An
Industry Oriented Mini Project Report
On

Face Recognition Attendance System

Submitted in partial fulfillment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

K.NITHIN	227Z1A0576
L.BINDU	227Z1A0591
K.PRAVEEN	227Z1A0582

Under the Guidance of
Mr. S. Srikanth Reddy
Assisstant Professor

SCHOOL OF ENGINEERING
Department of Computer Science and Engineering

NALLA NARASIMHA REDDY
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS
(AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi, Chowdariguda (V) Korremula 'x'Roads, via Narapally, Ghatkesar (Mandal) Medchal (Dist), Telangana-500088

2024-2025

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report titled **"FACE RECOGNITION ATTENDANCE SYSTEM"** is being submitted by **K. Nithin (227Z1A0576), L.Bindu (227Z1A0591) and K. Praveen (227Z1A0582)** in Partial fulfillment for the award of **Bachelor of technology in Computer Science & Engineering** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
(Mr. S. Srikanth Reddy)

Head of the Department
(Dr.K.Rameshwaraiah)

Submitted for the University Examination held on…….

External Examiner

DECLARATION

We K. Nithin, L. Bindu and K. Praveen the students of **Bachelor of Technology in Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **FACE RECOGNITION ATTENDANCE SYSTEM** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

K. Nithin	227Z1A0576
L. Bindu	227Z1A0591
K. Praveen	227Z1A0582

Date:

Signature:

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mr. S. Srikanth Reddy,** Assistant Professor, Department of Computer Science and Engineering, NNRESGI, who motivated throughout the period of the project and also for his valuable and intellectual suggestions, guidance, and constant encouragement right throughout our work.

We would like to express our profound gratitude to our project coordinator **Mr. Durga Prasad**, Assistant Professor, Department of Computer Science and Engineering, NNRESGI, for his support and guidance in completing our project and for giving us this opportunity to present the project work.

We profoundly express thanks to **Dr. K. Rameshwaraiah,** Professor & Head, Department of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju,** Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr**. **C. V. Krishna Reddy,** Director, NNRESGI, for providing the facilities for completion of the project.

Finally, we would like to thank Project Review Committee (PRC) members, all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By

K. Nithin 227Z1A0576

L. Bindu 227Z1A0591

K. Praveen 227Z1A0582

ABSTRACT

The "Face Recognition Attendance System using Python" provides a contactless and automated solution for attendance management in workplaces, schools, and events. Developed using Python, the system incorporates OpenCV for real-time image processing, Face Recognition for precise facial detection, and NumPy for computational tasks. It features real-time face detection and recognition through camera feeds using pre-trained models, automatic attendance logging with timestamps stored in secure databases, and an easy registration process for capturing facial data and personal details. Additional functionalities include real-time notifications for attendance confirmation or alerts for unregistered users, detailed attendance reports and trend analytics through a user-friendly interface, and scalability to accommodate large populations while integrating with management systems like payroll or LMS. By addressing challenges such as impersonation, buddy punching, and manual errors, this system enhances efficiency and accuracy. Future developments may include multi-camera support, mobile integration, and advanced features such as emotion detection and mask recognition, making it a cost-effective and versatile solution for various applications.

Keywords:

Image processing, Face detection, Face recognition, Automated attendance, Secured database.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S. No.	Abbreviation	Definitions
1	LBPH	Local Binary Pattern Histogram
2	GUI	Graphical User Interface
3	ROI	Region of Interest
4	ID	Identification
5	CSV	Comma-Separated Values
6	AI	Artificial Intelligence
7	OpenCV	Open Source Computer Vision Library

1. INTRODUCTION

An automated attendance system using face recognition can streamline daily operations, save valuable time, and ensure accurate record-keeping. Such a system enhances productivity by minimizing administrative overhead and allows institutions to focus more on core activities like teaching and learning. Moreover, it reinforces security by verifying identities, thus preventing unauthorized access or impersonation.

1.1Motivation

The motivation behind this project stems from the inefficiencies and limitations of traditional attendance systems commonly used in educational institutions and organizations. Manual attendance-taking methods are often time-consuming, tedious, and prone to human error. Furthermore, these systems are susceptible to proxy attendance, where one individual marks the presence of another, compromising the accuracy and integrity of attendance records.

With the rapid advancements in technology, particularly in the fields of computer vision, machine learning, and facial recognition, there is a significant opportunity to modernize and enhance the attendance process. Face recognition technology provides a non-intrusive, reliable, and automated solution that can identify and verify individuals in real-time, ensuring authenticity and reducing the likelihood of fraud.

1.2 Problem Definition

The traditional manual attendance system is outdated and lacks the automation needed to meet the demands of modern institutions. It is not only time-consuming but also vulnerable to various forms of manipulation. One of the major drawbacks is the significant time wasted during roll calls, especially in large classrooms or organizations. This reduces the time available for actual teaching or productive work.

In addition, tracking and maintaining attendance records manually is cumbersome and prone to human error. Over time, these errors can accumulate, leading to inaccurate records and administrative confusion. The manual system is also highly susceptible to proxy attendance and fraud, where individuals mark attendance on behalf of others. This

undermines the reliability of the entire process and can lead to false reporting of participation and engagement.

1.3 Objective of Project

The primary objectives of the project are as below:

1.3.1 To develop a GUI-based face recognition attendance system

- A graphical user interface (GUI) ensures the system is easy to use for non-technical users such as teachers and administrators.
- The GUI will include buttons and forms for adding students, taking attendance, and managing records, enhancing user interaction**.**

1.3.2 To use OpenCV and machine learning for face detection and recognition

- OpenCV will handle real-time image capture and facial feature extraction for accurate detection.
- Machine learning algorithms like LBPH will be used to recognize faces and match them to the stored database for verification**.**

1.3.3 To automate attendance entry and storage in a CSV format

- The system will automatically log attendance once a face is recognized, reducing the need for manual data entry.
- Attendance data will be stored in CSV format, allowing for easy import, export, and analysis in spreadsheet software**.**

1.3.4 To allow for deletion and viewing of student records and attendance

- Users can delete outdated or incorrect student records to maintain a clean and accurate database.
- Administrators can view attendance reports and student details directly from the interface for easy monitoring and auditing**.**

1.3.5 To provide an efficient and secure solution for real-time attendance monitoring

- Real-time face recognition ensures that attendance is marked instantly and accurately as students arrive.
- Security features like restricted access to admin controls help prevent unauthorized data modification or misuse.

1.4 Limitations of Project

1. Low Light Performance Issues

Face recognition accuracy may drop in poorly lit environments, making detection and identification unreliable.

Proper lighting is essential for consistent and clear image capture.

2. Training Data Quality Dependency

The system's accuracy heavily relies on high-quality and diverse training images of each individual.

Insufficient or low-resolution images can lead to misidentification or failed recognition.

3. Impact of Facial Obstructions

Wearing masks, sunglasses, or hats can obstruct facial features and hinder recognition.

The system may struggle to match faces that are partially hidden or distorted.

4. Hardware Requirements

A webcam and sufficient processing power are necessary for smooth, real-time face detection and recognition.

Low-spec devices may experience lag or reduced performance.

5. Scalability Challenges

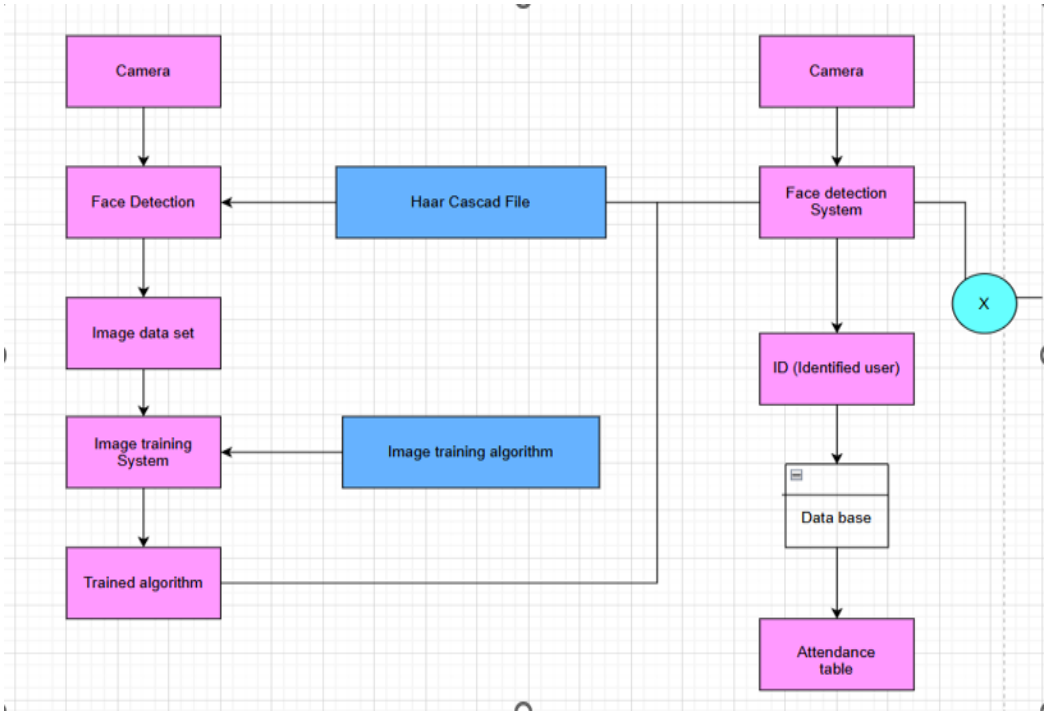
As the number of users increases, recognition time and data management become more complex.

Without proper optimization, the system may slow down with large datasets.

2. LITERATURE SURVEY

S.No	Paper Title	Authors	Year	Methodology	Findings	Limitations
1	Automated Attendance System Using Face Recognition	A. Sharma, B. Patel	2020	CNN-based facial recognition	Achieved 95% accuracy in controlled environments	Performance drops in low light conditions
2	Real-Time Face Recognition for Attendance Using OpenCV	K. Singh, M. Verma	2019	OpenCV with Haar Cascade	Fast and lightweight implementation	Struggles with occlusions and angle variations
3	IoT-Enabled Face Recognition for Smart Attendance	S. Das, L. Roy	2022	Raspberry Pi with cloud-based processing	Remote access and real-time monitoring	Requires stable internet connection
4	Hybrid Model for Face Recognition Attendance System	J. Lee, T. Wong	2023	CNN + LBP for feature extraction	Improved accuracy under varied lighting	Icccccncreased processing time

2.1 Existing System



2.2 Proposed System

The proposed system uses computer vision to detect and recognize faces through a webcam. Key features include:

- Training the model using facial data of registered students.
- Storing attendance records in CSV files.
- GUI-based interface for ease of use.
- Options to delete records and retrain the system.

Advantages:

- No physical contact or additional hardware.
- Reduces chances of proxy attendance.
- Provides visual confirmation and log of each entry.

3. SYSTEM ANALYSIS

3.1 Functional Requirements

3.1.1 Register student: Capture and save ID, Name, and facial data.

3.1.2 Train model: Use captured images to train the facial recognition model.

3.1.3 Mark attendance: Detect and recognize faces in real time and log attendance.

3.1.4 Delete student: Remove all records of a student including images and details.

3.1.5 Open student records/attendance logs: View saved CSV files.

3.2Non-Functional Requirements

3.2.1 Usability: Simple and intuitive GUI for all functions.

3.2.2 Reliability: Consistent performance under various lighting conditions.

3.2.3 Maintainability: Modular code structure to ease debugging and enhancements.

3.2.4 Performance: Real-time face recognition with minimal lag.

3.2.5 Security: Controlled access to data and functionality to delete records securely.

3.3 Software Requirements

1. Python 3.x
2. OpenCV
3. Tkinter
4. PIL (Pillow)
5. NumPy
6. CSV module
7. Operating System: Windows/Linux

3.4 Hardware Requirements

1. Webcam (built-in or external)
2. Minimum 4 GB RAM
3. Dual-core Processor (i3 or above recommended)
4. Minimum 500 MB free disk space
5. Display: 1280x720 resolution for optimal GUI display

3.5 Block Diagram Of The Proposed System

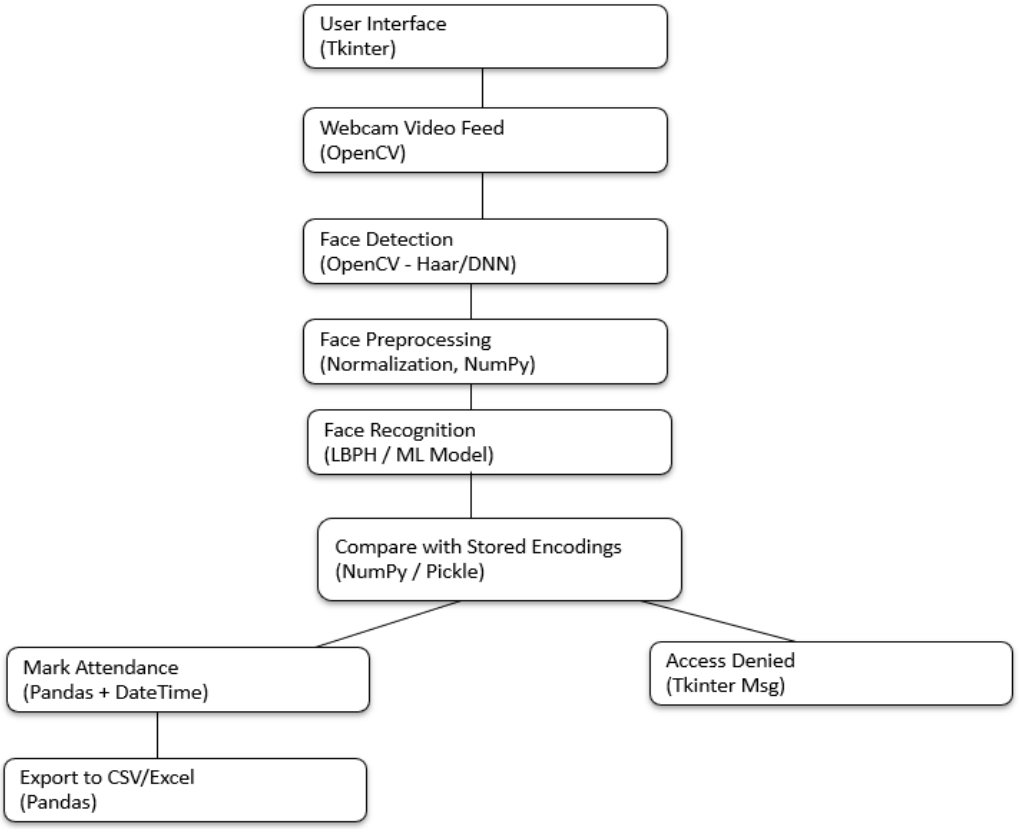


Fig 3.5: Block diagram of the system

4. SYSTEM DESIGN

4.1 Introduction

The system design adopts a modular architecture to ensure scalability, maintainability, and efficiency. A centralized Graphical User Interface (GUI) serves as the main control panel for administrators to manage attendance, register users, and view reports. The Face Recognition Module handles real-time image capture, face detection, and identification using pre-trained machine learning models. The Database Module securely stores user information, facial data, and attendance records. A Training Module processes facial images to update the recognition model for improved accuracy. Each module operates independently but interacts seamlessly for smooth system functionality. Error handling and logging mechanisms ensure system reliability. The modular approach also allows for future enhancements like cloud integration or mobile compatibility.

4.2 UML Diagrams

UML, which stands for Unified Modelling Language, is a way to visually represent the architecture, design, and implementation of complex software systems. Looking at code and understanding the system is difficult, instead we can use UML diagrams to understand the system. UML diagrams can be used to explain the components of the software to people who don't have technical knowledge.

It is a standard language for specifying, visualizing, constructing, and documenting the artefacts of the software systems. UML is different from other common programming languages like C++, Java, and COBOL etc. It is pictorial language used to make software blueprints.

Although typically used in software engineering it is a rich language that can be used to model an application structure, behaviour and even business processes. There are 4 UML diagram types to help us model this behaviour.

4.2.1 Class Diagram

Because a lot of software is based on object-oriented programming, where developers define types of functions that can be used, class diagrams are the most commonly used type of UML diagram. Class diagrams show the static structure of a system, including classes, their attributes and behaviours, and the relationships between each class. A class is represented by a rectangle that contains three compartments stacked vertically: Class name, attributes and functions of that class with access specifiers. The class diagram also contains association, generalization and aggregation. Where generalization is to show inheritance of a class and aggregation is to show that objects are assembled or configured together to create a more complex object.

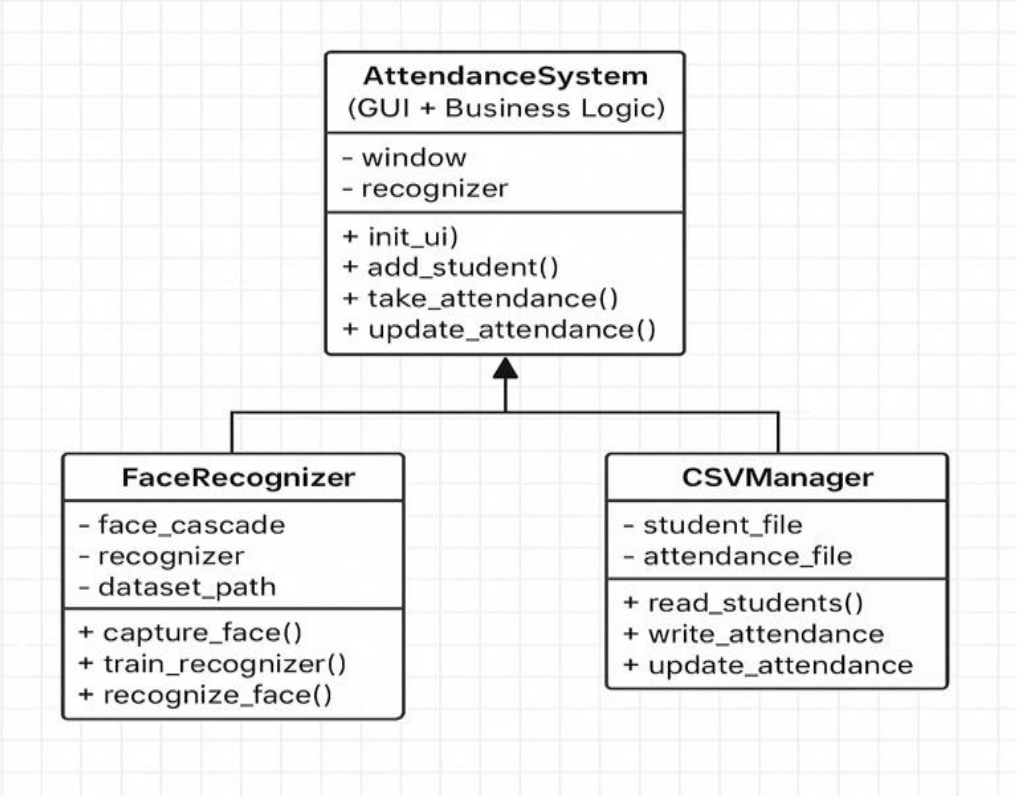


Fig 4.2.1: Class diagram

4.2.2 Activity Diagram

The Unified Modelling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behaviour diagrams. Activity diagrams, along with use case and state machine diagrams, are considered behaviour diagrams because they describe what must happen in the system being modelled. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

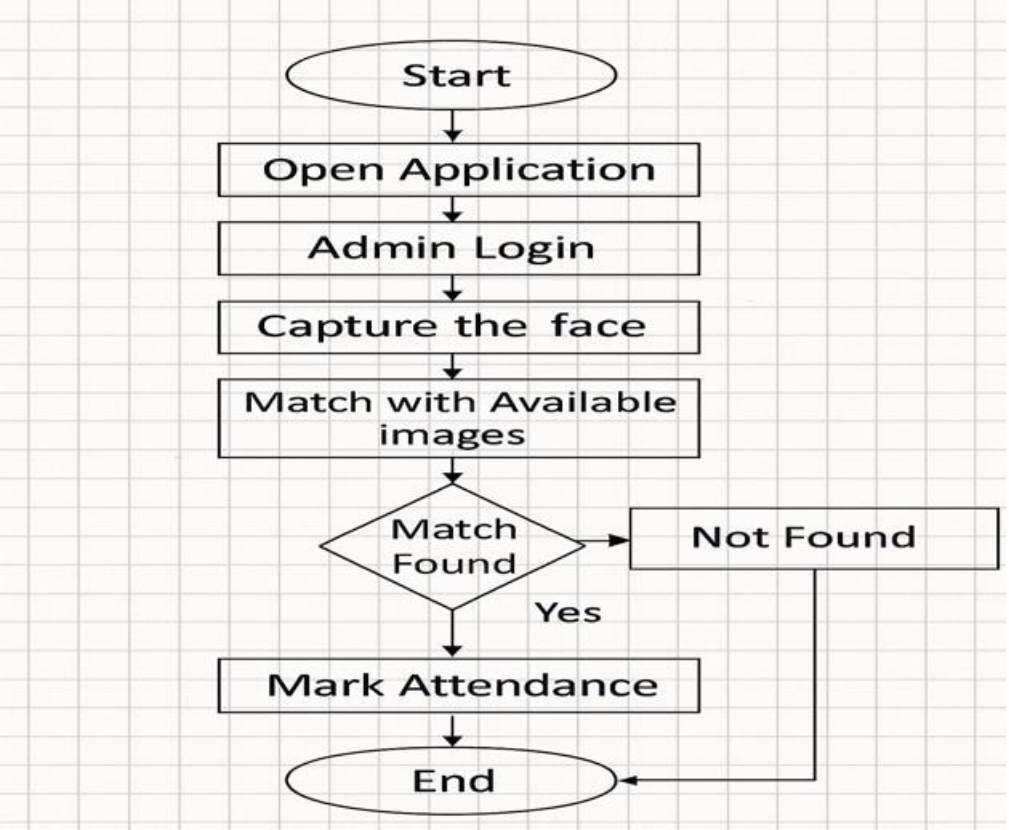


Fig 4.2.2: Activity diagram

4.2.3 Use case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. It is useful in the following situations: Scenarios in which your system or application interacts with people, organizations, or external systems, Goals that your system or application helps those entities (known as actors) achieve, The scope of your system. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how).

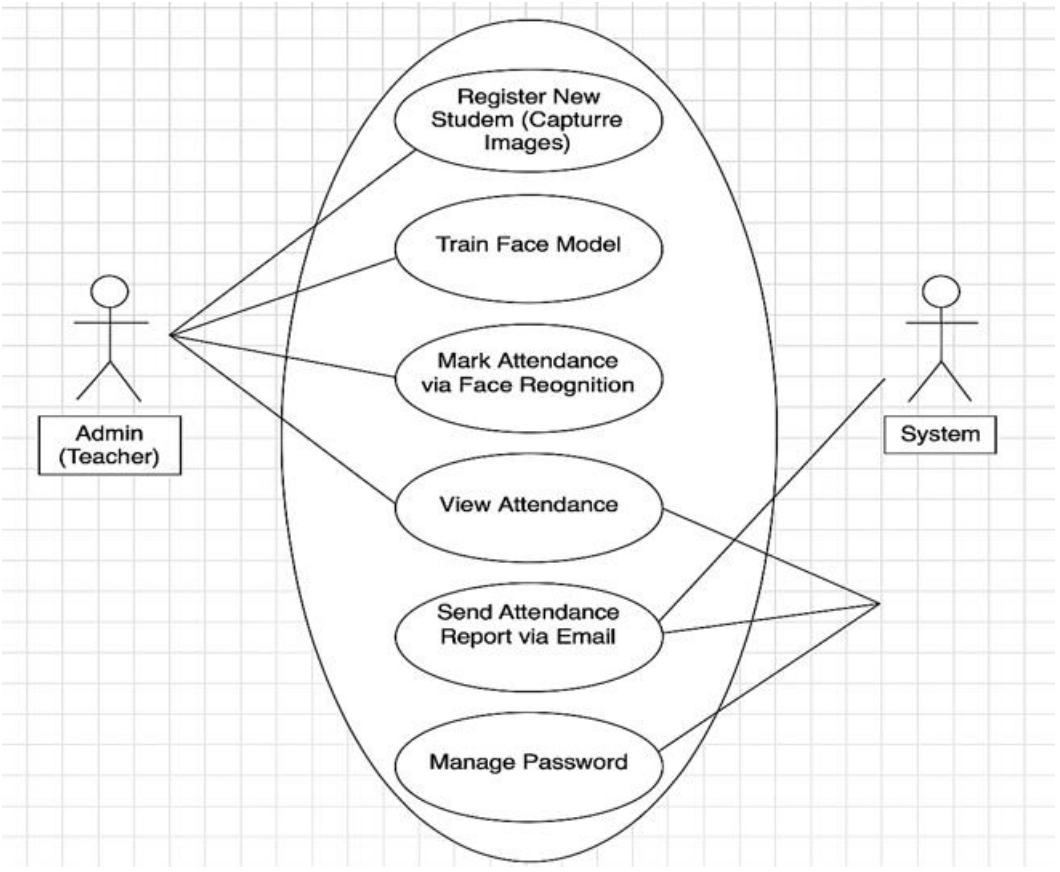


Fig 4.2.3: Use case diagram

4.2.4 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, A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

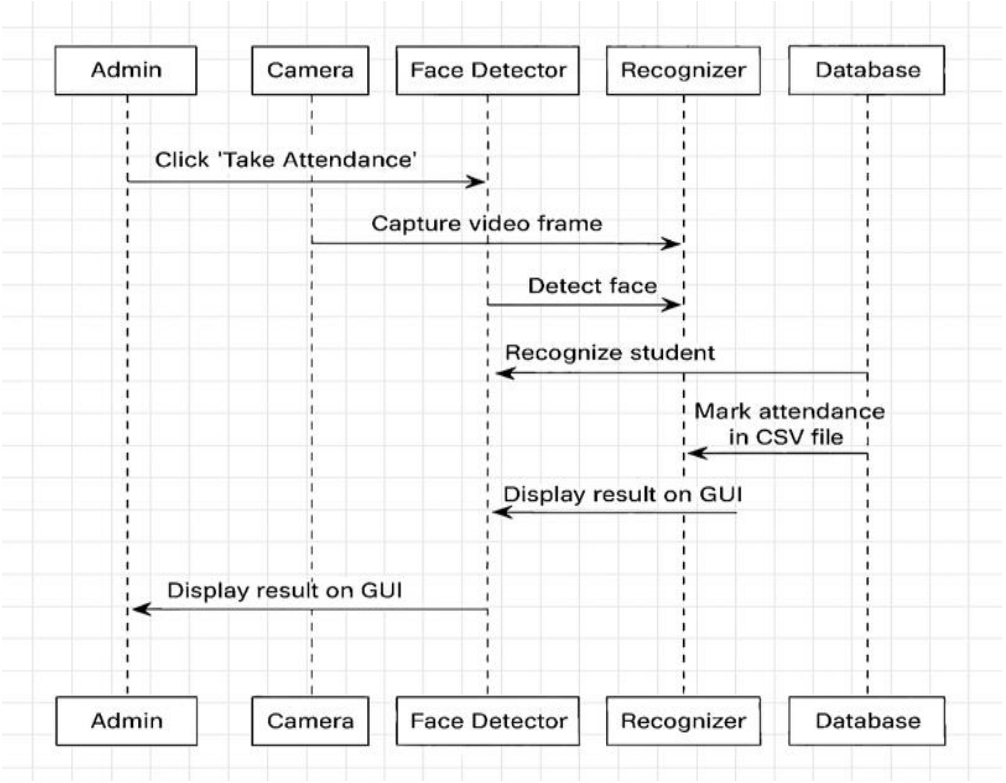


Fig 4.2.4: Sequence diagram

4.3 Modules

4.3.1 Data Collection and Preprocessing Module

💡 Purpose:

Collect, clean, and structure facial image data to prepare it for training the face recognition model.

✅ Responsibilities:

Import image datasets (e.g., captured images or images from databases).

Perform face detection and cropping from raw images.

Handle missing or low-quality images.

Normalize and resize facial images for uniformity.

Split the dataset into training, validation, and test sets.

🛠Tools Used:

Python libraries: OpenCV, pandas, numpy.

4.3.2 Feature Extraction Module

💡 Purpose:

Extract unique facial features from images for accurate recognition.

✅ Responsibilities:

Apply face embedding techniques (e.g., using FaceNet, Dlib, or DeepFace).

Convert images into feature vectors for model input.

🛠Tools Used:

Python libraries: OpenCV

4.3.3 Attendance Management Module

💡 Purpose:

Automate attendance marking based on recognized faces.

✅ Responsibilities:

Detect and recognize faces in real-time using a camera feed.

Mark attendance and record it in a database or file system.

Generate attendance reports.

🛠 Tools Used:

Python libraries: OpenCV, pandas, SQL Lite

4.3.4 Face Recognition & Classification Module

💡 Purpose:

Train a machine learning model to recognize and classify faces.

✅ Responsibilities:

Train the model using extracted facial features.

Implement classifiers (e.g., SVM, KNN, or Deep Learning models).

Evaluate model accuracy and performance.

🛠Tools Used:

Python libraries: Open CV,Numpy

4.3.5 User Interface Module

💡 Purpose:

Provide a user-friendly interface for users to interact with the system.

✅ Responsibilities:

Design GUI for user registration, attendance view, and report generation.

Display real-time face detection and recognition.

🛠Tools Used:

Python libraries: Tkinter

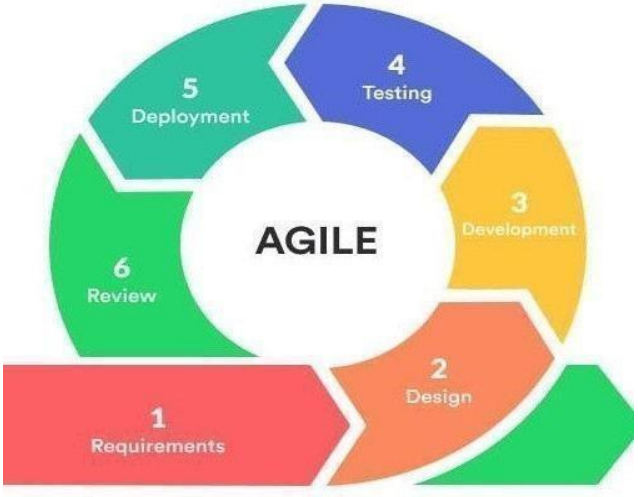
5.IMPLEMENTATION AND RESULTS

5.1Introduction

The implementation phase turns the design into a working system. In the Face Recognition Attendance System, facial recognition is developed using the LBPH algorithm and OpenCV to detect and identify faces in real-time via webcam. Recognized faces are matched with stored data, and attendance is marked with date and time. A database securely stores user profiles and attendance records, preventing duplication. The system was tested under various conditions to ensure accuracy. All components—face detection, recognition, data storage, and reporting—were integrated into a unified application for efficient, automated, and reliable attendance tracking.

5.2Method of Implementation

To ensure flexible and adaptive development, we adopted the Agile methodology, which emphasizes iterative progress, stakeholder collaboration, and continuous feedback. The project was broken down into focused development sprints, each targeting specific core functionalities:



- **Sprint 1:** UI Development & Camera Setup
  Designed the graphical user interface using Tkinter and integrated webcam support for live image capture.

- **Sprint 2:** Face Detection & Recognition
  Implemented facial recognition using OpenCV and the LBPH (Local Binary Pattern Histogram) algorithm for training and matching faces.
- **Sprint 3:** Database & Attendance Management
  Integrated CSV-based storage for student data and automated attendance logs, including timestamps.
- **Sprint 4:** Admin Controls & Reporting
  Added password-protected access, data deletion options, and exportable attendance reports for administrators.

Each sprint was reviewed and improved based on testing outcomes and usability feedback, ensuring a stable and user-friendly system

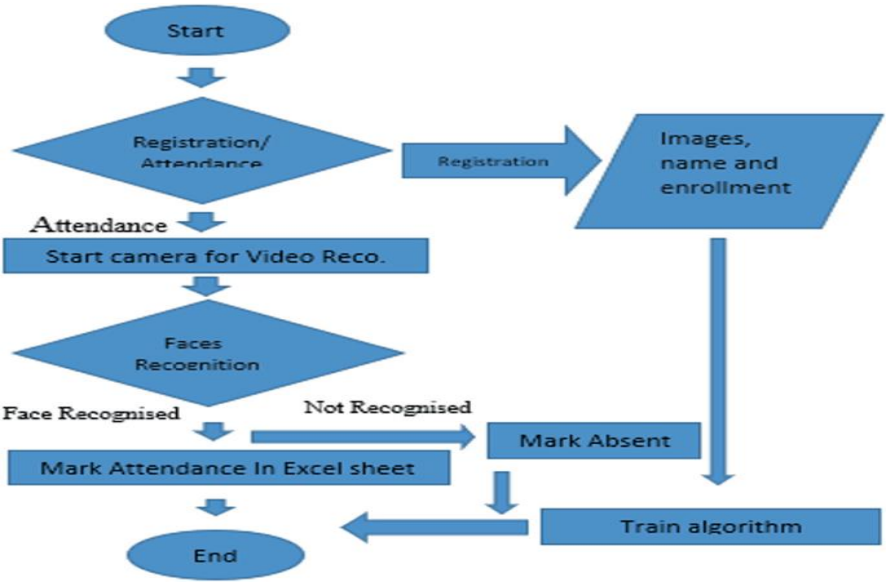
5.3 Algorithm and Flowcharts

Face Detection Algorithm

We use the Haar Cascade Classifier, a machine learning-based approach in OpenCV. It detects objects (faces) in images by scanning them with varying window sizes and comparing them to pre-trained features.

- **Input:** Live webcam video frame.
- **Process:** Converts frame to grayscale → applies Haar classifier → identifies coordinates of detected face.
- **Output:** Bounding box over detected face.

FlowChart:



5.4 Control Flow of Implementation

1. User starts the application.
2. Camera captures real-time image frames.
3. Face is detected using Haar Cascade.
4. Recognized faces are matched with database records.
5. If match found: Attendance marked with time & date.
6. Data stored in SQLite/MySQL.
7. Admin can view/download reports.

5.5 Sample Code

5.5.1 Utility Functions

These support operations like creating folders, checking files, or formatting.

- **assure_path_exists(path)**

```
def assure_path_exists(path):
        dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)
```

- **tick()**

```
def tick():
    # Get the current time
    current_time = time.strftime('%I:%M:%S %p')
    # Update the clock label with the current time
    clock.config(text=current_time)
    # Schedule the next update after 1000 milliseconds (1 second)
    clock.after(1000, tick)
```

- **contact()**

```
def contact():
    mess._show(title='Contact us', message="Please contact us on : 'nithinkata29@gmail.com'
")
```

- **check_haarcascadefile()**

```
def check_haarcascadefile():
```

```
        exists = os.path.isfile("haarcascade_frontalface_default.xml")
        if exists:
            pass
        else:
            mess._show(title='Some file missing', message='Please contact us for help')
            window.destroy()
```

- **clear() and clear2()**

```
    def clear():
        txt.delete(0, 'end')
        res = "1)Take Images  >>>  2)Save Profile"
        message1.configure(text=res)
    def clear2():
        txt2.delete(0, 'end')
        res = "1)Take Images  >>>  2)Save Profile"
        message1.configure(text=res)
```

5.5.2 Password Management

Handles password setting, validation, and updates.

- save_pass()
- change_pass()
- psw()

```
def save_pass():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below',
        show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!! Please try again')
        else:
```

```
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was registered
      successfully!!')
            return
    op = (old.get())
    newp= (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open("TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old password.')
        return
    mess._show(title='Password Changed', message='Password changed successfully!!')
    master.destroy()
##################################################################################
def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
    master.configure(background="white")
    lbl4 = tk.Label(master,text='    Enter Old Password',bg='white',font=('comic', 12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('comic', 12, ' bold '),show='*')
    old.place(x=180,y=10)
```

```
    lbl5 = tk.Label(master, text='   Enter New Password', bg='white', font=('comic', 12, ' bold '))
    lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('comic', 12, ' bold
      '),show='*')
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('comic', 12, ' bold '))
    lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('comic', 12, ' bold
      '),show='*')
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy ,fg="black"   ,bg="red"
      ,height=1,width=25 , activebackground = "white" ,font=('comic', 10, ' bold '))
    cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black", bg="#00fcca",
      height = 1,width=25, activebackground="white", font=('comic', 10, ' bold '))
    save1.place(x=10, y=120)
    master.mainloop()
################################################################################
def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below',
       show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!! Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
```

```
        tf.write(new_pas)
        mess._show(title='Password Registered', message='New password was registered
    successfully!!')
        return
  password = tsd.askstring('Password', 'Enter Password', show='*')
  if (password == key):
     TrainImages()
  elif (password == None):
     pass
  else:
          mess._show(title='Wrong Password', message='You have entered wrong password')
```

5.5.3 Image Capture & Training

Involves capturing user images and training the facial recognition model.

- **TakeImages()**
- **TrainImages()**

```
  def TakeImages():
     check_haarcascadefile()
     columns = ['SERIAL NO.', '', 'ID', '', 'NAME']
     assure_path_exists("StudentDetails/")
     assure_path_exists("TrainingImage/")
     serial = 0
     exists = os.path.isfile("StudentDetails\StudentDetails.csv")
     if exists:
        with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
           reader1 = csv.reader(csvFile1)
           for l in reader1:
              serial = serial + 1
        serial = (serial // 2)
        csvFile1.close()
     else:
        with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile1:
           writer = csv.writer(csvFile1)
```

```
            writer.writerow(columns)
            serial = 1
        csvFile1.close()
    Id = (txt.get())
    name = (txt2.get())
    if ((name.isalpha()) or (' ' in name)):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        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 TrainingImage
                cv2.imwrite("TrainingImage\ " + name + "." + str(serial) + "." + Id + '.' +
str(sampleNum) + ".jpg",
                            gray[y:y + h, x:x + w])
                # display the frame
                cv2.imshow('Taking Images', img)
            # wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum > 70:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Taken for ID : " + Id
```

```
            row = [serial, '', Id, '', name]
            with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:
                writer = csv.writer(csvFile)
                writer.writerow(row)
            csvFile.close()
            message1.configure(text=res)
        else:
            if (name.isalpha() == False):
                res = "Enter Correct name"
                message.configure(text=res)
    ##########################################################################
    def TrainImages():
        check_haarcascadefile()
        assure_path_exists("TrainingImageLabel/")
        recognizer = cv2.face_LBPHFaceRecognizer.create()
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        faces, ID = getImagesAndLabels("TrainingImage")
        try:
            recognizer.train(faces, np.array(ID))
        except:
            mess._show(title='No Registrations', message='Please Register someone first!!!')
            return
        recognizer.save("TrainingImageLabel\Trainner.yml")
        res = "Profile Saved Successfully"
        message1.configure(text=res)
        message.configure(text='Total Registrations till now  : ' + str(ID[0]))
```

5.5.4 Attendance Tracking

Uses the trained model to detect and recognize faces for attendance.

- **TrackImages()**

```
def TrackImages():
    check_haarcascadefile()
```

```
assure_path_exists("Attendance/")
assure_path_exists("StudentDetails/")
for k in tv.get_children():
    tv.delete(k)
msg = ''
i = 0
j = 0
recognizer = cv2.face.LBPHFaceRecognizer_create()  # cv2.createLBPHFaceRecognizer()
exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")
if exists3:
    recognizer.read("TrainingImageLabel\Trainner.yml")
else:
    mess._show(title='Data Missing', message='Please click on Save Profile to reset data!!')
    return
harcascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(harcascadePath);

cam = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
col_names = ['Id', '', 'Name', '', 'Date', '', 'Time']
exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")
if exists1:
    df = pd.read_csv("StudentDetails\StudentDetails.csv")
else:
    mess._show(title='Details Missing', message='Students details are missing, please check!')
    cam.release()
    cv2.destroyAllWindows()
    window.destroy()
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:
```

```
        cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
        serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
        if (conf < 50):
            ts = time.time()
            date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
            timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%I:%M:%S %p')
            aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values
            ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
            ID = str(ID)
            ID = ID[1:-1]
            bb = str(aa)
            bb = bb[2:-2]
            attendance = [str(ID), '', bb, '', str(date), '', str(timeStamp)]
        else:
            Id = 'Unknown'
            bb = str(Id)
        cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
    cv2.imshow('Taking Attendance', im)
    if (cv2.waitKey(1) == ord('q')):
        break
ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
exists = os.path.isfile("Attendance\Attendance_" + date + ".csv")
if exists:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
        writer = csv.writer(csvFile1)
        writer.writerow(attendance)
    csvFile1.close()
else:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as csvFile1:
        writer = csv.writer(csvFile1)
        writer.writerow(col_names)
        writer.writerow(attendance)
```

```
        csvFile1.close()
    with open("Attendance\Attendance_" + date + ".csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for lines in reader1:
            i = i + 1
            if (i > 1):
                if (i % 2 != 0):
                    iidd = str(lines[0]) + '   '
                    tv.insert('', 0, text=iidd, values=(str(lines[2]), str(lines[4]), str(lines[6])))
    csvFile1.close()
    cam.release()
        cv2.destroyAllWindows()
```

5.5.5 Data Deletion

Removes registered data by ID, or bulk deletes stored files.

- **deletePersonCompletely(person_id)**

```
#function to delete person details completely
def deletePersonCompletely(person_id):
    found = False
 # 1. Delete training images
    image_dir = "TrainingImage"
    for file in os.listdir(image_dir)
  if file.split('.')[2] == person_id:
            os.remove(os.path.join(image_dir, file))
            found = True


    # 3. Remove from StudentDetails.csv
    student_path = "StudentDetails/StudentDetails.csv"
    if os.path.exists(student_path):
        with open(student_path, "r") as f:
            rows = list(csv.reader(f))
        with open(student_path, "w", newline='') as f:
            writer = csv.writer(f)
```

```
        for row in rows:
            if row and row[0] != person_id:
                writer.writerow(row)
            else:
                found = True

# 4. Update registration count file (if exists)
reg_path = "StudentDetails/registration.csv"
if os.path.exists(reg_path):
    with open(reg_path, "r") as f:
        count = int(f.read().strip())
    count = max(0, count - 1)
    with open(reg_path, "w") as f:
        f.write(str(count))

# 5. Retrain
if found:
    TrainImages()
    message.configure(text=f"Deleted person ID {person_id} from all records.")
else:
        message.configure(text=f"No records found for ID {person_id}.")
```

5.5.6 GUI Construction

All code for building the graphical interface and linking functions to widgets**.**

- window = tk.Tk() to window.mainloop()
- Frames, Labels, Buttons, TreeView, Scrollbars, Entries

```
########################GUI-FRONEND##########################
window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
```

```
window.title("Attendance System")
window.configure(background='#2d420a')

# Load the background image
bg_image = Image.open("C:/Users/NITHIN KATA/Desktop/finalmini/Face-Recognition-Based-Attendance-Monitoring-System/nnrg-ai-background.png")
bg_photo = ImageTk.PhotoImage(bg_image)

# Set the background image of the window
background_label = tk.Label(window, image=bg_photo)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

frame1 = tk.Frame(window, bg="#c79cff")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#c79cff")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="Face Recognition Based Attendance Monitoring System" ,fg="white",bg="#2d420a" ,width=55 ,height=1,font=('comic', 29, ' bold '))
message3.place(x=10, y=10)

datef = tk.Label(window, text=day + "-" + mont[month] + "-" + year, fg="#ff61e5", bg="green",
                 width=20, font=('comic', 15, ' bold '))
datef.pack(fill='both', expand=True)
datef.place(relx=0.30, rely=0.09)

clock = tk.Label(window, fg="#ff61e5", bg="green", width=20, font=('comic', 15, ' bold '))
clock.place(relx=0.50, rely=0.09)
```

```
tick()

head2 = tk.Label(frame2, text="                For New Registrations                ",
    fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )
head2.place(x=0,y=-5)

head1 = tk.Label(frame1, text="                For Already Registered                ",
    fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )
head1.place(x=0,y=-5)

lbl = tk.Label(frame2, text="Enter ID",width=20  ,height=1  ,fg="black"
    ,bg="#c79cff" ,font=('comic', 17, ' bold ') )
lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))
txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter Name",width=20  ,fg="black"
    ,bg="#c79cff" ,font=('comic', 17, ' bold '))
lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold ')  )
txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images  >>>  2)Save Profile"
    ,bg="#c79cff" ,fg="black"  ,width=39 ,height=1, activebackground =
    "#3ffc00" ,font=('comic', 15, ' bold '))
message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#c79cff" ,fg="black"
    ,width=39,height=1, activebackground = "#3ffc00" ,font=('comic', 16, ' bold
    '))
message.place(x=7, y=450)
```

```
lbl3 = tk.Label(frame1, text="Attendance",width=20 ,fg="black"
    ,bg="#c79cff" ,height=1 ,font=('comic', 15, ' bold '))
lbl3.place(x=100, y=125)


res=0
exists = os.path.isfile("StudentDetails\StudentDetails.csv")
if exists:
    with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            res = res + 1
    res = (res // 2) - 1
    csvFile1.close()
else:
    res = 0
message.configure(text='Total Registrations till now  : '+str(res))
######################MENUBAR#################################
menubar = tk.Menu(window,relief='ridge')
filemenu = tk.Menu(menubar,tearoff=0)
filemenu.add_command(label='Change Password', command = change_pass)
filemenu.add_command(label='Contact Us', command = contact)
filemenu.add_command(label='Exit',command = window.destroy)
menubar.add_cascade(label='Help',font=('comic', 29, ' bold '),menu=filemenu)
##################TREEVIEWATTENDANCETABLE###############
tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))
tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)
tv.heading('#0',text ='ID')
```

```
tv.heading('name',text ='NAME')
tv.heading('date',text ='DATE')
tv.heading('time',text ='TIME')
######################SCROLLBAR#################################
scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)
scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')
tv.configure(yscrollcommand=scroll.set)

# Add horizontal scrollbar to Treeview
scroll_x = ttk.Scrollbar(frame1, orient='horizontal', command=tv.xview)
scroll_x.grid(row=3, column=0, pady=(0, 20), padx=(0, 100), sticky='ew')
tv.configure(xscrollcommand=scroll_x.set)
######################BUTTONS#################################
    ##

clearButton = tk.Button(frame2, text="Clear", command=clear  ,fg="black"
    ,bg="#ff7221"  ,width=11 ,activebackground = "white" ,font=('comic', 11, '
    bold '))
clearButton.place(x=335, y=86)
clearButton2 = tk.Button(frame2, text="Clear", command=clear2  ,fg="black"
    ,bg="#ff7221"  ,width=11 , activebackground = "white" ,font=('comic', 11, '
    bold '))
clearButton2.place(x=335, y=172)
takeImg = tk.Button(frame2, text="Take Images", command=TakeImages
    ,fg="white"  ,bg="#6d00fc"  ,width=34  ,height=1, activebackground =
    "white" ,font=('comic', 15, ' bold '))
takeImg.place(x=30, y=300)
trainImg = tk.Button(frame2, text="Save Profile", command=psw ,fg="white"
    ,bg="#6d00fc"  ,width=34  ,height=1, activebackground = "white"
    ,font=('comic', 15, ' bold '))
trainImg.place(x=30, y=380)
```

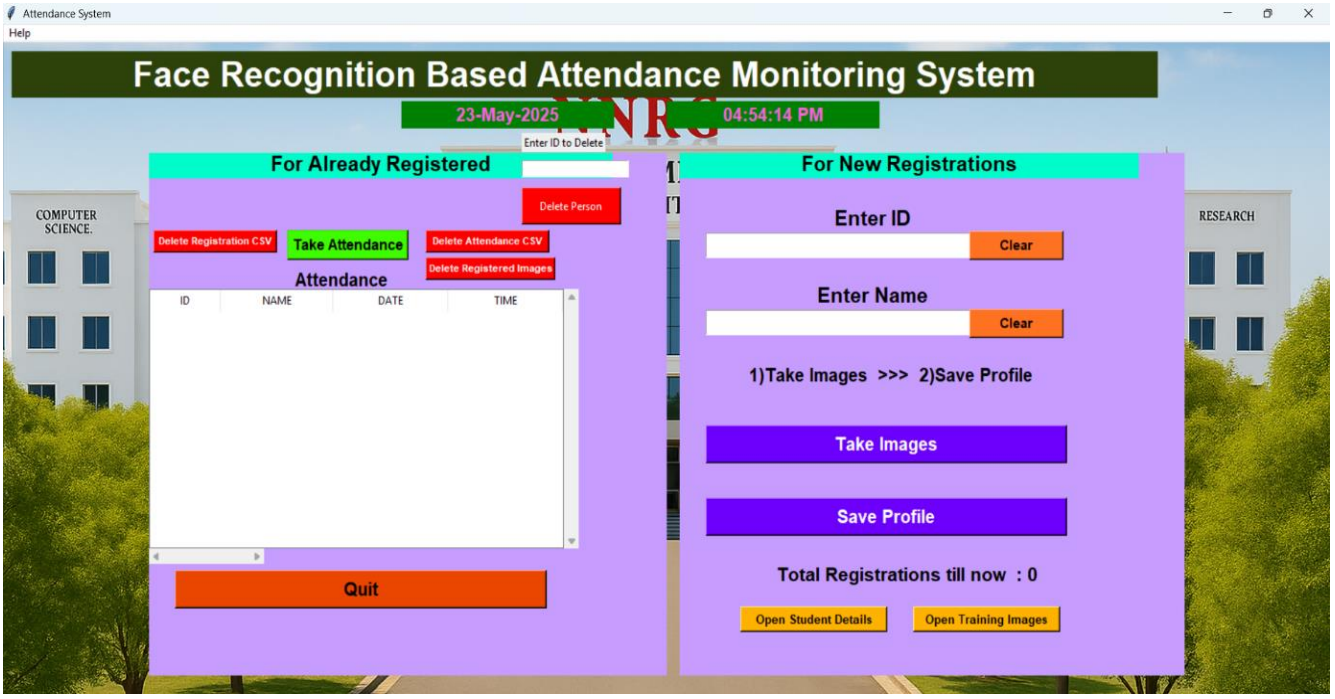
```
trackImg = tk.Button(frame1, text="Take Attendance", command=TrackImages
    ,fg="black"  ,bg="#3ffc00"  ,width=13  ,height=1, activebackground =
    "white" ,font=('comic', 12, ' bold '))
trackImg.place(x=160,y=85)
quitWindow = tk.Button(frame1, text="Quit", command=window.destroy
    ,fg="black"  ,bg="#eb4600"  ,width=35 ,height=1, activebackground =
    "white" ,font=('comic', 15, ' bold '))
quitWindow.place(x=30, y=460)

def delete_registration_csv():
    registration_csv_path = "StudentDetails\StudentDetails.csv"
    if os.path.exists(registration_csv_path):
        os.remove(registration_csv_path)
        mess.showinfo("Success", "Registration CSV file deleted successfully.")
    else:
        mess.showinfo("Error", "Registration CSV file not found.")

def delete_attendance_csv():
    today = datetime.datetime.now().strftime('%d-%m-%Y')
    attendance_csv_path = f"Attendance\Attendance_{today}.csv"
    if os.path.exists(attendance_csv_path):
        os.remove(attendance_csv_path)
        mess.showinfo("Success", f"Attendance CSV file for {today} deleted
        successfully.")
    else:
        mess.showinfo("Error", f"Attendance CSV file for {today} not found.")

# Create buttons for deleting registration and attendance CSV files
delete_registration_button = tk.Button(frame1, text="Delete Registration CSV",
    command=delete_registration_csv, fg="white", bg="red", width=19,
    font=('comic', 8, 'bold'))
delete_registration_button.place(x=5, y=85)
```

```
delete_attendance_button = tk.Button(frame1, text="Delete Attendance CSV",
    command=delete_attendance_csv, fg="white", bg="red", width=19,
    font=('comic', 8, 'bold'))
delete_attendance_button.place(x=320, y=85)

def delete_registered_images():
    folder_path = "TrainingImage/"
    if os.path.exists(folder_path):
        # Get all files in the folder
        files = os.listdir(folder_path)
        for file in files:
            file_path = os.path.join(folder_path, file)
            try:
                # Delete the file
                os.remove(file_path)
            except Exception as e:
                # Show error message if deletion fails
                mess.showinfo("Error", f"Failed to delete {file}: {e}")
        mess.showinfo("Success", "Registered images deleted successfully.")
    else:
        mess.showinfo("Error", "TrainingImage folder not found.")

# Create a button to delete registered images
delete_images_button = tk.Button(frame1, text="Delete Registered Images",
    command=delete_registered_images, fg="white", bg="red", width=20,
    font=('comic', 8, 'bold'))
delete_images_button.place(x=320, y=115)

# create a button to delete complete deatails of a particular person
deleteId = tk.StringVar()
tk.Label(window, text="Enter ID to Delete").place(x=600, y=100)
tk.Entry(window, textvariable=deleteId, width=20).place(x=600, y=130)
```

```
tk.Button(window, text="Delete Person", command=lambda:
    deletePersonCompletely(deleteId.get()),fg="white", bg="red" , width=15,
    height=2).place(x=600, y=160)


def open_student_details():
    path = "StudentDetails"
    if os.path.exists(path):
        os.startfile(path)  # Opens with default app
    else:
        mess.showinfo("Error", "StudentDetails.csv not found.")

def open_training_images():
    folder = "TrainingImage"
    if os.path.exists(folder):
        os.startfile(folder)  # Opens the folder
    else:
        mess.showinfo("Error", "TrainingImage folder not found.")

# Add the buttons at the bottom-right of frame2
student_details_btn = tk.Button(frame2, text="Open Student Details",
    command=open_student_details, bg="#ffb300", fg="black", font=('comic',
    10, 'bold'), width=20)
student_details_btn.place(x=70, y=500)

training_images_btn = tk.Button(frame2, text="Open Training Images",
    command=open_training_images, bg="#ffb300", fg="black", font=('comic',
    10, 'bold'), width=20)
training_images_btn.place(x=270, y=500)
```

5.6 Output Screens

5.6.1.Home Interface – System Dashboard



Allows users to take attendance, view logs, or delete data.

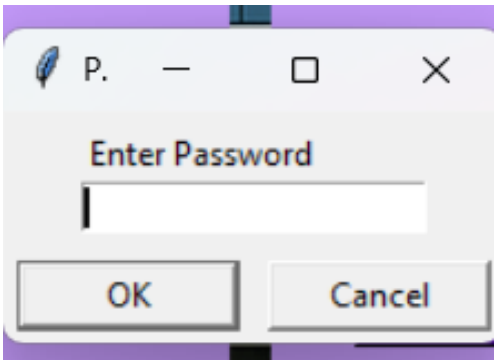
- For New Registrations:

  Fields to enter a new student's ID and name, and buttons to capture their face and save the profile.

  Other features include:

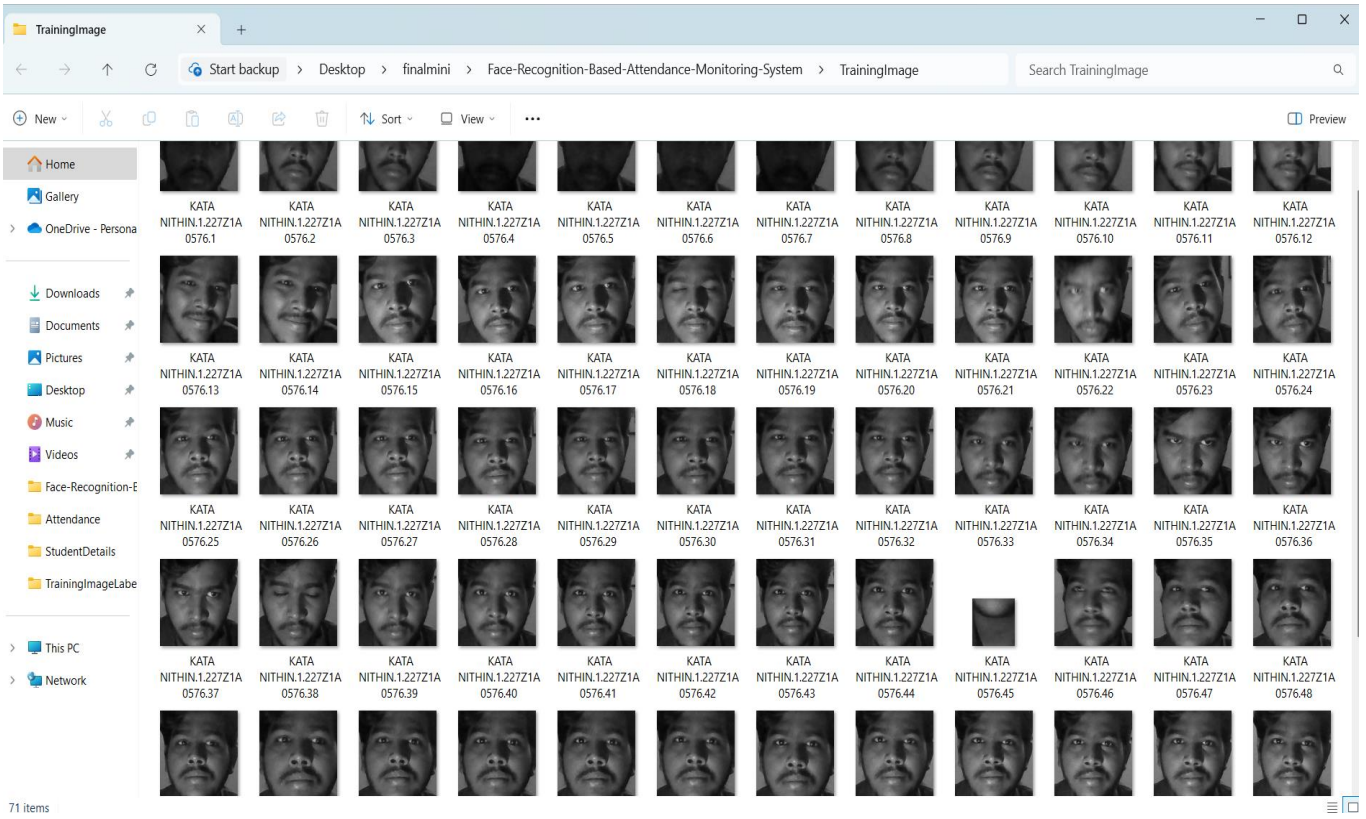
- Live date and time
- Options to delete registration CSV, delete attendance, and delete stored images
- Display of attendance table

5.6.2. Password Entry Prompt



When a user attempts to **train or update images**, a password prompt is shown. This secures the system, ensuring only authorized users can modify the training set.

5.6.3. Face Data Collection – Training Images



This screenshot shows the **TrainingImage folder** where 70+ grayscale images are saved for each registered person. These images are taken during the registration process and are later used to train the face recognition model using the **LBPH algorithm**.

5.6.4. Student Registration Details – CSV Record

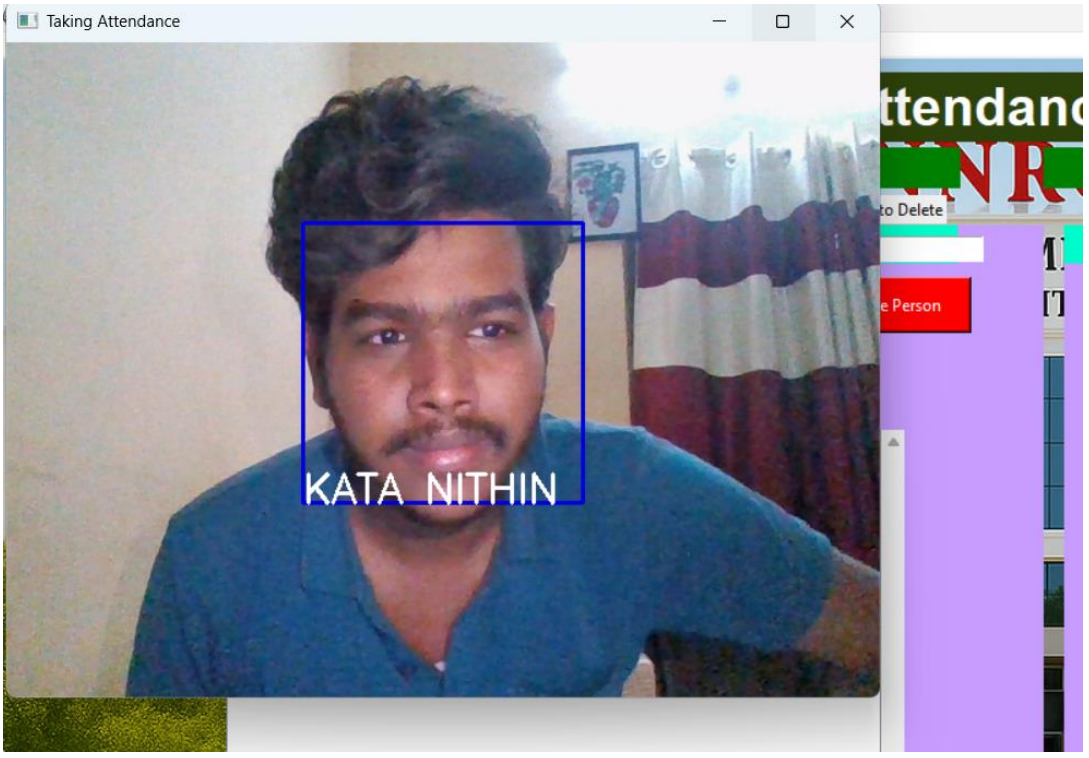
A	B	C	D	E	F
SERIAL NO.		ID		NAME	
1		227Z1A0576		KATA NITHIN	
2		227Z1A0591		L. BINDU	
3		227Z1A0582		K. PRAVEEN	

Along with attendance logs, the system maintains a **StudentDetails.csv** file. This file is updated each time a new student is registered. It stores:

- Serial Number
- Student ID
- Student Name

This CSV acts as a master record and is essential for face recognition. During attendance, the system references this file to match recognized faces with student IDs and names. It can be accessed through the "Open Student Details" button in the GUI.

5.6.5. Real-Time Face Recognition



This shows the system in **attendance mode**. The camera captures live video, detects faces using the **Haar Cascade Classifier**, and recognizes known users based on trained data. A blue rectangle marks the face and the name label is displayed if matched.

5.6.6.Attendance Log Output

Id	Name	Date	Time
' 22Z1A0576'	KATA NITHIN	29-04-2025	8:39:54 PM

When a match is found, the system records:

- **ID**
- **Name**

- **Date**
- **Time**

The data is stored in a daily CSV file in the Attendance folder. This enables easy tracking and exporting of attendance logs.

6.TESTING

6.1 Introduction to Testing

Software testing is a process to evaluate the functionality and performance of a software application to determine whether it meets specified requirements. In this project, testing ensured the system accurately recognized faces, recorded attendance reliably, and was secure and user-friendly. The goal was to detect and fix any defects to ensure the system is robust, error-free, and ready for deployment.

6.2 Types of Tests Considered

- **Application Testing:**
  Tested the overall application workflow including registration, face capture, recognition, and data storage.
- **System Testing:**
  Validated the end-to-end functioning of the complete system, including GUI, face recognition, and CSV logging.
- **GUI Testing:**
  Checked user interface elements like buttons, text fields, and the real-time attendance display for correctness and responsiveness.
- **Security Testing:**
  Tested password-protected areas (e.g., training module access) to ensure unauthorized access is blocked.

6.3 Various Test Case Scenarios Considered

Test cases were designed to verify all functional and edge cases in the system. Below is a summary:

Table 6.3: Test Cases for Face Recognition Attendance System

TEST CASE ID	TEST CASE SCENARIO	INPUTS	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TC01	Login to training module	Enter correct password	Access to image training allowed	Access granted	PASS
TC02	Register new user	Enter ID and Name, capture face images	Images saved and profile stored	Profile saved successfully	PASS
TC03	Take attendance (known face)	Face in front of camera	Face recognized, attendance marked with timestamp	Attendance marked	PASS
TC04	Take attendance (unknown face)	Unregistered face shown	Display "Unknown", attendance not marked	Unknown shown	PASS
TC05	View attendance records	Open attendance CSV	Display of date-wise attendance data	CSV opened successfully	PASS
TC06	Delete a student's data	Enter ID to delete	Related images and data removed, model retrained	Data deleted, model updated	PASS
TC07	GUI functionality	Click buttons, enter data	Buttons perform their respective actions	All buttons working correctly	PASS

7. Conclusion and Future Enhancements

7.1 Conclusion

The Face Recognition Attendance System developed in this project provides a modern, efficient, and automated method for recording attendance using facial recognition technology. It eliminates the need for manual attendance marking, reduces human errors, and enhances the accuracy and speed of attendance tracking. By leveraging OpenCV and the LBPH algorithm, the system effectively detects and recognizes faces in real time. The GUI built using Tkinter ensures ease of use for both administrators and users. Security features such as password protection for critical functions ensure safe operation. Overall, the system meets its objectives and can be deployed in schools, colleges, or workplaces for effective attendance monitoring.

7.2 Future Enhancements

To further improve the system and adapt it to real-world needs, the following enhancements are proposed:

- Mask Detection Integration: To recognize faces with masks, especially in post-COVID environments.
- Mobile App Integration: Allow users to mark attendance via a mobile camera app linked to the main system.
- Cloud Storage: Store attendance data on a cloud server for remote access and backup.
- Advanced Face Recognition (Deep Learning): Use CNN-based models for higher accuracy and faster processing.
- Real-time Notifications: Send attendance alerts to admin or students via email or SMS.
- Geo-Fencing: Mark attendance only if the student is present within a specific location boundary (e.g., campus).
- Multi-Camera Support: Support for multiple cameras for large-scale institutions

8. REFERENCES

8.1 Journals

[1] Patel, D., Patel, M., & Thakkar, P. (2021). "Automated Attendance System Using Face Recognition." *International Journal of Engineering Research & Technology (IJERT)*.

[2] Yadav, P., Chavan, S., & Patil, S. (2020). "Face Recognition-Based Attendance System using Machine Learning." *International Research Journal of Engineering and Technology (IRJET)*.

[3] Bhoi, A. K., & Satpathy, S. (2019). "Real-Time Face Recognition with Attendance Marking." *International Journal of Advanced Research in Computer Science*.

[4] Shrivastava, P., & Sharma, D. (2020). "Face Detection and Recognition System using Python and OpenCV." *Journal of Emerging Technologies and Innovative Research (JETIR)*.

[5] Kaur, S., & Kaur, R. (2022). "Implementation of LBPH Algorithm for Face Recognition in Attendance Systems." *IEEE Xplore Digital Library*.

8.2 Books

- *Python Programming for Beginners* by Adam Stark
- *Head First Python* (2nd Edition) by Paul Barry
- *Programming Computer Vision with Python* by Jan Erik Solem
- *Python for Data Analysis* by Wes McKinney

8.3 Sites

- https://opencv.org/
- https://realpython.com/
- https://www.geeksforgeeks.org/face-recognition-using-opencv-in-python/
- https://www.kaggle.com/