatial information, the cells are often grouped into larger blocks. Within each block, the histograms are concatenated, creating a feature vector that represents the block. This concatenation helps capture patterns and spatial relationships within the block.

To handle object deformations and variations in scale, the HOG algorithm employs a sliding window approach. A window of fixed size is moved across the image, and at each position, the HOG features are extracted. These features are then fed into a classifier, such as a Support Vector Machine (SVM) or a neural network, to determine the presence or absence of the object of interest.

The training phase of the HOG algorithm involves collecting positive and negative samples of the object to be detected. Positive samples contain the object of interest, while negative samples represent the background or other objects. The HOG features are extracted from these samples, and a classifier is trained to learn the discriminative characteristics of the object.

The HOG algorithm has proven to be effective in various computer vision tasks, particularly in object detection and pedestrian detection. Its key strengths lie in its ability to capture object shape and appearance despite variations in lighting, scale, and occlusion. The algorithm's reliance on gradient orientations enables it to focus on the edges and contours of objects, which are important cues for object recognition.

However, the HOG algorithm also has some limitations. It is relatively sensitive to changes in viewpoint, as it does not explicitly model the 3D structure of objects. It may struggle to handle complex or cluttered backgrounds and may produce false positives or negatives in such scenarios. Furthermore, the HOG algorithm may not perform optimally for small or highly textured objects, as it relies on local gradient information.

In recent years, the HOG algorithm has been enhanced and combined with other techniques to improve object detection accuracy. Extensions such as Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), or deep learning-based approaches like Convolutional Neural Networks (CNNs) have been integrated to address some of the limitations of the original HOG algorithm.

## Steps to calculate HOG Features

1. Take the input image you want to calculate HOG features of. Resize the image into an image of 128x64 pixels (128 pixels height and 64 width). This dimension was used in the paper and was suggested by the authors as their primary aim with this type of detection was to obtain better results on the task of pedestrian

detection. As the authors of this paper were obtaining exceptionally perfect results on the MIT pedestrian database, they decided to produce a new and significantly more challenging dataset called the 'INRIA' dataset .

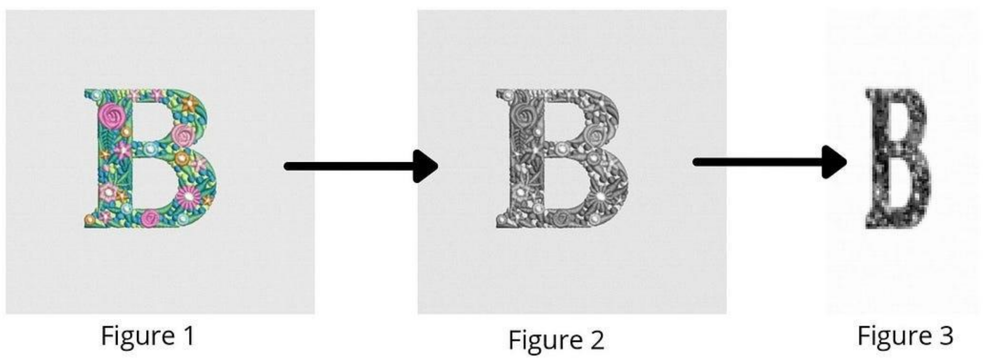


Figure 1 : The image imported to get HOG features of.
Figure 2 : The imported image grayscale for the process.
Figure 3 : Resized and grayscale image of the imported image.
fig. 4.1 Extraction of feature

2. The gradient of the image is calculated. The gradient is obtained by combining magnitude and angle from the image. Considering a block of 3x3 pixels, first Gx and Gy is calculated for each pixel. First Gx and Gy is calculated using the formulae below for each pixel value.

$$G_x(r,c) = I(r,c+1) - I(r,c-1) \quad G_y(r,c) = I(r-1,c) - I(r+1,c)$$

where r, c, refer to rows and columns respectively.

After calculating Gx and , magnitude and angle of each pixel is calculated using the formulae mentioned below.

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \quad Angle(\theta) = |\tan^{-1}(G_y/G_x)$$

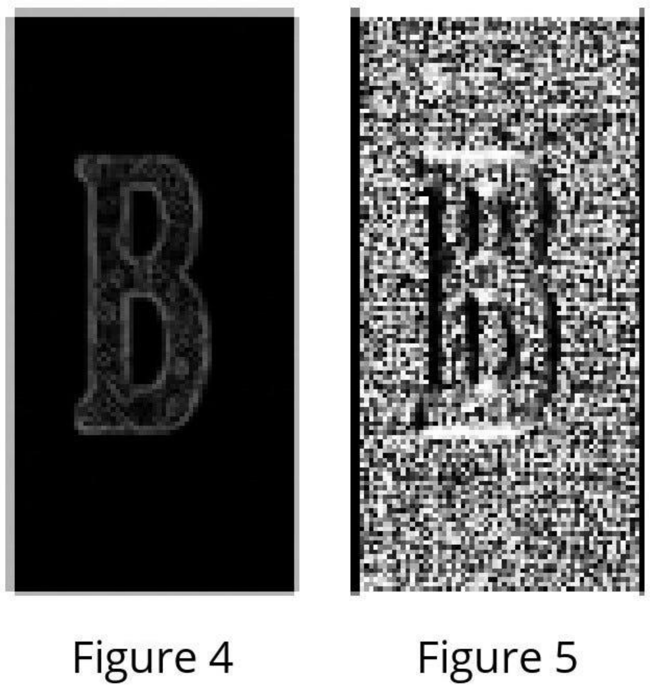


Figure 4 : Visualization of magnitude of the image.
Figure 5 : Visualization of angle of the image.
fig 4.2 Visualization of magnitue image

**3.** After obtaining the gradient of each pixel, the gradient matrices (magnitude and angle matrix) are divided into 8x8 cells to form a block. For each block, a 9-point histogram is calculated. A 9-point histogram develops a histogram with 9 bins and each bin has an angle range of 20 degrees. Figure 8 represents a 9 bin histogram in which the values are allocated after calculations. Each of these 9-point histograms can be plotted as histograms with bins outputting the intensity of the gradient in that bin. As a block contains 64 different values, for all 64 values of magnitude and gradient the following calculation is performed. As we are using 9 point histograms, hence:

$$Number\ of\ bins = 9(ranging\ from\ 0^\circ\ to\ 180^\circ)$$
$$Step\ size(\Delta\theta) = 180^\circ\ /\ Number\ of\ bins = 20^\circ$$

Each Jth bin, bin will have boundaries from :

$$[\Delta\theta \cdot j, \Delta\theta \cdot (j+1))$$

Value of the centre of each bin will be :

$$C_j = \Delta\theta(j + 0.5)$$

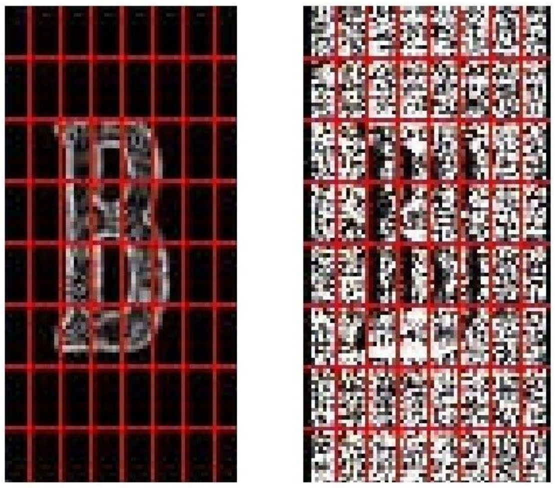


Figure 6 Figure 7

Figure 6 : 8x8 blocks on the magnitude image.
Figure 7 : 8x8 blocks on an angle image.
fig 4.3 8x8 blocks on magnitude image

**4.** For each cell in a block, we will first calculate the jth bin and then the value that will be provided to the jth and (j+1)th bin respectively. The value is given by the following formulae:

$$j = \lfloor \left( \frac{\theta}{\Delta\theta} - \frac{1}{2} \right) \rfloor$$

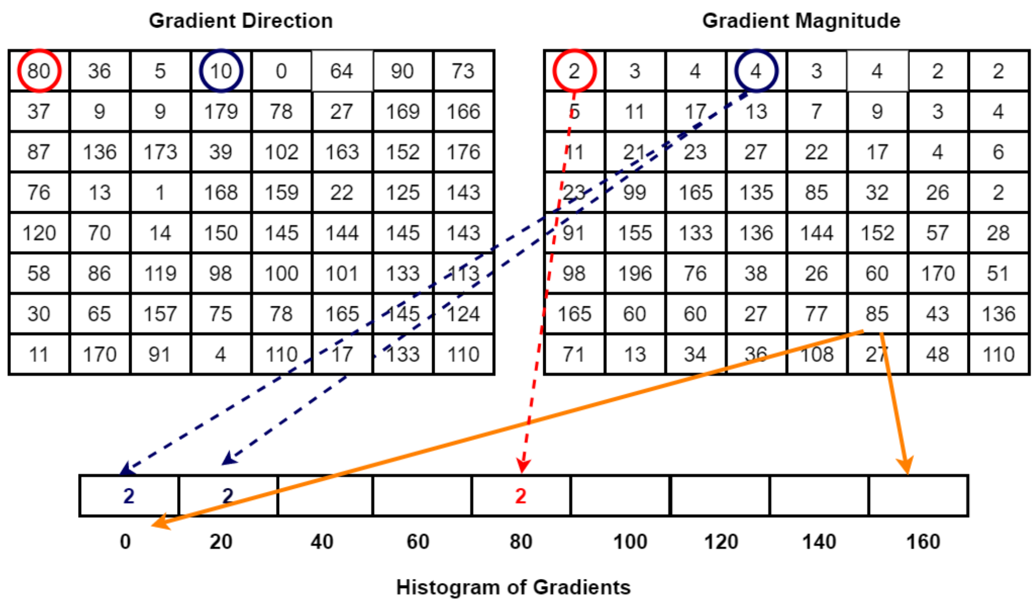
$$V_j = \mu \cdot \left[ \frac{\theta}{\Delta\theta} - \frac{1}{2} \right]$$

$$V_{j+1} = \mu \cdot \left[ \frac{\theta - C_j}{\Delta\theta} \right]$$

5. An array is taken as a bin for a block and values of Vj and Vj+1 is appended in the array at the index of jth and (j+1)th bin calculated for each pixel.

6. The resultant matrix after the above calculations will have the shape of 16x8x9.

7. Once histogram computation is over for all blocks, 4 blocks from the 9 point histogram matrix are clubbed together to form a new block (2x2). This clubbing is done in an overlapping manner with a stride of 8 pixels. For all 4 cells in a block, we concatenate all the 9 point histograms for each constituent cell to form a 36 feature vector.



Gradient Direction

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 80 | 36 | 5 | 10 | 0 | 64 | 90 | 73 |
| 37 | 9 | 9 | 179 | 78 | 27 | 169 | 166 |
| 87 | 136 | 173 | 39 | 102 | 163 | 152 | 176 |
| 76 | 13 | 1 | 168 | 159 | 22 | 125 | 143 |
| 120 | 70 | 14 | 150 | 145 | 144 | 145 | 143 |
| 58 | 86 | 119 | 98 | 100 | 101 | 133 | 113 |
| 30 | 65 | 157 | 75 | 78 | 165 | 145 | 124 |
| 11 | 170 | 91 | 4 | 110 | 17 | 133 | 110 |

Gradient Magnitude

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 4 | 3 | 4 | 2 | 2 |
| 5 | 11 | 17 | 13 | 7 | 9 | 3 | 4 |
| 11 | 21 | 23 | 27 | 22 | 17 | 4 | 6 |
| 23 | 99 | 165 | 135 | 85 | 32 | 26 | 2 |
| 91 | 155 | 133 | 136 | 144 | 152 | 57 | 28 |
| 98 | 196 | 76 | 38 | 26 | 60 | 170 | 51 |
| 165 | 60 | 60 | 27 | 77 | 85 | 43 | 136 |
| 71 | 13 | 34 | 36 | 108 | 27 | 48 | 110 |

fig 4.4

**8.** Values of fb for each block is normalized by the L2 norm :

$$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{||f_{bi}||^2 + \varepsilon}}$$

Where ε is a small value added to the square of fb in order to avoid zero division error. In code value taken of is 1e-05.

**9.** To normalize, the value of k is first calculated by the following formulae :

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + \ldots\ldots + b_{36}^2}$$
$$f_{bi} = \left[\left(\frac{b_1}{k}\right), \left(\frac{b_2}{k}\right), \left(\frac{b_3}{k}\right), \ldots\ldots, \left(\frac{b_{36}}{k}\right)\right]$$

**10.** This normalization is done to reduce the effect of changes in contrast between images of the same object. From each block. A 36 point feature vector is collected. In the horizontal direction there are 7 blocks and in the vertical direction there are 15 blocks. So the total length of HOG features will be : 7 x 15 x 36 = 3780. HOG features of the selected image are obtained.

fig 4.5

Visualization of HOG features on the same image using skimage library.

## 4.2 SVM Classifier

Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. SVMs are particularly effective in solving complex, high-dimensional problems and have gained popularity in various domains, including computer vision, text classification, and bioinformatics.

The underlying principle of SVMs is to find an optimal hyperplane that separates different classes of data with the largest margin. The hyperplane is a decision boundary that maximizes the distance between the closest data points of different classes, known as support vectors. This margin maximization helps SVMs achieve robust generalization performance.

In binary classification, an SVM algorithm aims to find a hyperplane that divides data points into two classes. However, SVMs can also handle multi-class classification problems by using techniques such as one-vs-one or one-vs-rest.

To find the optimal hyperplane, SVMs employ a kernel trick, which allows the transform the original input space into a higher-dimensional feature space. This transformation can make the data linearly separable in the new feature space, even if it was not separable in the original space. Commonly used kernels include the linear kernel, polynomial kernel, radial basis function (RBF) kernel, and sigmoid kernel.

Training an SVM involves solving an optimization problem, where the objective is to minimize the classification error while maximizing the margin. This optimization problem can be formulated as a convex quadratic programming task, which can be efficiently solved using algorithms such as Sequential Minimal Optimization (SMO) or the widely used LIBSVM library.

One of the key advantages of SVMs is their ability to handle high-dimensional feature spaces. This characteristic makes them suitable for tasks with a large number of features, such as image classification, text analysis, and gene expression analysis. SVMs can effectively capture complex relationships and patterns in high-dimensional data.

SVMs are also known for their robustness against overfitting. By maximizing the margin, SVMs aim to find the best generalizable decision boundary, which helps prevent overfitting. Regularization parameters, such as the C parameter in SVM, can be tuned to control the trade-off between margin maximization and training error minimization.

SVMs provide a dual representation, where the classification is based on the inner products of the support vectors. This property allows SVMs to efficiently perform classification even in high-dimensional spaces without explicitly computing the transformed feature vectors. It also enables the use of the kernel trick to handle nonlinear decision boundaries.

Interpreting SVMs is relatively straightforward. The support vectors, which lie on the margin or within the margin boundaries, play a crucial role in decision-making. SVMs classify new data points based on their distance from the decision boundary and the assigned class labels of the support vectors. This interpretation provides insights into the importance of specific data points in the classification process.

## Types of SVMs

There are two different types of SVMs, each used for different things:

### Simple SVM

Typically used for linear regression and classification problems.

### Kernel SVM

Has more flexibility for non-linear data because you can add more features to fit a hyperplane instead of a two-dimensional space. Those SVM which uses kernel functions to separate the data points is called Kernel SVM.

## Kernal Functions

### Linear

These are commonly recommended for text classification because most of these types of classification problems are linearly separable.

The linear kernel works really well when there are a lot of features, and text classification problems have a lot of features. Linear kernel functions are faster than most of the others and you have fewer parameters to optimize.

Here's the function that defines the linear kernel:

f(X) = w^T * X + b

### Polynomial

The polynomial kernel isn't used in practice very often because it isn't as computationally efficient as other kernels and its predictions aren't as accurate.

Here's the function for a polynomial kernel:

f(X1, X2) = (a + X1^T * X2) ^ b

This is one of the more simple polynomial kernel equations you can use. **f(X1, X2)** represents the polynomial decision boundary that will separate your data. **X1** and **X2** represent your data.

### Gaussian Radial Basis Function (RBF)

One of the most powerful and commonly used kernels in SVMs. Usually the choice for non-linear data. Here's the equation for an RBF kernel:

f(X1, X2) = exp(-gamma * ||X1 - X2||^2)

In this equation, **gamma** specifies how much a single training point has on the other data points around it. **||X1 - X2||** is the dot product between your features.

## Sigmoid

More useful in neural networks than in support vector machines, but there are occasional specific use cases.

Here's the function for a sigmoid kernel:

f(X, y) = tanh(alpha * X^T * y + C)

In this function, **alpha** is a weight vector and **C** is an offset value to account for some mis-classification of data that can happen.

Here are the steps regularly found in machine learning projects:

- Import the dataset
- Explore the data to figure out what they look like
- Pre-process the data
- Split the data into attributes and labels
- Divide the data into training and testing sets
- Train the SVM algorithm
- Make some predictions
- Evaluate the results of the algorithm

## 4.3 Dlib

Dlib is a versatile open-source C++ library that provides a comprehensive set of tools and algorithms for various computer vision and machine learning tasks. Developed by Davis King, Dlib is widely recognized for its efficiency, ease of use, and state-of-the-art implementations. While primarily written in C++, it also offers Python bindings, making it accessible to developers in both languages.

One of the main strengths of Dlib lies in its rich collection of algorithms for facial detection, facial landmark detection, and face recognition. The facial detection algorithm uses a combination of Haar cascades, HOG features, and a linear classifier to accurately locate faces in images or video streams. The facial landmark detection algorithm, known as the shape predictor, precisely identifies key points on the face, such as the eyes, nose, and mouth. These landmarks serve as essential cues for various applications like face alignment, emotion analysis, and facial expression recognition. Additionally, Dlib includes a highly accurate face recognition model based on deep learning techniques, allowing for robust face identification and verification.

Another notable feature of Dlib is its object detection capabilities. It provides an Implementation of the Histogram of Oriented Gradients (HOG) algorithm, which enables the detection of objects other than faces. The HOG-based object detector in Dlib is capable of accurately detecting various objects, such as vehicles, pedestrians, and animals, making it suitable for applications like autonomous driving, surveillance, and object tracking.

Dlib also offers machine learning algorithms, including support for Support Vector Machines (SVM), k-nearest neighbors (k-NN), and various deep learning architectures. These algorithms allow developers to build custom models for classification, regression, clustering, and other machine learning tasks. The library provides efficient tools for feature extraction, model training, and evaluation, enabling the development of robust machine learning pipelines.

Dlib incorporates tools for image processing, including geometric transformations, image I/O, matrix manipulation, and basic image editing operations. It offers utilities for handling image files in various formats, such as JPEG, PNG, and BMP, and supports image resizing, cropping, and rotation.

Dlib's capabilities extend beyond computer vision to include tools for numerical optimization, graph and networking operations, and data structures. It provides implementations of popular algorithms like gradient descent, network flow, and graph search algorithms, allowing developers to solve a wide range of optimization and graph-related problems efficiently.

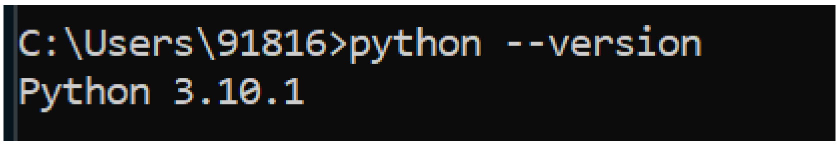
The versatility and performance of Dlib have made it a popular choice for numerous applications, including face detection and recognition systems, object detection and tracking, image processing pipelines, and machine learning tasks. Its well-documented API, extensive set of examples, and active user community contribute to its ease of use and support for developers.

## How to install dlib library

First of all we have to set up the environment for the installation of dlib library. There are a few things that should be pre-installed before installing the dlib library. If these things aren't properly installed or set up the dlib will not get installed properly. Let us see how we can do it.

## Step 1: Install Python

Install Python on Windows. and after that to check whether Python is properly installed or not check the python version using the below command.

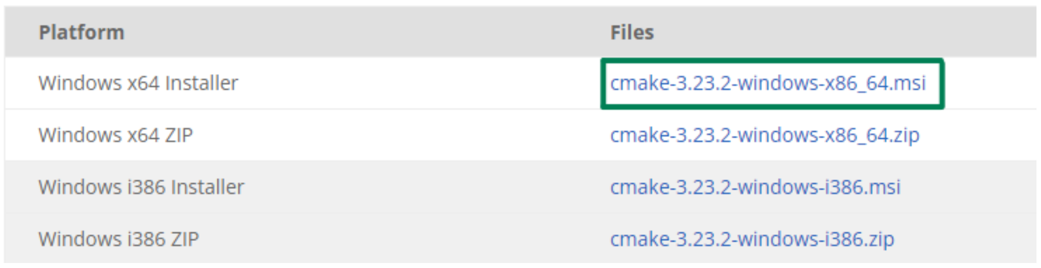


```
C:\Users\91816>python --version
Python 3.10.1
```

fig 4.6 install python

## Step 2: Install CMake

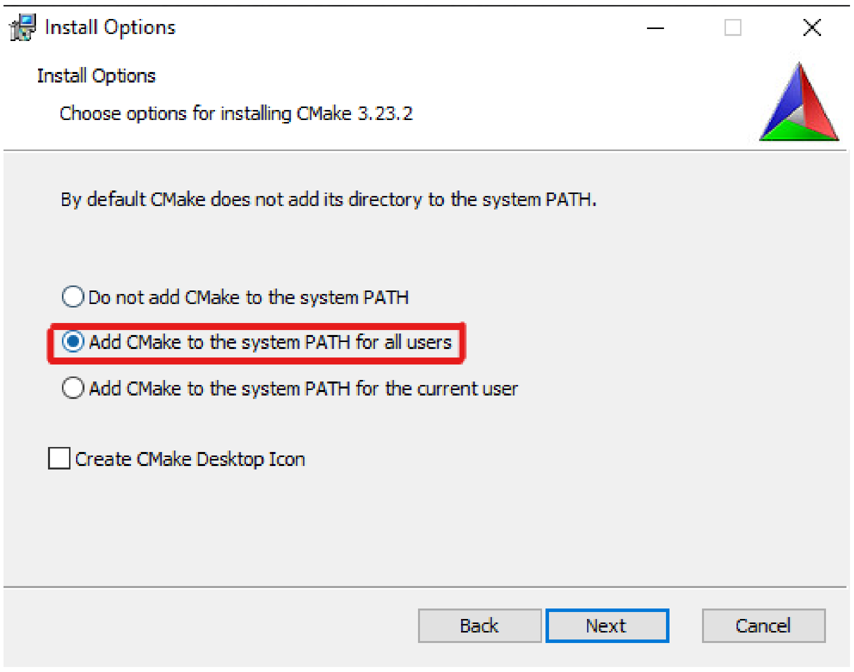
Install CMake from its official website https://cmake.org/download/ and make sure choose the right version according to your system configuration.



| Platform | Files |
|---|---|
| Windows x64 Installer | cmake-3.23.2-windows-x86_64.msi |
| Windows x64 ZIP | cmake-3.23.2-windows-x86_64.zip |
| Windows i386 Installer | cmake-3.23.2-windows-i386.msi |
| Windows i386 ZIP | cmake-3.23.2-windows-i386.zip |

fig 4.7 install Cmake

While installing CMake select Add CMake to the system PATH to avoid any error in the next steps.



Install Options

Install Options

Choose options for installing CMake 3.23.2

By default CMake does not add its directory to the system PATH.

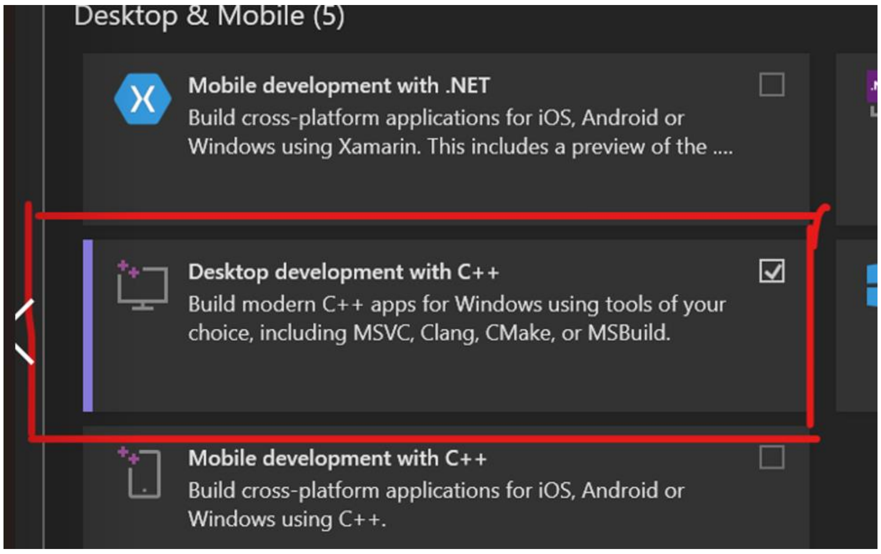
- ○ Do not add CMake to the system PATH
- ● Add CMake to the system PATH for all users
- ○ Add CMake to the system PATH for the current user

☐ Create CMake Desktop Icon

fig 4.8 setting path for Cmake

## Step 3: Install visual studio

Install the **c++ c**ompile**r** of the visual studio code community version. For that go to the visual studio code official website https://visualstudio.microsoft.com/visual-cpp-build-tools/ and download the visual studio code community version after that as you can see in the below image, select Desktop development with ***c++*** while installing VS code.



Desktop & Mobile (5)

**Mobile development with .NET** ☐
Build cross-platform applications for iOS, Android or Windows using Xamarin. This includes a preview of the ....

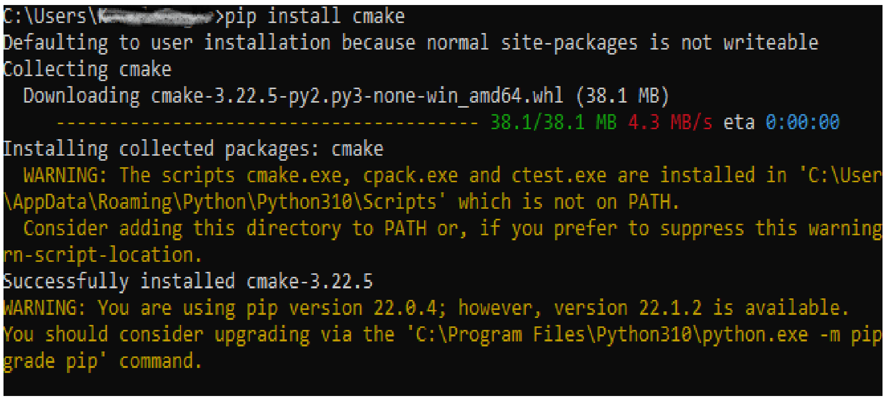
**Desktop development with C++** ☑
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

**Mobile development with C++** ☐
Build cross-platform applications for iOS, Android or Windows using C++.

fig 4.9 installing c++ in visual studio

## Step 4: Install cmake module

After you have installed visual studio [Desktop development with c++] successfully, now go to your command prompt and type "pip install cmake".

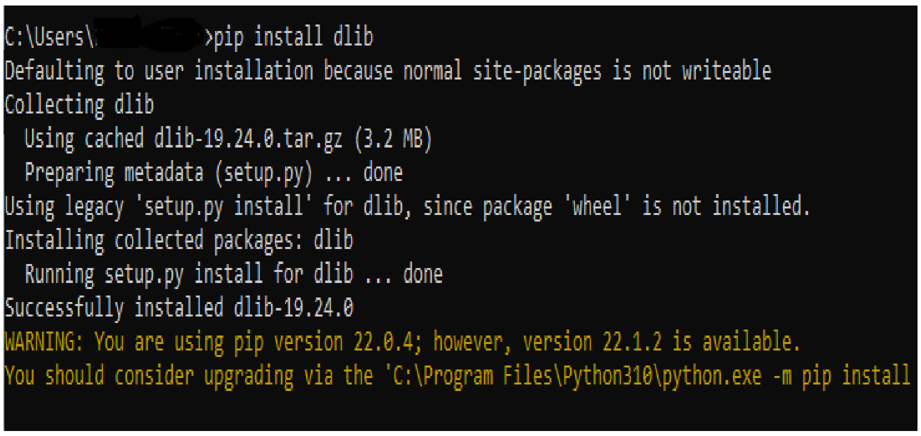


```
C:\Users\[redacted]>pip install cmake
Defaulting to user installation because normal site-packages is not writeable
Collecting cmake
  Downloading cmake-3.22.5-py2.py3-none-win_amd64.whl (38.1 MB)
     ---------------------------------------- 38.1/38.1 MB 4.3 MB/s eta 0:00:00
Installing collected packages: cmake
  WARNING: The scripts cmake.exe, cpack.exe and ctest.exe are installed in 'C:\User
\AppData\Roaming\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning
rn-script-location.
Successfully installed cmake-3.22.5
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Program Files\Python310\python.exe -m pip
grade pip' command.
```

fig 4.10 Cmake module installing

## Step 5: Install dlib library

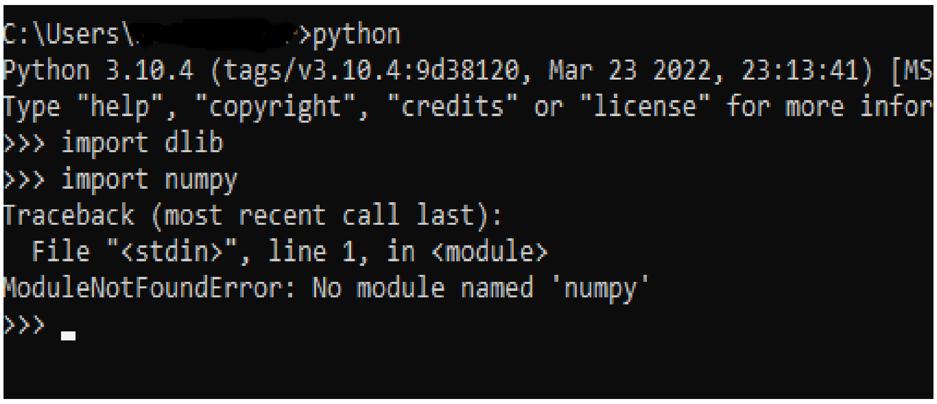
After you have installed cmake module successfully, go ahead and install the dlib library as shown in below image.



```
C:\Users\[redacted]>pip install dlib
Defaulting to user installation because normal site-packages is not writeable
Collecting dlib
  Using cached dlib-19.24.0.tar.gz (3.2 MB)
  Preparing metadata (setup.py) ... done
Using legacy 'setup.py install' for dlib, since package 'wheel' is not installed.
Installing collected packages: dlib
  Running setup.py install for dlib ... done
Successfully installed dlib-19.24.0
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Program Files\Python310\python.exe -m pip install
```

fig 4.11 dlib installation

Now, the dlib library is installed successfully and to verify the installation of dlib library open command prompt and type the command as shown in the below image.



```
C:\Users\[redacted]>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MS
Type "help", "copyright", "credits" or "license" for more infor
>>> import dlib
>>> import numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'numpy'
>>> _
```

fig 4.12 import library

# CHAPTER 5

# IMPLEMENTATION

## Main.py

```
import os
import pickle
import numpy as np
import cv2
import face_recognition
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
from datetime import datetime

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendacerealtime-bf845-default-rtdb.firebaseio.com/",
    'storageBucket':"faceattendacerealtime-bf845.appspot.com"
})
bucket = storage.bucket()

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

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

```
# 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)))
# print(len(imgModeList))

# 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)
        # print("matches", matches)
        # print("faceDis", faceDis)

        matchIndex = np.argmin(faceDis)
        # print("Match Index", matchIndex)

        if matches[matchIndex]:
            # print("Known Face Detected")
            # print(studentIds[matchIndex])
            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:
            # Get the Data
            studentInfo = db.reference(f'Students/{id}').get()
            print(studentInfo)
            # Get the Image from the storage
            blob = bucket.get_blob(f'Images/{id}.png')
```

```
array = np.frombuffer(blob.download_as_string(), np.uint8)
    imgStudent = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
    # Update data of attendance
    datetimeObject = datetime.strptime(studentInfo['last_attendance_time'],
                                       "%Y-%m-%d %H:%M:%S")
    secondsElapsed = (datetime.now() - datetimeObject).total_seconds()
    print(secondsElapsed)
    if secondsElapsed > 30:
        ref = db.reference(f'Students/{id}')
        #studentInfo['total_attendance'] += 1
        ref.child('total_attendance').set(studentInfo['total_attendance'])
        ref.child('last_attendance_time').set(datetime.now().strftime("%Y-%m-
        %d %H:%M:%S"))

 else:
        modeType = 3
        counter = 0
        imgBackground[44:44 + 633, 808:808 + 414] =
        imgModeList[modeType]

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("Webcam", img)
    cv2.imshow("Face Attendance", imgBackground)
    cv2.waitKey(1)
```

**EncodeGenerator.py**

```
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import  storage

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendacerealtime-bf845-default-rtdb.firebaseio.com/",
    'storageBucket':"faceattendacerealtime-bf845.appspot.com"
})



# Importing student images
folderPath = '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(studentIds)
```

```
def findEncodings(imagesList):
    encodeList = []
    for img in imagesList:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    return encodeList



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")
```

**AddDatatoDatabase.py**

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendacerealtime-bf845-default-rtdb.firebaseio.com/"
})


ref = db.reference('Students')

data = {
    "1154":
        {
            "name":"Parth Maurya ",
            "major":"Computer Application",
            "starting_year":"2021",
            "total_attendance":"6",
            "standing":"G",
            "year":4,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
"1056":
        {
            "name":"Deepsikha Singh",
            "major":"Economics",
            "starting_year":"2018",
            "total_attendance":"12",
            "standing":"B",
            "year":2,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
"1011":
        {
            "name":"Sharad Gupta",
            "major":"Physics",
            "starting_year":"2020",
            "total_attendance":"7",
            "standing":"G",
            "year":2,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
"1181":
        {
            "name":"Raj Singh",
            "major":"Computer Science",
            "starting_year":"2018",
            "total_attendance":"5",
            "standing":"G",
            "year":4,
            "last_attendance_time": "2022-06-14 00:54:34"
        },
"1126":
    {
            "name":"Yash Raj Singh ",
            "major":"MCA",
            "starting_year":"2021",
            "total_attendance":"4",
            "standing":"G",
            "year":2,
            "last_attendance_time": "2022-09-28 00:54:34"


    }
}

for key,value in data.items()
```

# CHAPTER 6

# RESULTS AND DISCUSSION

## 6.1 Storage

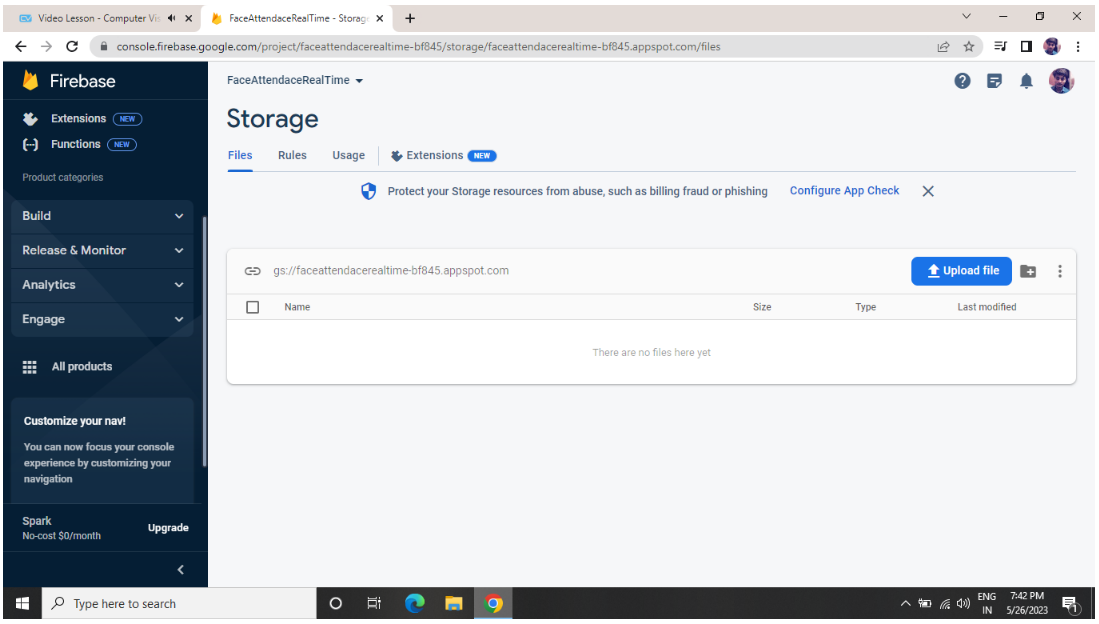
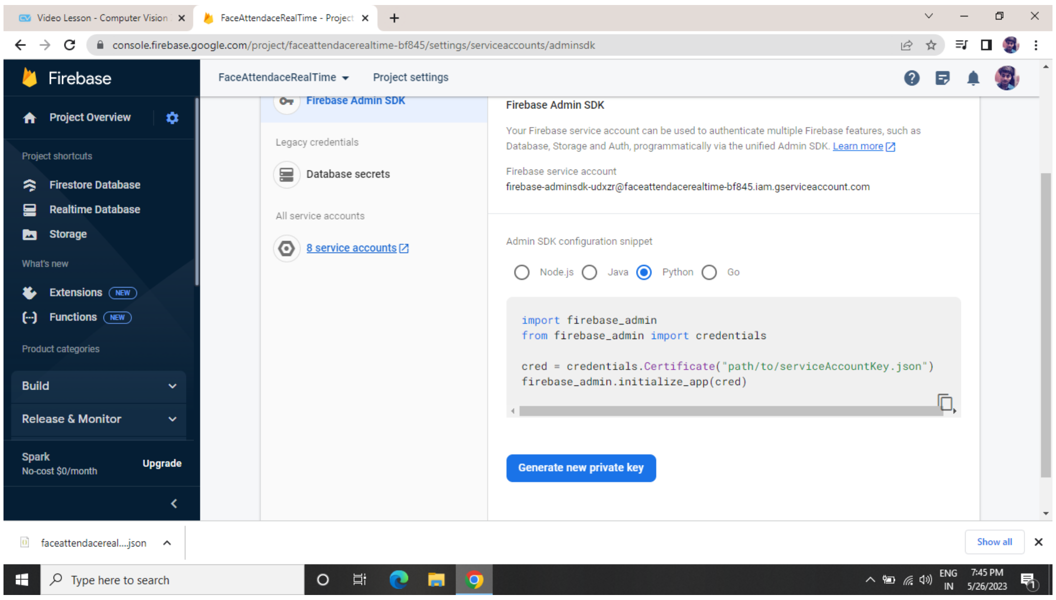


fig 6.1 Database Storage

## 6.2 Connecting to Firebase



Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. Learn more

Firebase service account
firebase-adminsdk-udxzr@faceattendacerealtime-bf845.iam.gserviceaccount.com

Admin SDK configuration snippet

Node.js Java [✓ Python] Go

```
import firebase_admin
from firebase_admin import credentials

cred = credentials.Certificate("path/to/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
```

Generate new private key

Fig 6.2 Connecting to firebase

## 6.3 Attendance System

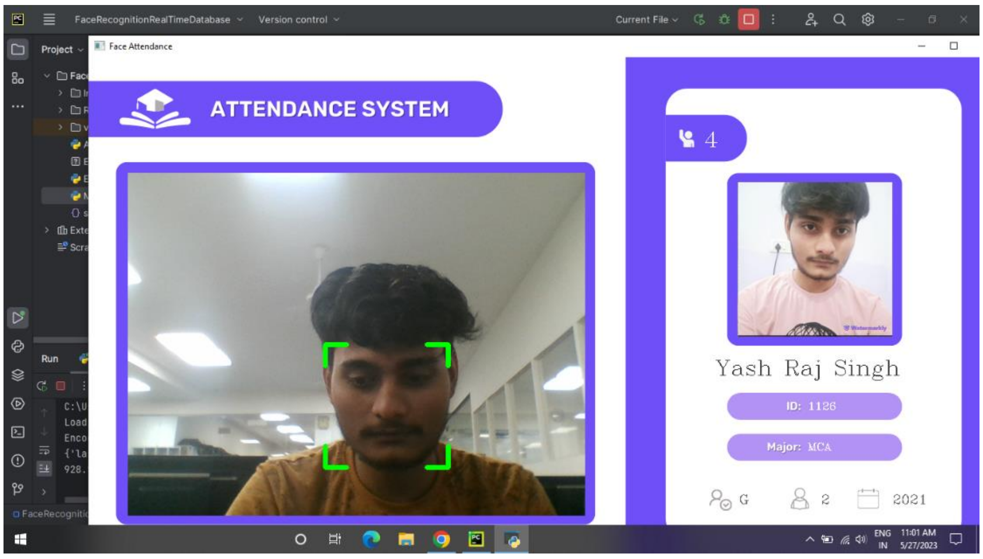


Fig 6.3  Attendance System

## 6.4 Marking Attendance

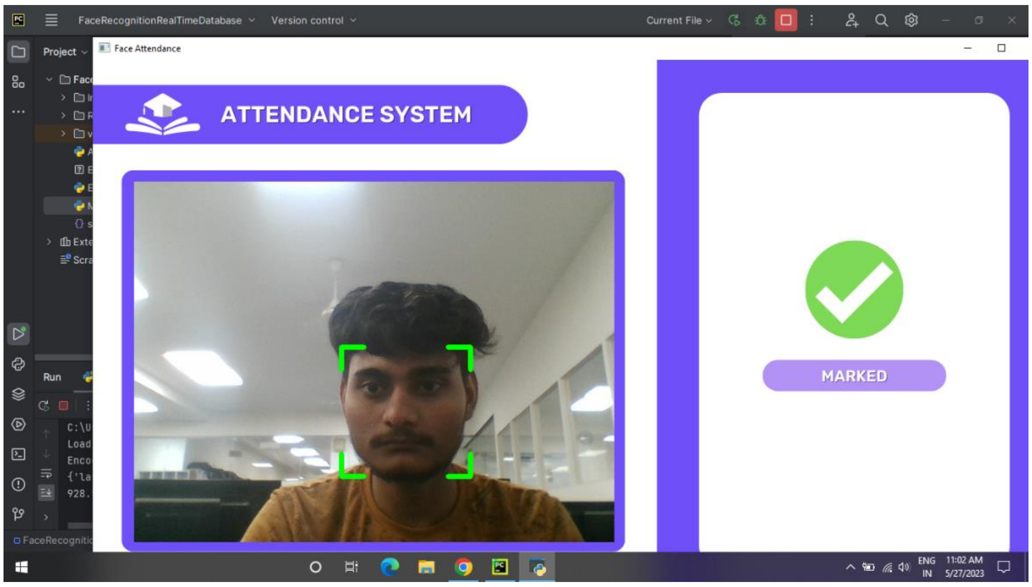


Fig 6.4 Marking attendance

## 6.5 Verifying Attendance

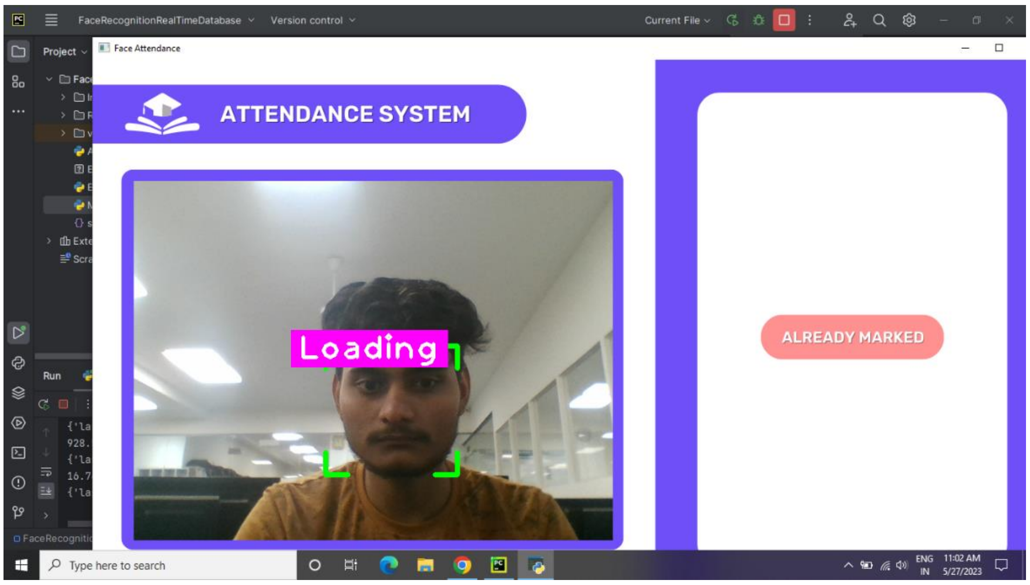
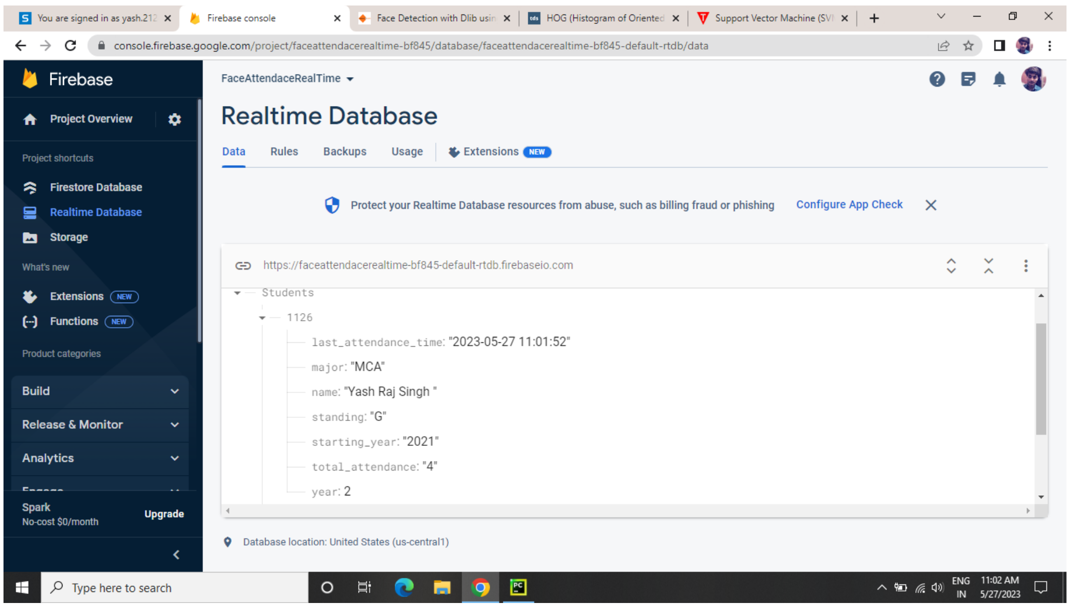


Fig 6.5 Verifying Attendance

## 6.6 Realtime Database



Realtime Database

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

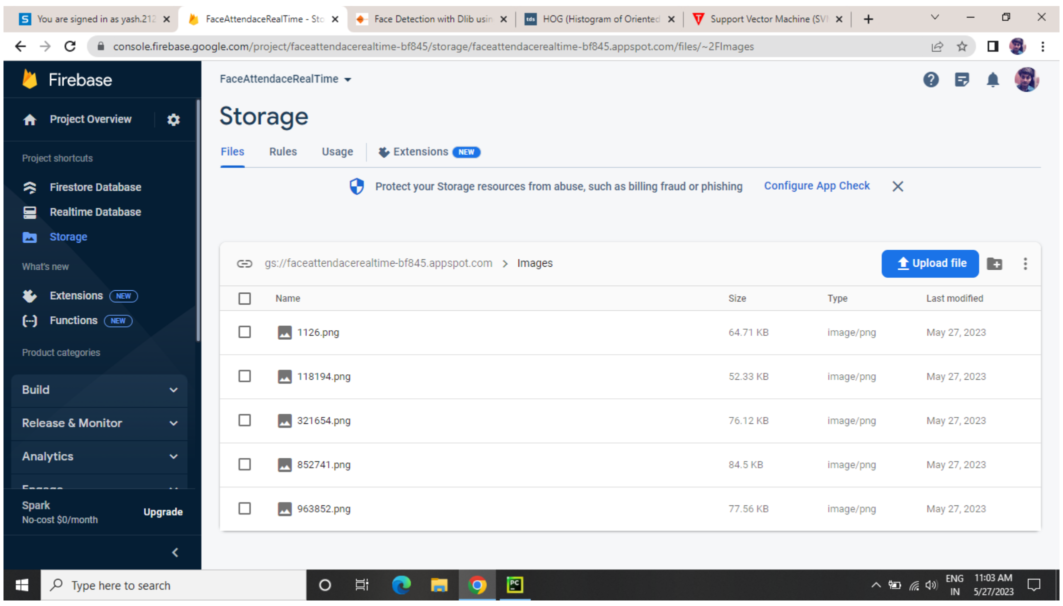
https://faceattendacerealtime-bf845-default-rtdb.firebaseio.com

```
Students
  1126
    last_attendance_time: "2023-05-27 11:01:52"
    major: "MCA"
    name: "Yash Raj Singh "
    standing: "G"
    starting_year: "2021"
    total_attendance: "4"
    year: 2
```

Database location: United States (us-central1)

Fig 6.6 Realtime Database

## 6.7 Images Dataset



Protect your Storage resources from abuse, such as billing fraud or phishing

gs://faceattendacerealtime-bf845.appspot.com > Images

| Name | Size | Type | Last modified |
|---|---|---|---|
| 1126.png | 64.71 KB | image/png | May 27, 2023 |
| 118194.png | 52.33 KB | image/png | May 27, 2023 |
| 321654.png | 76.12 KB | image/png | May 27, 2023 |
| 852741.png | 84.5 KB | image/png | May 27, 2023 |
| 963852.png | 77.56 KB | image/png | May 27, 2023 |

Fig 6.7 images dataset

# CHAPTER 7

# TESTING

## 7. Testing

| Sl.No | Action | Inputs | Expected Output | Actual Output | Test Result |
|---|---|---|---|---|---|
| 1 | Capture Images | A Person’s Face | Images are Captured and Stored | Images are captured and stored | Pass |
| 2 | Train the image Dataset | Stored images of a face | Create Histograms and store values | Histograms are created and values are stored | Pass |
| 3 | Face Recognition | A live stream of a person’s face | Name of detected person is displayed on the screen | Name of detected person is displayed on the screen | Pass |
| 4 | Update attendance for multiple people at once | Multiple faces from a live video stream | Update Attendance for all faces detected | Attendance is Updates only for a single face | Fail |
| 5 | Detect more than 7 faces | 7 people facing the camera | Detect all 7 faces facing the camera | Only 5 faces are detected at a time | Fail |

Table 7. Testing

# CHAPTER 8

# CONCLUSION

The Face recognition attendance system is meant to unravel the problems of existing manual systems. We have used the face recognition concept to mark the attendance of students and make the system better. The system performs very well in different poses and variations. In the future this system needs to be improved because this system sometimes fails to recognize students, also we have some processing limitations, working with a system of high processing may end in even better performance of this technique.

The proposed system is able to achieve high precision and more accuracy for recognizing faces and marking attendance in less computational complexity. The economical and physical interference of the system is also less. We have successfully used HOG and SVM algorithms. Using these advanced algorithms, we were able to achieve accuracy of 99.20% and F1 score of 0.9924. We have also observed that our systems accuracy varies with various conditions such as lightning but still we were able to achieve accuracy of 99.20%. We have observed that our proposed system is efficient than any other system using different algorithms such as HOG and SVM. Our system can be further improvised and can be used for different purposes such as for tracking person by including features such as instead of using a single camera, we can use multiple cameras at different locations and interconnect them to form a network which can track a person.

# REFERENCES

1. Kawaguchi, Y., Shoji, T., Lin, W., Kakusho, K., &amp; Minoh, M. (1970, January 1). [PDF] Face Recognition-based Lecture Attendance System: Semantic Scholar. [PDF] Face Recognition-based Lecture Attendance System | Semantic Scholar. https://www.semanticscholar.org/paper/Face-Recognition-based-Lecture-Attendance-System-Kawaguchi-Shoji/4b6811cd2a7a6924fed4967 c2b755c0942ca5351.

2. J. K. J. Julina and T. S. Sharmila, "Facial recognition using histogram of gradients and support vector machines," 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), 2017, pp. 1-5, doi: 10.1109/ICCCSP.2017.7944082.

3. H. Mady and S. M. S. Hilles, "Face recognition and detection using Random Forest and combination of LBP and HOG features," 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), 2018, pp. 1-7, doi: 10.1109/ICSCEE.2018.8538377.

4. S. Hapani, N. Prabhu, N. Parakhiya and M. Paghdal, "Automated Attendance System Using Image Processing," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697824.

5. Nithya. C, Ramya Bharathi. M, Santhini. M, Sowmya. R, 2020, Face Recognition Based-Automatic Attendance Management System, International Journal of Engineering Research & Technology (IJERT) NCICCT – 2020 (Volume 8 – Issue 08),

6. Algorithm for Efficient Attendance Management: Face ... (n.d.). http://ijcsi.org/papers/IJCSI-9-4-1-146-150.pdf.

7. Study of Face Recognition Techniques: A Survey. (n.d.).https://thesai.org/Downloads/Volume9 No6/Paper_6-Study_of_Face_Recognition _Techniques.pdf.

8. Albiol, A., Monzo, D., Martin, A., Sastre, J., &amp; Albiol, A. (2008, April 7). Face recognition using HOG–EBGM. Pattern Recognition Letters. https://www.sciencedirect. com/science/article/pii/S0167865508001104.

9. Chen, T., Gao, T., Li, S., Zhang, X., Cao, J., Yao, D., &amp; Li, Y. (2021, April 6). A novel face recognition method based on fusion of LBP and HOG. Institution of Engineering and Technology. https://ietresearch.onlinelibrary.wiley.com/doi/fu ll/10.1049/ipr2.12192.

10. Face Recognition and Human Tracking Using GMM, HOG and SVM ... (n.d.). https://link.springer.com/article/10.1007/s40745-017-0123-2.

11. H. Rathod, Y. Ware, S. Sane, S. Raulo, V. Pakhare and I. A. Rizvi, "Automated attendance system using machine learning approach," 2017 International Conference on Nascent Technologies in Engineering (ICNTE), 2017, pp. 1-5, doi: 10.1109/ICNTE.2017.7947889.

12. Ijes, T. I. T. (n.d.). An Automatic Attendance System Using Image processing. Academia .edu.https://www.academia.edu/19491815/An_Automatic_Attendance_System_Using_Image_processing.