A Project Report

on

Attendance System Using Face Recognition

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

Under The Supervision of
Mr. Gokul V. Rajan
Assistant Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1050043- SOUMYA SHREE

18SCSE1050015 - RICHA

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **" Attendance System Using Face Recognition "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. GOKUL V. RAJAN, Assistant Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

18SCSE1050043- SOUMYA SHREE
18SCSE1050015 - RICHA

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr.Gokul V. Rajan, Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1050043 - SOUMYA SHREE, 18SCSE1050015 - RICHA** has been held on _____and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ABSTRACT

Attendance marking system is very time-consuming and prone to error. This project proposed an automated attendance system that uses face recognition technology. The system was able to identify the faces of the students and determine the appropriate attendance mark.

Our app asks the student to fill the details and take image of the student. After completing the process, it saves the images and sends them to Training Image folder.

By clicking Track Image button, the camera of running machine will automatically open. If the face is recognized by system, the name and image of the person shown are shown on Image.

Table of Contents

List of Figures

List of Tables

Acronyms

LBPH	Logical Binary Pattern Histogram
ML	Machine Learning
UML	Unified Modelling Language
MAS	Manual Attendance System
AAS	Automatic Attendance System

CHAPTER-1

Introduction

The flow process in Face Recognition-based attendance management systems begins with the ability to detect and distinguish frontal faces from a database input dataset. In today's society, good classroom control has been proved to increase student engagement during lectures. The importance of high levels of student engagement cannot be overstated. Face detection and identification are not new concepts in modern society. The human mind's ability to recognise certain individuals is astounding. Face detection is the process of detecting faces with various expressions, sizes, and angles in photos with complex light and backdrop and feeding back face parameters. Face recognition analyses patterns and detects one or more faces in an image by processing it. To identify a match, this technique employs algorithms that extract information and compare them to a database.
This project is being carried out in response to the concerns stated about the methods used by lecturers to take attendance during lectures. These days, technology attempts to convey a significant amount of knowledge-based technical innovation. Machine learning is an intriguing domain in which a machine can teach itself by producing a suitable output during testing using various learning methods. Attendance is now regarded as critical for both students and teachers in educational institutions. With the developments in machine learning technology, the computer can now automatically recognise the students' attendance performance and keep a record of it. The reasons for building up this particular section Attendance system based on Face Recognition with LBPH.
In general, the attendance system of the student can be maintained in two different forms namely,

- Manual Attendance system (MAS)
- Automated Attendance System (AAS)

Manual Student Attendance Management is a technique in which a teacher responsible for a specific subject calls the students' names and manually records their attendance. Manual attendance can be a time–consuming operation, and it's not uncommon for the teacher to overlook someone, or for pupils to respond to the absence of their friends many times. When we consider the usual method of taking attendance in the classroom, we encounter a difficulty. We use Automated Attendance System (AAS) to tackle all of these problems. AAS is a technique that uses face recognition technology to automatically estimate a student's presence or absence in a classroom. It is also feasible to detect if a student is sleeping or awake during a lecture, and it can be used to ensure a student's presence during exam sessions. The presence of students may be determined by capturing their faces on a high–definition monitor video streaming service, making the machine's ability to understand the presence of all pupils in the classroom more dependable.
The two common Human Face Recognition techniques are:

- Feature-based approach
- Brightness-based approach.

The feature-based approach, also known as a local face recognition system, is used to point out important facial characteristics such as the eyes, ears, nose, and mouth. The brightness-based approach, also known as the global face recognition system, is used to recognize people from all around the world.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

Processor intel core i5 8th Gen	i5 8th Gen
Graphics Processing Unit (GPU)	NVIDIA GEFFORCE
Random Access Memory (RAM)	8GB
Hard Disk	1TB

SOFTWARE DEPENDENCIES:

Requirement	**Version**
Open CV	4.3.0
tkinter GUI	8.6
NumPy	1.18.1
Pandas	1.0.3
PIL	1.1.7

SOFTWARE REQUIREMENTS:

Operating system	Windows 10,8,7
Language	Python 3.6 version
Editor	Visual Studio Code
Design Tool	Star UML

CHAPTER-2

Literature Survey

In this chapter, a brief overview of studies made on face recognition will be introduced alongside some popular face detection and recognition algorithms. This will give a general idea of the history of systems and approaches that have been used so far.

Sr No.	Author	Algorithm	Problem	Summary
1.	Visar Shehu [1]	PCA	The recognition rate is 56%, having a problem to recogn	Using HAAR Classifier and computer vision algorithm to implement face recognition
2.	Viola, M. J. Jones [8]	Viola and Jones algorithm	In Viola and Jones the result depends on the data and weak classifiers. The quality of the final detection depends highly on the consistence of the training set. Both the size of the sets and the interclass variability are important factors to take in account.The analysis shows very bad results when in case of multiple person with different sequence	The training of the data should be done in correct manner so that the quality final detection will increase.System overview should contain the overall architecture that will give the clear and comprehensive information of the project.
3.	Kasar, M., Bhattacharyya, D. and Kim, T. [9]	Neural-Network	Detection process is slow and computation is complex.Overall performance is weaker than ViolaJones algorithm.	Accurate only if large size of image was trained.
4.	Pratiksha M. Patel [10]	Contrast Limited Adaptive Histogram Equalization (CLAHE)	More sensitive to noise compared to histogram equalization.	Unlike, HE which works on entire image, it works on small data regions. Each tile's contrast is enhanced to ensure uniformly distributed histogram. Bilinear interpolation is then used to merge the neighboring tiles. Advantage:- It prevent over enhancement as well as noise amplification.
5.	Suman Kumar Bhattacharyya & Kumar Rahul. [6]	Fisher face/ LDA (Linear Discriminant Analysis	Bigger database is required because images of different expression of the individual have to be trained in same class.It	Images of individual with different illumination, facial expressions able to be recognized if more samples are trained.

		)	depend more on databa	
6.	Varsha Gupta, Dipesh Sharma [7]	Successive mean quantizatio n transform (SMQT)	The region contain very similar to grey value regions will be misidentified as face.	1. Capable to deal with lighting problem in object detection. 2. Efficient in computation
7.	Syen navaz [2]	PCA, ANN	Low accuracy with the big size of images to train with PCA Hight Computational cost due to com	Using PCA to train and reduce dimensionality and ANN to classify input data and find the pattern. Using PCA and ANN to do a better attendance result

Overview of Face Recognition: Most face recognition systems rely on face recognition algorithms to complete the following functional task. The figure below shows a simplified diagram from the framework for face recognition.

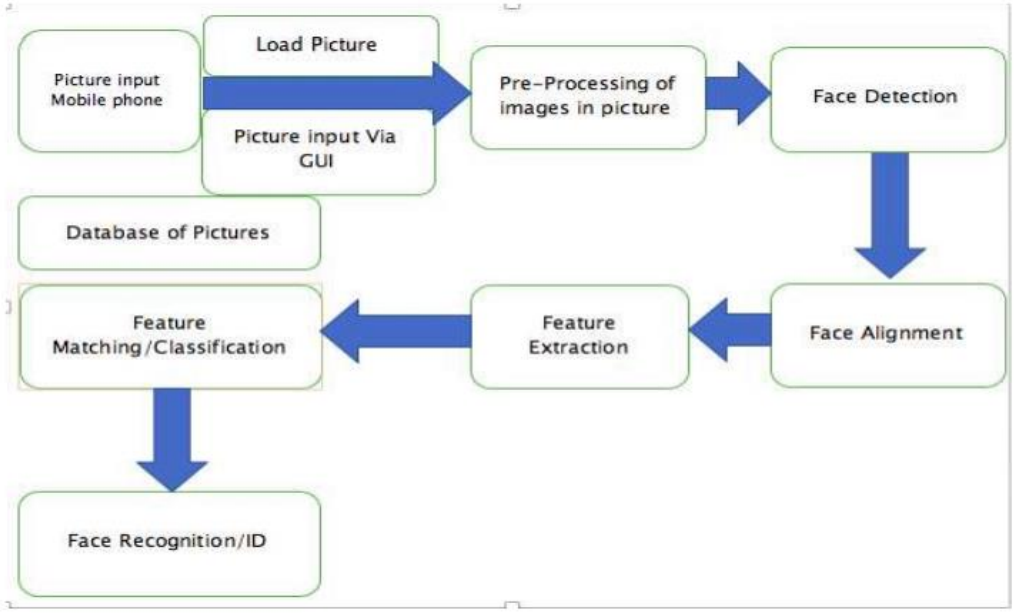


Fig 1:Working of Face detection and recognition

Face Detection, also known as face detector, will detect any given face in the given image or input dataset, as shown in the diagram above. Face localization uses bounding boxes to recognise where the faces are in the given image/input dataset. Face Alignment is when the system locates a face and aligns landmarks like the nose, eyes, chin, and mouth in order to extract features. Feature extraction is the process of extracting essential features such as the eyes, nose, and mouth so that they can be tracked. Matching and categorization of features. Matches a face with a trained data set of images from a database with a small number of images. Face recognition determines whether a recognized face is positive or negative based on feature matching and classification from a referenced facial image. Face detection is the process of detecting a face in a digital image using computer software designed specifically for this purpose.

Face identification implies determining a face's position in the image plane as well as its size or scale. As shown in the diagram, detecting a face in a digital image is a precondition for any subsequent face recognition or processing software. The technology's concept, known as the Student Attendance System, was realized using a machine learning approach. This technology automatically detects a student's performance and keeps track of important information such as attendance. As a result, by identifying the student's face, the student's attendance can be determined. When you recognize the details of attendance, you can move on to the next step. The Automated Attendance System with Face Recognition proposes that the system is based on face detection and detection and recognition algorithms, which are used to automatically detect the students face as they enter the classroom and the system is capable of recognizing him and marking his attendance. The effectiveness of the images is also being discussed in order to allow much faster image recognition.

CHAPTER-3

PROJECT DESIGN

UML Diagrams

The Unified Modelling Language (UML) is a modelling language that can be used for a variety of purposes. The primary goal of UML is to establish a standard for visualising the design of a system. It looks a lot like blueprints in other branches of engineering.

UML is a visual language rather than a programming language. UML diagrams are used to depict a system's behavior and structure. UML is a modelling, design, and analysis tool for software engineers, businesspeople, and system architects. Unified Modelling Language was approved as a standard by the Object Management Group (OMG) in 1997. Since then, OMG has been in charge of it. In 2005, the International Organization for Standardization (ISO) accepted UML as a standard. UML has been updated throughout time and is examined on a regular basis.

Goals of UML:
The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
2.Provide extensibility and specialization mechanisms to extend the core concepts.
3.Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modelling language.
5. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
6. Integrate best practice

3.1 The conceptual model of UML

A conceptual model can be defined as a model which is made of concept and their relationships. A conceptual model is the first step before drawing UML diagrams. It helps to understand the entities in the real world and how they interact with each other.
To understand how the UML works, we need to know the three elements:

1. UML basic building blocks
2. Rules to connect the building blocks
3.Common mechanisms that apply throughout in the UML.

3.2 Types of Diagrams

UML diagrams are divided into three different categories such as,

- Structural diagram
- Behavioral diagram
- Interaction diagram

Structural diagrams
Structural diagrams are used to depict a system's static view. It denotes a component of a system that contributes to the overall structure of the system. A structural diagram depicts the system's many objects.

Following are the various structural diagrams in UML:
- Class diagram
- Object diagram
- Package diagram
- Component diagram
- Deployment diagram

Behavioral diagrams
Any real-world system can be represented in one of two ways: static or dynamic. If a system can be stated in both static and dynamic forms, it is said to be complete. A system's behavior is depicted in a behavioral diagram.
Structural diagrams are UML diagrams that deal with the static element of a system. Behavioral diagrams are UML diagrams that deal with the system's moving or dynamic parts.
Following are the various behavioral diagrams in UML:
- Activity diagram
- Use case diagram
- State machine diagram

Interaction diagrams
A subset of behavioral diagrams is an interaction diagram. It's used to show how a system's many use case elements interact. Interaction diagrams are used to depict how two things interact and how data flows between them.
Following are the various interaction diagrams in UML:
- Timing diagram
- Sequence diagram
- Collaboration diagram

3.2.1 Use case diagram:
Use Case Diagram captures the system's functionality and requirements by using actors and use cases. Use Cases model the services, tasks, function that a system needs to perform. Use cases represent high-level functionalities and how a user will handle the system. Use-cases are the core concepts of Unified Modelling language modeling.
A Use Case consists of use cases, persons, or various things that are invoking the features called as actors and the elements that are responsible for implementing the use cases. Use case diagrams capture the dynamic behavior of a live system. It models how an external entity interacts with the system to make it work. Use case diagrams are responsible for visualizing the external things that interact with the part of the system.

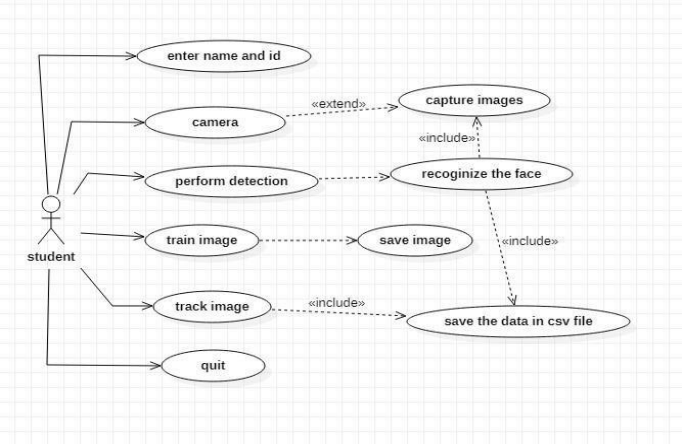


Fig 2: Use Case Diagram

3.2.2 Class diagram:

A class diagram depicts classes, properties, operations, and their relationships to provide an overview of a software system. The class name, properties, and operation are all divided into different compartments in this diagram. The Class Diagram depicts the various categories of items in the system as well as the various sorts of relationships between them. Almost all Object-Oriented Methods can be used with this modelling method. Another class can be referred to by a class. It is possible for a class to have its own objects or to inherit from other classes. The Class Diagram aids in the development of code for software applications.

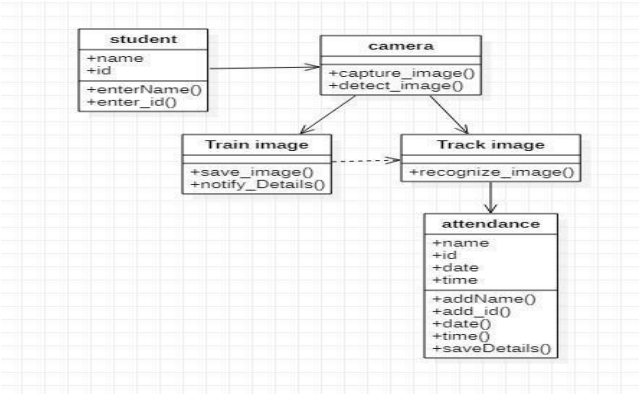


Fig 3: Class Diagram

3.2.3 Sequence Diagram:

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

1. Represent the details of a UML use case.
2. Model the logic of a sophisticated procedure, function, or operation.
3. See how objects and components interact with each other to complete a process.
4. Plan and understand the detailed functionality of an existing or future scenario.

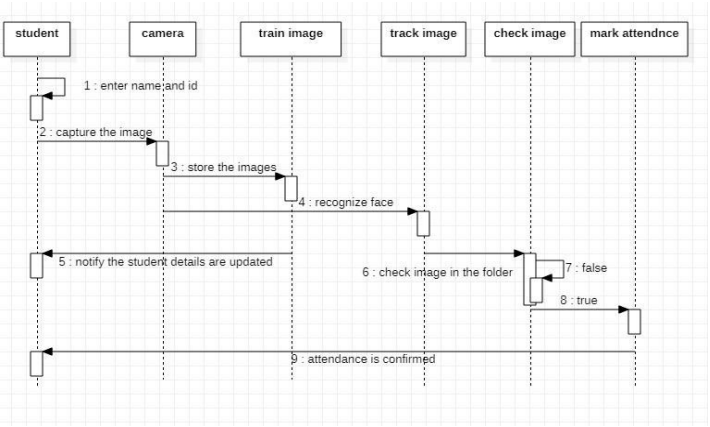


Fig 4: Sequence Diagram

3.2.4 Activity Diagram:

Activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called object-oriented flowchart. Activity diagrams consist of activities that are made up of actions which apply to behavioral modeling technology. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

Activity diagram is suitable for modelling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another.These systems can be database, external queues, or any other system.

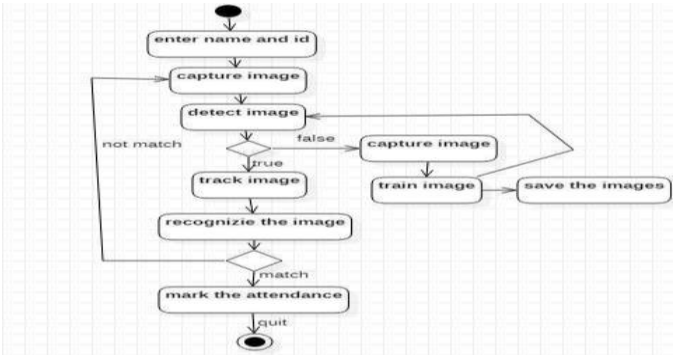


Fig 5: Activity Diagram

CHAPTER-4

Implementation and Description of project Modules

Methodology:

a) Local Binary Pattern Histogram (LBPH)
b) Haar Cascade Classifier

The project deployed on Haar Cascade classifier to find the positive and negative of the face and LBPH (Local binary pattern histogram) algorithm for face recognition by using python programming and OpenCV library.

4.1 Local Binary Pattern Histogram (LBPH):

In computer vision, a local binary pattern is a sort of visual descriptor used for categorization. LBP is a part of the Texture Spectrum model, which was first presented in 1990. In 1994, LBP was initially described. It has since been discovered to be a useful feature for texture classification; it has also been discovered that combining LBP with the Histogram of Oriented Gradients (HOG) descriptor significantly increases detection performance on specific datasets. Silva et al. compared numerous enhancements to the original LBP in the field of background subtraction in 2015. Bouwmans provides a comprehensive overview of the various LBO variations. Mahotas is a Python library for computer vision that includes an LBP implementation. As of version 2, the cascade classifiers in Open CV support LBPs.

The LBP Library is a collection of eleven local binary patterns (LBP) algorithms designed to solve the problem of background subtraction. Face recognition using the Local Binary Patterns Histogram method (LBPH). It is one of the top performing texture descriptors and is based on the local binary operator. The demand for facial recognition technologies is growing all the time. They're utilized for things like access control, surveillance, and unlocking smartphones. We will utilize LBPH to extract features from an input test image and match them with faces in the system's database in this project. In 2006, the Local Binary Pattern Histogram algorithm was proposed. It is based on a binary operator that is local to the user. Because of its computational simplicity and discriminative capability, it is commonly employed in facial recognition.
The steps involved to achieve this are:

- Creating dataset
- Face acquisition
- Feature extraction
- Classification

The LBPH algorithm is a part of OpenCV.

Steps:

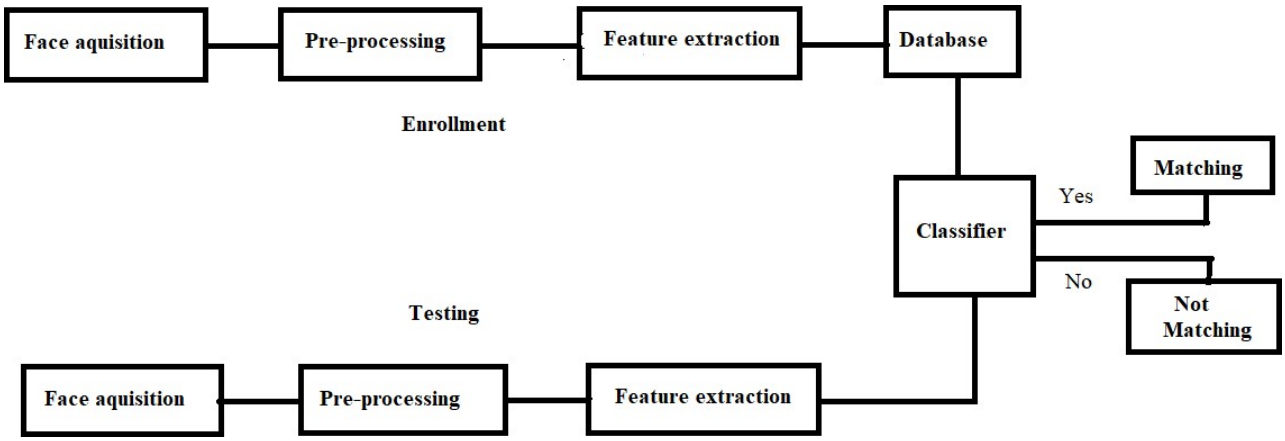


Fig 6:Process involved in LBPH

- Suppose we have an image having dimensions N x M.
- We divide it into regions of same height and width resulting in m x m dimension for every region.
- Local binary operator is used for every region. The LBP operator is defined in window of 3x3.

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

here '(Xc,Yc)' is central pixel with intensity 'Ic'. And 'In' being the intensity of the neighbor pixel

- Using median pixel value as threshold, it compares a pixel to its 8 closest pixels using this function.

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- It is set to 1 if the value of the neighbor is larger than or equal to the center value, else it is set to 0.
- As a result, the 8 neighbors yield a total of 8 binary value.
- We get an 8 bit binary number after combining these values, which is then converted to a decimal number for our convenience.

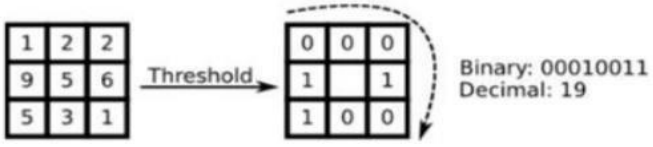


Fig 7: LBPH Pixel Value

- The pixel LBP value is a decimal number with a range of 0-255.
- The region's histogram is constructed after the LBP values are generated by counting the number of similar LBP values in the region.
- Following the generation of a histogram for each region, the histograms are merged to generate a single histogram, which is known as the image's feature vector.

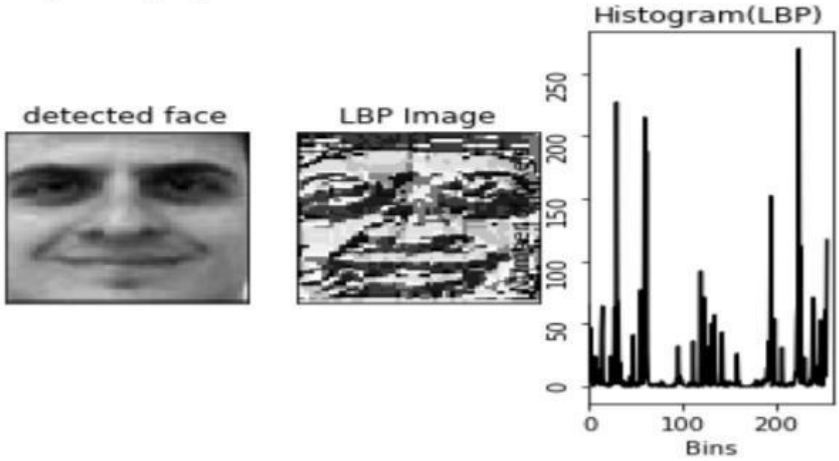


Fig 8: Feature vector of the image

- The histograms of the test image and the photographs in the database are now compared, and the image with the closest histogram is returned.
- Many techniques, such as Euclidean distance, chi-square, and absolute value, can be used to do this.
- The Euclidean distance is obtained by comparing the test picture features with features contained in the dataset. The matching rate is determined by the minimum distance between the test and the original image.

$$d(a,b) = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2}$$

- If the test image is recognized, we get an image ID from the database as an output.

LBPH can recognize both side and front faces, and it is unaffected by changes in lighting, making it more adaptable.

4.2 Haar Cascade:

Haar Cascade is a machine learning object recognition approach based on the concept of features developed by Paul Viola and Michael Jones in their 2001 paper "Rapid Object Detection using a Boosted Cascade of Basic Features." It's a machine-learning-based method in which a cascade function is learned using a large number of positive and negative images. After that, it's used to find things in their photos. Fortunately, OpenCV includes a preconfigured Haar Cascade algorithm that is divided into categories based on the photos that were used to train it.

Let's have a look at how this algorithm actually works. Haar Cascade is a concept using a 'filter' to extract features from photographs, analogous to the notion of the Kernel with convolutions Haar characteristics are the name for these filters.

Algorithm:

```
import numpy as np
import cv2
face_cascade = cv2. Cascade Classifier (“haarcascade_frontalface_default. Xml”)
eye_cascade =cv2.CascadeClassifier (1’haarcascade_eye.xml”)
img = cv2. imread (“image .jpg”)
gray =cv2. CvtColor (img, cv2.COLOR_BGR2GRAY)
faces = face_cascade. detectMultiScale (gray, 1.3, 5)
for (x, y, w, h) in faces:
img = cv2.rectangle (img, (x, y), (x+w, y+h), (255, 0, 0), 2)
roi_gray = gray [y: y+h, x: x+w]
roi_color = img[y: y+h, x: x=w]
eyes = eye_cascade.detectMultiScale (roi_gray)
for (ex, ey, ew, eh) in eyes:
cv2.rectangle (roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
cv2 .imshow (‘img’, img)
cv2.Waitkey (0)
cv2.destroyAllWindows ()
```

Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it. First step is to collect the Haar features.

- Haar Feature Selection
- Creating Integral images
- Adaboost Training
- Cascading Classifiers

As you can see, our algorithm performed admirably. If you look through the entire library of Haar Cascade algorithms, you'll notice that their unique models improve when trained on different aspects of a human's physical appearance, thus you can improve your model by detecting new characteristics. However, the majority of the traits we calculated are irrelevant. Consider the following image. Two good aspects are shown in the first row. The first attribute chosen appears to be the fact that the area around the eyes is frequently darker than the area around the nose and cheekbones. The second feature chosen is based on the fact that the eyes are darker than the nasal bridge. However, the same windows applied to cheeks or any other location are ineffective. So, how do we pick the best features from a list of 160000+ options? Adaboost is the one who accomplishes it.

We can do this by applying each feature to all of the training photos. It determines the appropriate threshold for each feature to identify the faces as positive or negative. However, there will probably be rate misclassifications. We choose characteristics with the lowest error rate, which implies they're the ones that best distinguish between face and non-facial photos. The Haar Cascade classifier uses the Haar Wavelet approach to break down pixels in images into squares based on their function. The "features" discovered are computed using "integral pictures" principles. Haar Cascades employ the Adaboost learning algorithm, which picks a small number of significant features from a large number in order to get an efficient output. Classifiers then employ cascade approaches to recognize faces in images. The Haar Cascade classifier is based on the Viola Jones detection technique, which is learned by feeding it a set of input faces and non-faces and training a classifier to recognize them. The Viola Jones face detection algorithm has been trained, and the weights have been saved to disc. All we have to do is apply the characteristics from the file to our

image; if there are faces in the image, we receive the face location. A Haar Cascade is essentially a classifier that detects the object for which it was trained from the source. Using high-quality photos and increasing the number of stages for which the classifier is trained gives better results. Face recognition can thus be made simple by employing the haar cascade classifier technique.

SYSTEM MODULE

1. Database Creation Module
2. Face Detection Module
3. Extraction and Database Matching Module
4. Report Generation Figure

The camera must be placed in a classroom where it can efficiently take photographs of all of the students. This photograph is now being processed to produce the desired results. The following is a basic description of the functionality:

The following are the characteristics of several system modules:
1. Create a database:
• Input: Provide input in the form of a single student image captured by the camera.
• Output: When the entire procedure is completed.
The characteristics of all students are stored in a database.

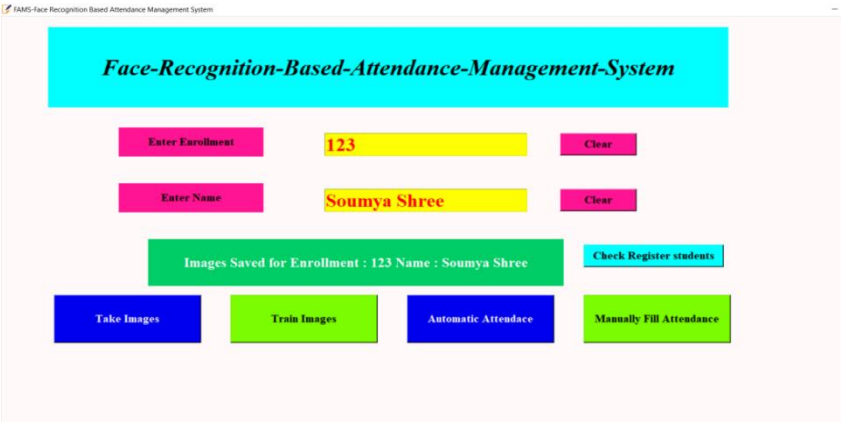


Figure 9: Database Creation

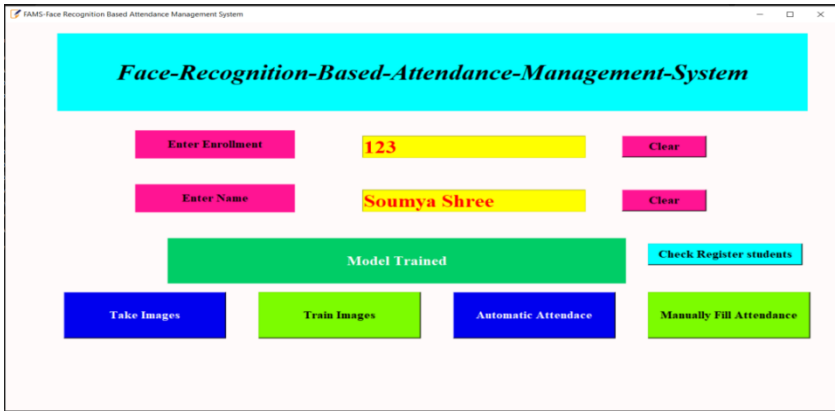


Figure 10: Train Database

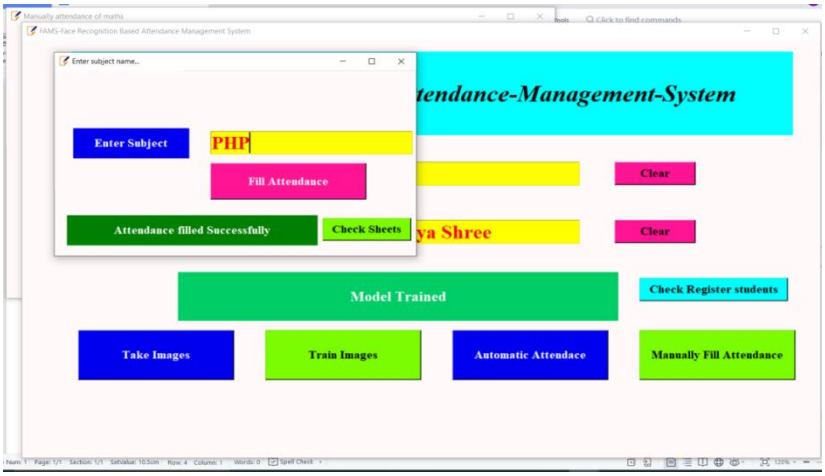


Figure 11: Attendance marking

Figure 12: Store Train Images

2. Detecting faces:

• Input: This is the input frame.

• Output: From a bunch of photos, find the Face and divide the images.

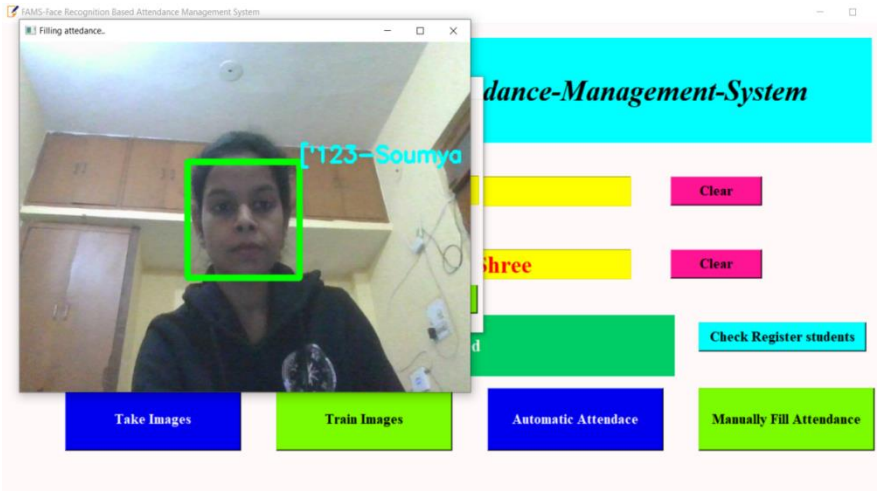


Figure 6: Face Detection

3. Extraction of the Face:
   - Input: The face detection module's result
   - Output: Extract the pupils' identified images that match the database.

4.Excel sheet Generation:
   - Input: An extracted image as an individual.
   - Output: Attendance mark for present or absent and store data and Generate report.

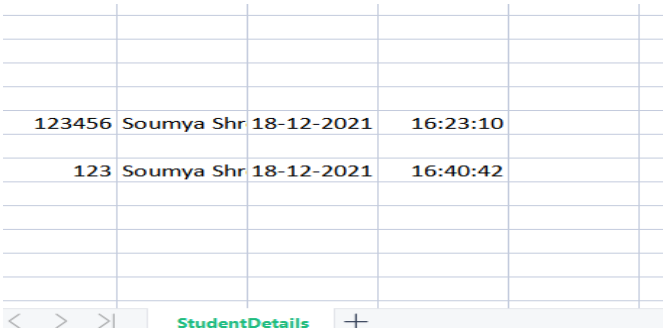


Figure 7: Report Generation

CHAPTER-5

CODING

AMS_Run.py

```
import tkinter as tk
from tkinter import *
import cv2
import csv
import os
import numpy as np
from PIL import Image,ImageTk
import pandas as pd
import datetime
import time

window = tk.Tk()
window.title("FAMS-Face Recognition Based Attendance Management System")

window.geometry('1280x720')
window.configure(background='snow')

####GUI for manually fill attendance

def manually_fill():
    global sb
    sb = tk.Tk()
    sb.iconbitmap('AMS.ico')
    sb.title("Enter subject name...")
    sb.geometry('580x320')
    sb.configure(background='snow')

    def err_screen_for_subject():

        def ec_delete():
            ec.destroy()
        global ec
        ec = tk.Tk()
        ec.geometry('300x100')
        ec.iconbitmap('AMS.ico')
        ec.title('Warning!!')
        ec.configure(background='snow')
        Label(ec, text='Please enter your subject name!!!', fg='red', bg='white', font=('times', 16, ' bold
')).pack()
        Button(ec, text='OK', command=ec_delete, fg="black", bg="lawn green", width=9, height=1,
activebackground="Red",
               font=('times', 15, ' bold ')).place(x=90, y=50)
```

```
    def fill_attendance():
        ts = time.time()
        Date = datetime.datetime.fromtimestamp(ts).strftime('%Y_%m_%d')
        timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
        Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
        Hour, Minute, Second = timeStamp.split(":")
        ####Creatting csv of attendance

        ##Create table for Attendance
        date_for_DB = datetime.datetime.fromtimestamp(ts).strftime('%Y_%m_%d')
        global subb
        subb=SUB_ENTRY.get()
        DB_table_name = str(subb + "_" + Date + "_Time_" + Hour + "_" + Minute + "_" + Second)

        import pymysql.connections

        ###Connect to the database
        try:
            global cursor
            connection = pymysql.connect(host='localhost', user='root', password='',
db='manually_fill_attendance')
            cursor = connection.cursor()
        except Exception as e:
            print(e)

        sql = "CREATE TABLE " + DB_table_name + """
                    (ID INT NOT NULL AUTO_INCREMENT,
                     ENROLLMENT varchar(100) NOT NULL,
                     NAME VARCHAR(50) NOT NULL,
                     DATE VARCHAR(20) NOT NULL,
                     TIME VARCHAR(20) NOT NULL,
                        PRIMARY KEY (ID)
                        );
                    """



        try:
            cursor.execute(sql)  ##for create a table
        except Exception as ex:
            print(ex)  #

        if subb=='':
            err_screen_for_subject()
        else:
            sb.destroy()
            MFW = tk.Tk()
            MFW.iconbitmap('AMS.ico')
            MFW.title("Manually attendance of "+ str(subb))
```

```
        MFW.geometry('880x470')
        MFW.configure(background='snow')

        def del_errsc2():
            errsc2.destroy()

        def err_screen1():
            global errsc2
            errsc2 = tk.Tk()
            errsc2.geometry('330x100')
            errsc2.iconbitmap('AMS.ico')
            errsc2.title('Warning!!')
            errsc2.configure(background='snow')
            Label(errsc2, text='Please enter Student & Enrollment!!!', fg='red', bg='white',
                  font=('times', 16, ' bold ')).pack()
            Button(errsc2, text='OK', command=del_errsc2, fg="black", bg="lawn green", width=9,
height=1,
                   activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)

        def testVal(inStr, acttyp):
            if acttyp == '1':  # insert
                if not inStr.isdigit():
                    return False
            return True

        ENR = tk.Label(MFW, text="Enter Enrollment", width=15, height=2, fg="white",
bg="blue2",
                       font=('times', 15, ' bold '))
        ENR.place(x=30, y=100)

        STU_NAME = tk.Label(MFW, text="Enter Student name", width=15, height=2,
fg="white", bg="blue2",
                            font=('times', 15, ' bold '))
        STU_NAME.place(x=30, y=200)

        global ENR_ENTRY
        ENR_ENTRY = tk.Entry(MFW, width=20,validate='key', bg="yellow", fg="red",
font=('times', 23, ' bold '))
        ENR_ENTRY['validatecommand'] = (ENR_ENTRY.register(testVal), '%P', '%d')
        ENR_ENTRY.place(x=290, y=105)

        def remove_enr():
            ENR_ENTRY.delete(first=0, last=22)

        STUDENT_ENTRY = tk.Entry(MFW, width=20, bg="yellow", fg="red", font=('times', 23,
' bold '))
        STUDENT_ENTRY.place(x=290, y=205)

        def remove_student():
```

```
            STUDENT_ENTRY.delete(first=0, last=22)

        ####get important variable
        def enter_data_DB():
            ENROLLMENT = ENR_ENTRY.get()
            STUDENT = STUDENT_ENTRY.get()
            if ENROLLMENT==='':
                err_screen1()
            elif STUDENT==='':
                err_screen1()
            else:
                time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                Hour, Minute, Second = time.split(":")
                Insert_data = "INSERT INTO " + DB_table_name + "
(ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"
                VALUES = (str(ENROLLMENT), str(STUDENT), str(Date), str(time))
                try:
                    cursor.execute(Insert_data, VALUES)
                except Exception as e:
                    print(e)
                ENR_ENTRY.delete(first=0, last=22)
                STUDENT_ENTRY.delete(first=0, last=22)

        def create_csv():
            import csv
            cursor.execute("select * from " + DB_table_name + ";")
            csv_name='C:/Users/kusha/PycharmProjects/Attendace managemnt
system/Attendance/Manually Attendance/'+DB_table_name+'.csv'
            with open(csv_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file)
                csv_writer.writerow([i[0] for i in cursor.description])  # write headers
                csv_writer.writerows(cursor)
                O="CSV created Successfully"
                Notifi.configure(text=O, bg="Green", fg="white", width=33, font=('times', 19, 'bold'))
                Notifi.place(x=180, y=380)
            import csv
            import tkinter
            root = tkinter.Tk()
            root.title("Attendance of " + subb)
            root.configure(background='snow')
            with open(csv_name, newline="") as file:
                reader = csv.reader(file)
                r = 0

                for col in reader:
                    c = 0
                    for row in col:
                        # i've added some styling
                        label = tkinter.Label(root, width=13, height=1, fg="black", font=('times', 13, '
```

```
bold '),
                                    bg="lawn green", text=row, relief=tkinter.RIDGE)
                label.grid(row=r, column=c)
                c += 1
            r += 1
        root.mainloop()

    Notifi = tk.Label(MFW, text="CSV created Successfully", bg="Green", fg="white",
width=33,
                      height=2, font=('times', 19, 'bold'))


    c1ear_enroll = tk.Button(MFW, text="Clear", command=remove_enr, fg="black",
bg="deep pink", width=10,
                         height=1,
                         activebackground="Red", font=('times', 15, ' bold '))
    c1ear_enroll.place(x=690, y=100)

    c1ear_student = tk.Button(MFW, text="Clear", command=remove_student, fg="black",
bg="deep pink", width=10,
                          height=1,
                          activebackground="Red", font=('times', 15, ' bold '))
    c1ear_student.place(x=690, y=200)

    DATA_SUB = tk.Button(MFW, text="Enter Data",command=enter_data_DB, fg="black",
bg="lime green", width=20,
                     height=2,
                     activebackground="Red", font=('times', 15, ' bold '))
    DATA_SUB.place(x=170, y=300)

    MAKE_CSV = tk.Button(MFW, text="Convert to CSV",command=create_csv, fg="black",
bg="red", width=20,
                     height=2,
                     activebackground="Red", font=('times', 15, ' bold '))
    MAKE_CSV.place(x=570, y=300)

    def attf():
        import subprocess
        subprocess.Popen(r'explorer /select,"C:\Users\Soumya Shree\Downloads\Project
material\Attendace_management_system\Attendance\Manually Attendance\-------Check
atttendance-------"')

    attf = tk.Button(MFW,  text="Check Sheets",command=attf,fg="black"  ,bg="lawn green"
,width=12  ,height=1 ,activebackground = "Red" ,font=('times', 14, ' bold '))
    attf.place(x=730, y=410)

    MFW.mainloop()
```

```
    SUB = tk.Label(sb, text="Enter Subject", width=15, height=2, fg="white", bg="blue2",
font=('times', 15, ' bold '))
    SUB.place(x=30, y=100)

    global SUB_ENTRY

    SUB_ENTRY = tk.Entry(sb, width=20, bg="yellow", fg="red", font=('times', 23, ' bold '))
    SUB_ENTRY.place(x=250, y=105)

    fill_manual_attendance = tk.Button(sb, text="Fill Attendance",command=fill_attendance,
fg="white", bg="deep pink", width=20, height=2,
                       activebackground="Red", font=('times', 15, ' bold '))
    fill_manual_attendance.place(x=250, y=160)
    sb.mainloop()

##For clear textbox
def clear():
    txt.delete(first=0, last=22)

def clear1():
    txt2.delete(first=0, last=22)
def del_sc1():
    sc1.destroy()
def err_screen():
    global sc1
    sc1 = tk.Tk()
    sc1.geometry('300x100')
    sc1.iconbitmap('AMS.ico')
    sc1.title('Warning!!')
    sc1.configure(background='snow')
    Label(sc1,text='Enrollment & Name required!!!',fg='red',bg='white',font=('times', 16, ' bold
')).pack()
    Button(sc1,text='OK',command=del_sc1,fg="black"  ,bg="lawn green"  ,width=9  ,height=1,
activebackground = "Red" ,font=('times', 15, ' bold ')).place(x=90,y= 50)

##Error screen2
def del_sc2():
    sc2.destroy()
def err_screen1():
    global sc2
    sc2 = tk.Tk()
    sc2.geometry('300x100')
    sc2.iconbitmap('AMS.ico')
    sc2.title('Warning!!')
    sc2.configure(background='snow')
    Label(sc2,text='Please enter your subject name!!!',fg='red',bg='white',font=('times', 16, ' bold
')).pack()
    Button(sc2,text='OK',command=del_sc2,fg="black"  ,bg="lawn green"  ,width=9  ,height=1,
activebackground = "Red" ,font=('times', 15, ' bold ')).place(x=90,y= 50)
```

```
###For take images for datasets
def take_img():
    l1 = txt.get()
    l2 = txt2.get()
    if l1 == '':
        err_screen()
    elif l2 == '':
        err_screen()
    else:
        try:
            cam = cv2.VideoCapture(0)
            detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
            Enrollment = txt.get()
            Name = txt2.get()
            sampleNum = 0
            while (True):
                ret, img = cam.read()
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = detector.detectMultiScale(gray, 1.3, 5)
                for (x, y, w, h) in faces:
                    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                    # incrementing sample number
                    sampleNum = sampleNum + 1
                    # saving the captured face in the dataset folder
                    cv2.imwrite("TrainingImage/ " + Name + "." + Enrollment + '.' + str(sampleNum) +
".jpg",
                                gray[y:y + h, x:x + w])
                    cv2.imshow('Frame', img)
                # wait for 100 miliseconds
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
                # break if the sample number is morethan 100
                elif sampleNum > 70:
                    break
            cam.release()
            cv2.destroyAllWindows()
            ts = time.time()
            Date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
            Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
            row = [Enrollment, Name, Date, Time]
            with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:
                writer = csv.writer(csvFile, delimiter=',')
                writer.writerow(row)
                csvFile.close()
            res = "Images Saved for Enrollment : " + Enrollment + " Name : " + Name
            Notification.configure(text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))
            Notification.place(x=250, y=400)
        except FileExistsError as F:
```

```
            f = 'Student Data already exists'
            Notification.configure(text=f, bg="Red", width=21)
            Notification.place(x=450, y=400)


###for choose subject and fill attendance
def subjectchoose():
    def Fillattendances():
        sub=tx.get()
        now = time.time()  ###For calculate seconds of video
        future = now + 20
        if time.time() < future:
            if sub == '':
                err_screen1()
            else:
                recognizer = cv2.face.LBPHFaceRecognizer_create()  #
cv2.createLBPHFaceRecognizer()
                try:
                    recognizer.read("TrainingImageLabel\Trainner.yml")
                except:
                    e = 'Model not found,Please train model'
                    Notifica.configure(text=e, bg="red", fg="black", width=33, font=('times', 15, 'bold'))
                    Notifica.place(x=20, y=250)

                harcascadePath = "haarcascade_frontalface_default.xml"
                faceCascade = cv2.CascadeClassifier(harcascadePath)
                df = pd.read_csv("StudentDetails\StudentDetails.csv")
                cam = cv2.VideoCapture(0)
                font = cv2.FONT_HERSHEY_SIMPLEX
                col_names = ['Enrollment', 'Name', 'Date', 'Time']
                attendance = pd.DataFrame(columns=col_names)
                while True:
                    ret, im = cam.read()
                    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
                    faces = faceCascade.detectMultiScale(gray, 1.2, 5)
                    for (x, y, w, h) in faces:
                        global Id

                        Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
                        if (conf <70):
                            print(conf)
                            global Subject
                            global aa
                            global date
                            global timeStamp
                            Subject = tx.get()
                            ts = time.time()
                            date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
                            timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
```

```
                    aa = df.loc[df['Enrollment'] == Id]['Name'].values
                    global tt
                    tt = str(Id) + "-" + aa
                    En = '15624031' + str(Id)
                    attendance.loc[len(attendance)] = [Id, aa, date, timeStamp]
                    cv2.rectangle(im, (x, y), (x + w, y + h), (0, 260, 0), 7)
                    cv2.putText(im, str(tt), (x + h, y), font, 1, (255, 255, 0,), 4)

                else:
                    Id = 'Unknown'
                    tt = str(Id)
                    cv2.rectangle(im, (x, y), (x + w, y + h), (0, 25, 255), 7)
                    cv2.putText(im, str(tt), (x + h, y), font, 1, (0, 25, 255), 4)
            if time.time() > future:
                break

            attendance = attendance.drop_duplicates(['Enrollment'], keep='first')
            cv2.imshow('Filling attedance..', im)
            key = cv2.waitKey(30) & 0xff
            if key == 27:
                break

        ts = time.time()
        date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
        timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
        Hour, Minute, Second = timeStamp.split(":")
        fileName = "Attendance/" + Subject + "_" + date + "_" + Hour + "-" + Minute + "-" +
Second + ".csv"
        attendance = attendance.drop_duplicates(['Enrollment'], keep='first')
        print(attendance)
        attendance.to_csv(fileName, index=False)

        ##Create table for Attendance
        date_for_DB = datetime.datetime.fromtimestamp(ts).strftime('%Y_%m_%d')
        DB_Table_name = str( Subject + "_" + date_for_DB + "_Time_" + Hour + "_" + Minute
+ "_" + Second)
        import pymysql.connections

        ###Connect to the database
        try:
            global cursor
            connection = pymysql.connect(host='localhost', user='root', password='',
db='Face_reco_fill')
            cursor = connection.cursor()
        except Exception as e:
            print(e)

        sql = "CREATE TABLE " + DB_Table_name + """
        (ID INT NOT NULL AUTO_INCREMENT,
```

```
            ENROLLMENT varchar(100) NOT NULL,
            NAME VARCHAR(50) NOT NULL,
            DATE VARCHAR(20) NOT NULL,
            TIME VARCHAR(20) NOT NULL,
               PRIMARY KEY (ID)
               );
            """
            ####Now enter attendance in Database
            insert_data =  "INSERT INTO " + DB_Table_name + "
(ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"
            VALUES = (str(Id), str(aa), str(date), str(timeStamp))
            try:
               cursor.execute(sql)  ##for create a table
               cursor.execute(insert_data, VALUES)##For insert data into table
            except Exception as ex:
               print(ex)  #

            M = 'Attendance filled Successfully'
            Notifica.configure(text=M, bg="Green", fg="white", width=33, font=('times', 15, 'bold'))
            Notifica.place(x=20, y=250)

            cam.release()
            cv2.destroyAllWindows()

            import csv
            import tkinter
            root = tkinter.Tk()
            root.title("Attendance of " + Subject)
            root.configure(background='snow')
            cs = 'C:\Users\Soumya Shree\Downloads\Project
            material\Attendace_management_system/' + fileName
            with open(cs, newline="") as file:
               reader = csv.reader(file)
               r = 0

               for col in reader:
                  c = 0
                  for row in col:
                     # i've added some styling
                     label = tkinter.Label(root, width=8, height=1, fg="black", font=('times', 15, ' bold
'),
                                           bg="lawn green", text=row, relief=tkinter.RIDGE)
                     label.grid(row=r, column=c)
                     c += 1
                  r += 1
            root.mainloop()
            print(attendance)

      ###windo is frame for subject chooser
```

```
    windo = tk.Tk()
    windo.iconbitmap('AMS.ico')
    windo.title("Enter subject name...")
    windo.geometry('580x320')
    windo.configure(background='snow')
    Notifica = tk.Label(windo, text="Attendance filled Successfully", bg="Green", fg="white",
width=33,
                        height=2, font=('times', 15, 'bold'))

    def Attf():
        import subprocess
        subprocess.Popen(r'explorer /select,"C:\Users\Soumya Shree\Downloads\Project
        material\Attendace_management_system\Attendance\-------Check atttendance-------"')

    attf = tk.Button(windo,  text="Check Sheets",command=Attf,fg="black"  ,bg="lawn green"
,width=12  ,height=1 ,activebackground = "Red" ,font=('times', 14, ' bold '))
    attf.place(x=430, y=255)

    sub = tk.Label(windo, text="Enter Subject", width=15, height=2, fg="white", bg="blue2",
font=('times', 15, ' bold '))
    sub.place(x=30, y=100)

    tx = tk.Entry(windo, width=20, bg="yellow", fg="red", font=('times', 23, ' bold '))
    tx.place(x=250, y=105)

    fill_a = tk.Button(windo, text="Fill Attendance", fg="white",command=Fillattendances,
bg="deep pink", width=20, height=2,
                       activebackground="Red", font=('times', 15, ' bold '))
    fill_a.place(x=250, y=160)
    windo.mainloop()

def admin_panel():
    win = tk.Tk()
    win.iconbitmap('AMS.ico')
    win.title("LogIn")
    win.geometry('880x420')
    win.configure(background='snow')

    def log_in():
        username = un_entr.get()
        password = pw_entr.get()

        if username == 'soumya9525' :
            if password == 'Soumya123@':
                win.destroy()
                import csv
                import tkinter
                root = tkinter.Tk()
                root.title("Student Details")
```

```
            root.configure(background='snow')

            cs = 'C:\Users\Soumya Shree\Downloads\Project
material\Attendace_management_system\StudentDetails\StudentDetails.csv'
            with open(cs, newline="") as file:
                reader = csv.reader(file)
                r = 0

                for col in reader:
                    c = 0
                    for row in col:
                        # i've added some styling
                        label = tkinter.Label(root, width=8, height=1, fg="black", font=('times', 15, ' bold
'),
                                              bg="lawn green", text=row, relief=tkinter.RIDGE)
                        label.grid(row=r, column=c)
                        c += 1
                    r += 1
                root.mainloop()
            else:
                valid = 'Incorrect ID or Password'
                Nt.configure(text=valid, bg="red", fg="black", width=38, font=('times', 19, 'bold'))
                Nt.place(x=120, y=350)

        else:
            valid ='Incorrect ID or Password'
            Nt.configure(text=valid, bg="red", fg="black", width=38, font=('times', 19, 'bold'))
            Nt.place(x=120, y=350)


    Nt = tk.Label(win, text="Attendance filled Successfully", bg="Green", fg="white", width=40,
                  height=2, font=('times', 19, 'bold'))
    # Nt.place(x=120, y=350)

    un = tk.Label(win, text="Enter username", width=15, height=2, fg="white", bg="blue2",
                  font=('times', 15, ' bold '))
    un.place(x=30, y=50)

    pw = tk.Label(win, text="Enter password", width=15, height=2, fg="white", bg="blue2",
                  font=('times', 15, ' bold '))
    pw.place(x=30, y=150)

    def c00():
        un_entr.delete(first=0, last=22)

    un_entr = tk.Entry(win, width=20, bg="yellow", fg="red", font=('times', 23, ' bold '))
    un_entr.place(x=290, y=55)

    def c11():
```

```
        pw_entr.delete(first=0, last=22)

    pw_entr = tk.Entry(win, width=20,show="*", bg="yellow", fg="red", font=('times', 23, ' bold '))
    pw_entr.place(x=290, y=155)

    c0 = tk.Button(win, text="Clear", command=c00, fg="black", bg="deep pink", width=10,
height=1,
                       activebackground="Red", font=('times', 15, ' bold '))
    c0.place(x=690, y=55)

    c1 = tk.Button(win, text="Clear", command=c11, fg="black", bg="deep pink", width=10,
height=1,
                activebackground="Red", font=('times', 15, ' bold '))
    c1.place(x=690, y=155)

    Login = tk.Button(win, text="LogIn", fg="black", bg="lime green", width=20,
                      height=2,
                      activebackground="Red",command=log_in, font=('times', 15, ' bold '))
    Login.place(x=290, y=250)
    win.mainloop()


###For train the model
def trainimg():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    global detector
    detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    try:
        global faces,Id
        faces, Id = getImagesAndLabels("TrainingImage")
    except Exception as e:
        l='please make "TrainingImage" folder & put Images'
        Notification.configure(text=l, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))
        Notification.place(x=350, y=400)

    recognizer.train(faces, np.array(Id))
    try:
        recognizer.save("TrainingImageLabel\Trainner.yml")
    except Exception as e:
        q='Please make "TrainingImageLabel" folder'
        Notification.configure(text=q, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))
        Notification.place(x=350, y=400)

    res = "Model Trained"  # +",".join(str(f) for f in Id)
    Notification.configure(text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))
    Notification.place(x=250, y=400)

def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
```

```
    # create empth face list
    faceSamples = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image

        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces = detector.detectMultiScale(imageNp)
        # If a face is there then append that in the list as well as Id of it
        for (x, y, w, h) in faces:
            faceSamples.append(imageNp[y:y + h, x:x + w])
            Ids.append(Id)
    return faceSamples, Ids

window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)
window.iconbitmap('AMS.ico')

def on_closing():
    from tkinter import messagebox
    if messagebox.askokcancel("Quit", "Do you want to quit?"):
        window.destroy()
window.protocol("WM_DELETE_WINDOW", on_closing)

message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System",
bg="cyan", fg="black", width=50,
                   height=3, font=('times', 30, 'italic bold '))

message.place(x=80, y=20)

Notification = tk.Label(window, text="All things good", bg="Green", fg="white", width=15,
                        height=3, font=('times', 17, 'bold'))

lbl = tk.Label(window, text="Enter Enrollment", width=20, height=2, fg="black", bg="deep pink",
font=('times', 15, ' bold '))
lbl.place(x=200, y=200)

def testVal(inStr,acttyp):
    if acttyp == '1': #insert
        if not inStr.isdigit():
            return False
    return True
```

```
txt = tk.Entry(window, validate="key", width=20, bg="yellow", fg="red", font=('times', 25, ' bold
'))
txt['validatecommand'] = (txt.register(testVal),'%P','%d')
txt.place(x=550, y=210)

lbl2 = tk.Label(window, text="Enter Name", width=20, fg="black", bg="deep pink", height=2,
font=('times', 15, ' bold '))
lbl2.place(x=200, y=300)

txt2 = tk.Entry(window, width=20, bg="yellow", fg="red", font=('times', 25, ' bold '))
txt2.place(x=550, y=310)

clearButton = tk.Button(window, text="Clear",command=clear,fg="black"  ,bg="deep pink"
,width=10  ,height=1 ,activebackground = "Red" ,font=('times', 15, ' bold '))
clearButton.place(x=950, y=210)

clearButton1 = tk.Button(window, text="Clear",command=clear1,fg="black"  ,bg="deep pink"
,width=10 ,height=1, activebackground = "Red" ,font=('times', 15, ' bold '))
clearButton1.place(x=950, y=310)

AP = tk.Button(window, text="Check Register students",command=admin_panel,fg="black"
,bg="cyan"  ,width=19 ,height=1, activebackground = "Red" ,font=('times', 15, ' bold '))
AP.place(x=990, y=410)

takeImg = tk.Button(window, text="Take Images",command=take_img,fg="white"  ,bg="blue2"
,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
takeImg.place(x=90, y=500)

trainImg = tk.Button(window, text="Train Images",fg="black",command=trainimg ,bg="lawn
green"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
trainImg.place(x=390, y=500)

FA = tk.Button(window, text="Automatic Attendace",fg="white",command=subjectchoose
,bg="blue2"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
FA.place(x=690, y=500)

quitWindow = tk.Button(window, text="Manually Fill Attendance", command=manually_fill
,fg="black"  ,bg="lawn green"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, '
bold '))
quitWindow.place(x=990, y=500)

window.mainloop()
```

testing.py

```
import cv2
import numpy as np

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('TrainingImageLabel/trainner.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
font = cv2.FONT_HERSHEY_SIMPLEX

cam = cv2.VideoCapture(0)
while True:
    ret, im =cam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=faceCascade.detectMultiScale(gray, 1.2,5)
    for(x,y,w,h) in faces:
        Id, conf = recognizer.predict(gray[y:y+h,x:x+w])

        # # else:
        # #     Id="Unknown"
        # cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 260, 0), 7)
        cv2.putText(im, str(Id), (x,y-40),font, 2, (255,255,255), 3)

        # cv2.putText(im, str(Id), (x + h, y), font, 1, (0, 260, 0), 2)
    cv2.imshow('im',im)
    if cv2.waitKey(10) & 0xFF==ord('q'):
        break
cam.release()
cv2.destroyAllWindows()
```

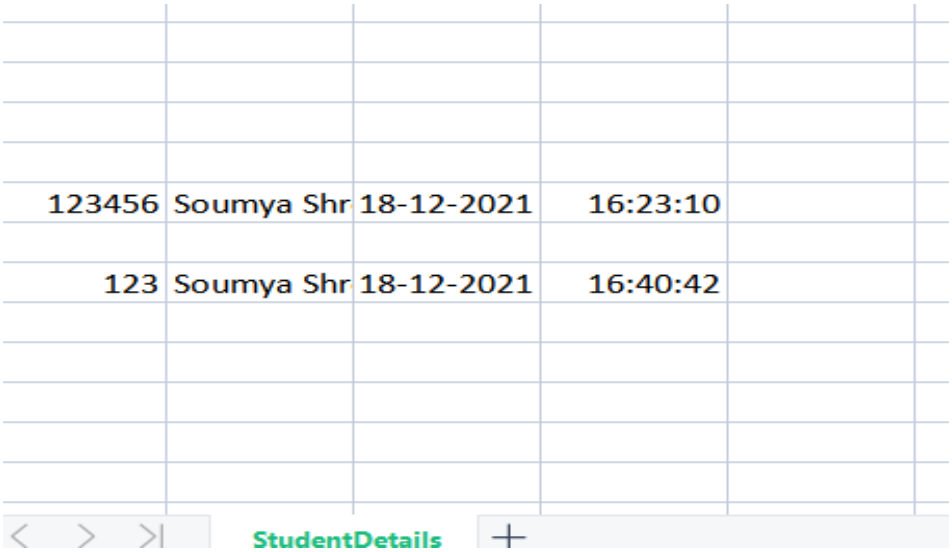
training.py

```
import cv2,os
import numpy as np
from PIL import Image
#
# recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer=cv2.face.createFisherFaceRecognizer_create()
detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #create empth face list
    faceSamples=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image

        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces=detector.detectMultiScale(imageNp)
        #If a face is there then append that in the list as well as Id of it
        for (x,y,w,h) in faces:
            faceSamples.append(imageNp[y:y+h,x:x+w])
            Ids.append(Id)
    return faceSamples,Ids

faces,Ids = getImagesAndLabels('TrainingImage')
recognizer.train(faces, np.array(Ids))
recognizer.save('TrainingImageLabel/trainner.yml')
```

CHAPTER-6

RESULTS

The existing system for managing attendance is simple and effective. After 20 minutes of lecture, the camera records the photos. After receiving and acknowledging students, attendance is recorded in the database, and a report is generated for each student. The Automated Face Recognition Attendance System aids in the improvement of accuracy and speed, resulting in high-precision real-time attendance to fulfil the demand for automatic classroom evaluation. This technique is intended to reduce the amount of time and effort required by humans to manually take attendance at each college. The amount of precision is determined by the image quality captured by the camera. It is necessary to have appropriate illumination in order to see the face clearly. When using a high-resolution camera, performance is likely to improve.



CHAPTER-7

Conclusion and Future Scope

The goal of our project is to capture images of students, convert them into frames, connect them to a database to confirm their presence or absence, and record attendance for each student. Because the attendance marking procedure is done without human intervention, which is the system's main scope. Teachers benefit from automated attendance systems in terms of saving time and lowering workload. However, this technique could be improved by displaying the number of pupils present on the class display board right away. As a result, the teacher will be able to recognize the number of missing students in order to rectify the situation.

Furthermore, by utilizing multiple facial detections to indicate attendance of all visible faces in a single attempt, we may simplify and improve the system's efficiency. Face recognition will be used for attendance marking in a more cost-effective and efficient manner. In the near future, we may potentially develop an Android application for this system.

REFRENCES

[1] V. Shehu and A. Dika, “Using real time computer vision algorithms in automatic attendance management systems,” Inf. Technol. Interfaces (ITI), 2010 32nd Int. Conf., pp. 397–402, 2010.
[2] A. S. S. NAVAZ and T. D. S. P. MAZUMDER, “Face Recognition using Principal Component Analysis and Neural Networks,” vol. 1, no. April, pp. 91–94, 2001.
[3] Grundland M, Dodgson N (2007) Decolorize: Fast, contrast enhancing, color to grayscale conversion. Pattern Recognition 40:
2891-2896.
[4] F. Ibikunle, Agbetuvi F. and Ukpere G. “Face Recognition Using Line Edge Mapping Approach.” American Journal of
Electrical and Electronic Engineering 1.3(2013): 52-59
[5] T. Kanade, Computer Recognition of Human Faces. Basel and Stuttgart: Birkhauser Verlag 1997.
[6] Suman Kumar Bhattacharyya & Kumar Rahul. (2013), “Face Recognition by Linear Discriminant Analysis”, International Journal of Communication Network Security, V2(2), pp 31-35. (LDA)
[7] Varsha Gupta, Dipesh Sharma. (2014), “A Study of Various Face Detection Methods”, International Journal of Advanced Research in Computer and Communication Engineering), vol.3, no. 5.
[8] Viola, M. J. Jones. (2004), “Robust Real-Time Face Detection”, International Journal of Computer Vision 57(2), 137– 154
[9] Pratiksha M. Patel (2016). Contrast Enhancement of Images and videos using Histogram Equalization. International Journal on Recent and Innovation Trends in Computing and Communication.V4 (11).
[10] Kasar, M., Bhattacharyya, D. and Kim, T. (2016). Face Recognition Using Neural Network: A Review. International Journal of Security and Its Applications, 10(3), pp.81-100