INTRODUCTION

1.1 PROJECT INTRODUCTION

This project explores the integration of OpenCV and LBPH for real-time attendance monitoring while ensuring accurate and efficient student identification. OpenCV, a powerful computer vision library, is utilized to detect and recognize faces, enabling real-time responses without the need for manual attendance processes. By integrating the LBPH (Local Binary Patterns Histogram) face recognizer, the project ensures that the system can handle facial recognition with high precision and consistency, even in diverse lighting conditions and facial expressions.

A core component of the project is OpenCV's role in detecting and processing facial images. Using Haar Cascade Classifiers, the system is capable of identifying faces from live video feeds or static images. These detections are then processed using the LBPH algorithm, which extracts critical features from the facial images for recognition. This combination ensures that the system can operate efficiently in real-time, reducing the time required to capture and log attendance across sessions.

The project also focuses on data management using a CSV database, where attendance records are securely stored and updated in real-time. Pandas, a powerful data manipulation library in Python, is used to handle the data, ensuring that the attendance logs are properly maintained, filtered, and easily accessible for future reference. The integration of a graphical user interface (GUI) built with Tkinter provides users with a seamless experience for selecting subjects and monitoring attendance across multiple sessions.

Incorporating real-time facial recognition and data logging enhances the system's ability to track student attendance efficiently. The system captures student information automatically without manual input, ensuring accuracy in recording and minimizing human error. This integration allows educational institutions to automate their attendance processes, improving operational efficiency and enabling real-time monitoring and reporting.

Tkinter's GUI plays a crucial role in simplifying user interaction. By allowing administrators to easily select subjects and manage attendance records through a user-friendly interface, the system streamlines the attendance marking process, reducing the need for technical expertise and making it accessible to non-technical users.

In addition to leveraging OpenCV and LBPH for accurate face detection and recognition, this project emphasizes the importance of real-time data processing and automation. By utilizing these technologies, the system ensures that attendance is captured instantly, reducing time spent on manual tasks and providing real-time access to attendance data. This approach underscores the significance of automation in educational environments, where accurate attendance tracking is essential for academic and administrative purposes.

Lastly, scalability and adaptability are significant benefits of this system. OpenCV's flexible nature ensures that the system can be adapted to different environments and user requirements, such as varying lighting conditions or different class sizes. Combined with the ability to store and process attendance data in real-time, this solution minimizes operational overhead while providing a robust and automated attendance management system that meets both academic and administrative needs.

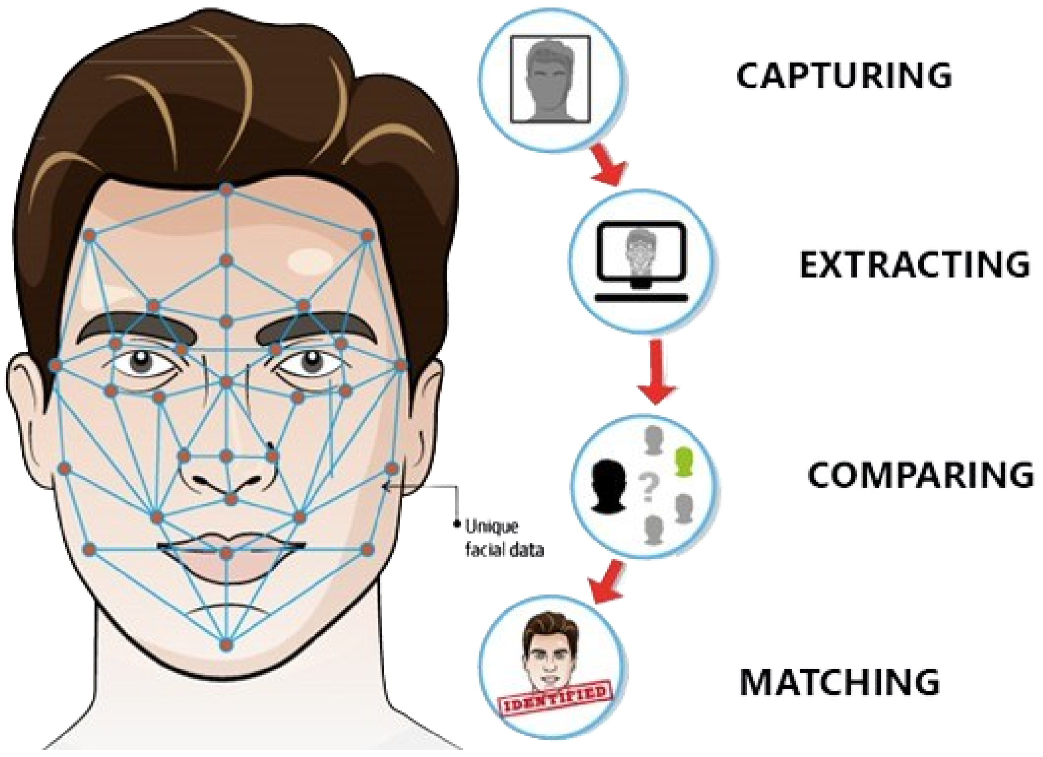

Fig 1.1: Face Recognition

WORKING ENVIRONMENT

2.1 Hardware Requirements

- Standard Keyboard and Mouse
- High-Resolution Monitor

2.2 Software Requirements

- **Operating System**: Windows 7 or 10
- **Platform**: Local development environment using Python and OpenCV for face detection and recognition.
- **Language Used**: Python for core programming and GUI development.
- **Python Libraries**:
    - **OpenCV** for image and video processing.
    - **Pandas** for data handling and CSV management.
    - **Tkinter** for GUI development.
    - **NumPy** for numerical computations and matrix operations.

2.3 System Software

- **Processor**: Intel i3 @ 2.5 GHz or AMD 5 Series
- **RAM**: Minimum of 8 GB
- **Hard Disk/SSD**: Minimum of 128 GB

SYSTEM ANALYSIS

3.1 Feasibility Analysis

The feasibility of the project is analyzed during the initial phase, presenting a high-level plan and a rough cost estimate. This ensures that the proposed system is both practical and sustainable. During the system analysis, a feasibility study of the real-time attendance system is conducted to ensure that it meets the operational needs of educational institutions without becoming a burden. Understanding the key system requirements is crucial for this evaluation.

A comprehensive feasibility study provides a background on the system's objectives, technical components, cost analysis, and user adoption. The study precedes the development and implementation phases, evaluating the project’s potential for success based on the resources and requirements identified. To ensure objectivity, the study focuses on five key areas of feasibility: Economic, Technical, Social, Operational, and Organizational, providing a clear framework for decision-making.

3.1.1 Economic Feasibility

Economic feasibility focuses on the cost-effectiveness of implementing the attendance system. The project leverages widely available hardware and free or low-cost software resources, such as OpenCV and Python libraries, reducing the need for costly infrastructure. The system is designed to work on existing computing resources with minimal hardware upgrades, such as a standard computer setup and a high-resolution camera, making it accessible to educational institutions of varying sizes. The primary cost is related to initial setup and training, which is a one-time investment, with low operational costs thereafter. This ensures that the system remains financially viable for institutions over time.

3.1.2 Technical Feasibility

From a technical standpoint, the integration of OpenCV with LBPH for real-time facial recognition ensures high accuracy in identifying students. The system operates efficiently in real-time, using Python for image processing and GUI management. The use of CSV files for data storage ensures that the system remains lightweight and simple to maintain. Additionally, the project’s reliance on widely adopted tools like Pandas for data handling and Tkinter for the interface makes it easy to integrate into existing infrastructure. The hardware and software requirements are within reach for most institutions, ensuring that the system is technically feasible to deploy and maintain.

However, technical limitations could arise from the system's dependency on lighting conditions and camera quality for facial recognition accuracy. Institutions must ensure that proper hardware is in place, such as high-resolution cameras, to optimize the system’s performance. Training staff on how to operate the system and troubleshoot common technical issues will also be critical for its success.

3.1.3 Social Feasibility

Social feasibility assesses how well the system will be accepted by its users, particularly teachers and administrators. The user-friendly interface developed with Tkinter makes the system easy to operate, reducing the learning curve for non-technical users. By automating attendance tracking, the system removes the burden of manual attendance marking, saving time for teachers and ensuring accuracy in record-keeping.

Feedback from initial users is crucial to ensure that the system meets their needs and expectations. Training sessions will be conducted to familiarize users with the system, ensuring they feel comfortable and confident in its use. A user-centric approach during deployment and support will increase the system’s acceptance and effectiveness in the long term.

3.1.4 Operational Feasibility

Operationally, the project is designed to run smoothly in a real-world educational environment. By using OpenCV’s face detection capabilities and LBPH for recognition, the system ensures efficient and reliable attendance monitoring. The step-by-step configuration process for facial recognition, data handling, and attendance logging is well-documented, ensuring minimal disruption during implementation.
Real-time attendance tracking and seamless data handling through Pandas and CSV ensure that the system is both scalable and maintainable. Educational institutions can rely on this system to handle large class sizes and multiple sessions without performance issues. The Tkinter GUI provides an intuitive platform for managing subjects and monitoring attendance, ensuring that administrators can easily oversee the system's operations.

3.1.5 Organizational Feasibility

The organizational feasibility is supported by the system’s adaptability to different institutional settings. The project’s design allows for scalability, making it suitable for both small and large institutions. With proper role-based access control and database management, the system aligns with the organizational goals of improving attendance management and reducing administrative overhead. The system’s emphasis on automation and real-time data processing fits well with the objective of modernizing institutional processes, enhancing overall efficiency, and fostering organizational readiness for digital transformation.

3.2 Existing System and Its Drawbacks

In traditional attendance systems, manual methods such as roll calls or paper-based registers are commonly used to record student attendance. These methods are not only time-consuming but also prone to human error, leading to inaccurate records. Teachers are required to manually enter data into a central system, which introduces further delays and inconsistencies. Moreover, there is no real-time tracking or efficient reporting of student attendance data.

Existing digital systems that use RFID cards or biometric fingerprint scanning partially automate the process, but they come with their own set of challenges. RFID-based systems rely on students carrying cards, which can be lost or swapped. Biometric systems like fingerprint scanners can face issues with hygiene, especially in a post-pandemic world, as well as being susceptible to technical failures if the fingerprint sensor malfunctions.

Drawbacks:

- Time-consuming manual attendance marking.
- Prone to human error, leading to incorrect records.
- Lack of real-time attendance tracking and reporting.
- RFID systems can be manipulated or hindered by lost cards.
- Biometric systems can face hygiene concerns and sensor malfunctions.

3.3 Proposed System and Its Benefits

The proposed system leverages real-time face recognition using OpenCV’s Haar Cascade Classifier and LBPH (Local Binary Patterns Histograms) algorithm to automatically mark student attendance. This system eliminates the need for manual attendance marking and provides a highly accurate, automated solution.

The system works by capturing the students' faces in real-time through a camera, which is then processed by the facial recognition module to identify each student. Once identified, the attendance is marked in a CSV file, allowing for easy data handling and report generation.

Key Benefits:

- **Automated Attendance:** The system automatically marks attendance, saving time and eliminating human errors.
- **Real-Time Tracking:** Attendance is tracked in real-time, allowing teachers to instantly access and verify records.
- **Accuracy:** The use of facial recognition ensures high accuracy in identifying students, reducing the possibility of manipulation.
- **Hands-Free Operation:** The system reduces physical contact, making it safer in terms of hygiene compared to biometric systems.
- **Scalability:** The system can be easily scaled to handle large numbers of students or multiple classrooms without performance degradation.
- **Data Management:** The attendance records are stored in CSV files, making it easy to generate reports and integrate with other systems.

3.4 Scope of the Project

The scope of this project is to design and implement a real-time attendance monitoring system using facial recognition technology. The system is intended for educational institutions that wish to streamline the attendance process, improve accuracy, and reduce the administrative burden on teachers and staff.

The project covers the following areas:

- **Face Detection and Recognition:** Developing a system that can accurately detect and recognize faces in real-time using a high-resolution camera.
- **Data Handling and Storage:** Efficiently storing attendance data using Pandas and CSV files for easy report generation and future analysis.
- **User Interface:** Building a user-friendly GUI with Tkinter, enabling teachers to monitor and manage attendance for different subjects and sessions.
- **Real-Time Alerts:** Providing instant feedback on the attendance status of students, ensuring timely updates for teachers.
- **Scalability:** Ensuring that the system can handle up to 50 students per session and can be adapted for larger classes or institutions.

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) for this project serves as a visual representation of the flow of information within the real-time attendance monitoring system using facial recognition technology. It illustrates how data is captured, processed, stored, and managed across various components of the system, including face detection and recognition, data handling and storage, GUI interaction, and report generation. By visualizing these interactions, the DFD provides a high-level understanding of how the system operates and how different modules collaborate to achieve the project's objectives.

In this project, the DFD outlines the flow of data starting from the camera, which captures the students' faces, to the facial recognition module that processes the image data. Once the student's face is identified, the attendance is automatically marked and stored in a CSV file using Pandas. The user interface, built with Tkinter, allows teachers to select subjects, view attendance records, and generate reports. Additionally, the DFD highlights how real-time alerts are triggered to notify teachers of attendance statuses.

Each entity in the DFD represents a distinct part of the system, demonstrating how inputs (captured images) are transformed into outputs (marked attendance) through defined processes such as facial recognition, data storage, and GUI-based interaction. The DFD not only aids in system design and development but also helps stakeholders understand the data handling processes within the facial recognition system.

By using the DFD, developers and project managers can identify potential bottlenecks, redundancies, or areas for optimization in the data flow. It serves as a communication tool among team members, ensuring everyone has a shared understanding of how data is processed and managed. The DFD is crucial to the project's success, as it aligns all components towards a common goal of delivering an efficient, automated, and accurate attendance monitoring system.

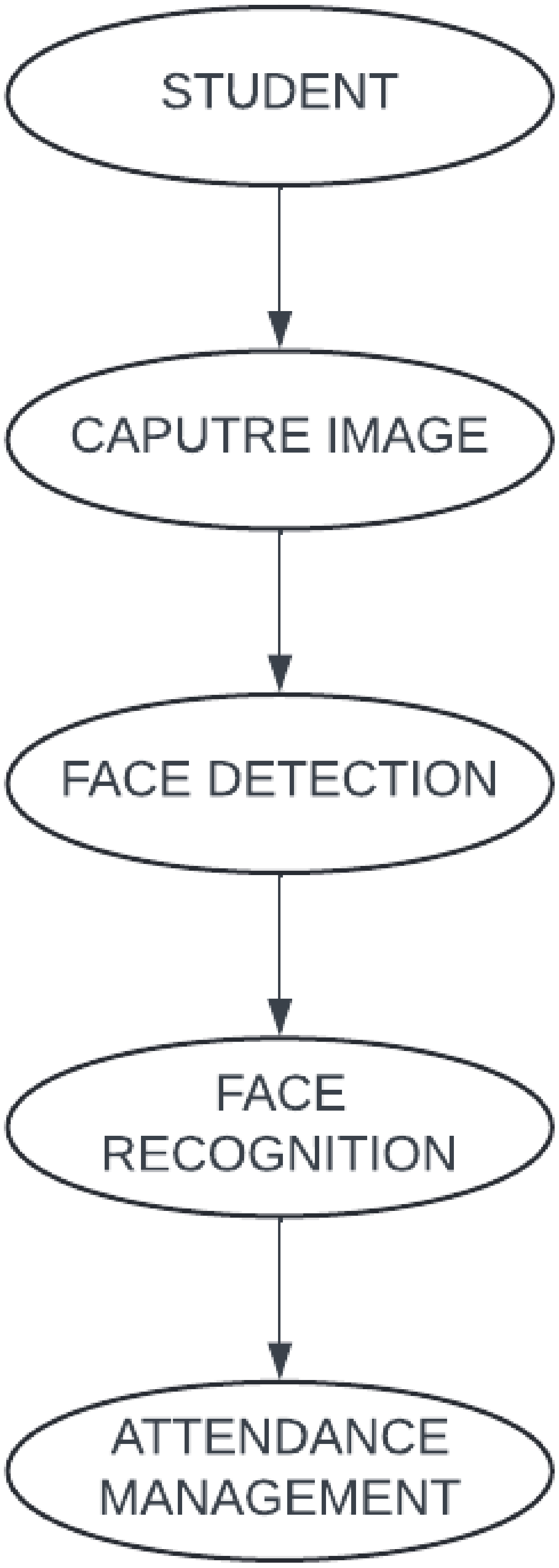

Fig 4.1: Representation of Data Flow Diagram

4.2 ARCHITECTURE DIAGRAM

The Architectural Diagram for this project provides a detailed visual representation of the system's structure, illustrating how various components interact within the facial recognition-based real-time attendance monitoring system. This diagram highlights the integration of essential technologies and software modules, including the facial recognition module, the data storage mechanism, the user interface (UI), and the report generation system. By showcasing the relationships and dependencies among these components, the architectural diagram serves as a foundational blueprint that guides both the development and implementation phases of the project.

In this project, the architectural diagram outlines the flow of data and control between the modules, emphasizing how the camera captures students' images, which are then processed by the facial recognition system. The attendance records are stored in a CSV file using Pandas, while the Tkinter-based UI allows teachers to select subjects and view real-time attendance status. The diagram also encapsulates the interactions between various software components, demonstrating how they collaborate to achieve efficient data processing, storage, and reporting.

Utilizing an architectural diagram enables stakeholders to quickly grasp the system's design, facilitating discussions around scalability, security, and operational efficiency. It serves as a vital communication tool for developers, project managers, and other stakeholders, ensuring that everyone involved has a unified understanding of the system's architecture. By providing a clear visual framework, the architectural diagram aids in identifying potential risks, optimizing resource allocation, and ensuring that all components align with the project's overall objectives. Ultimately, this diagram is crucial for the successful execution of the project, serving as a reference point throughout the development lifecycle.

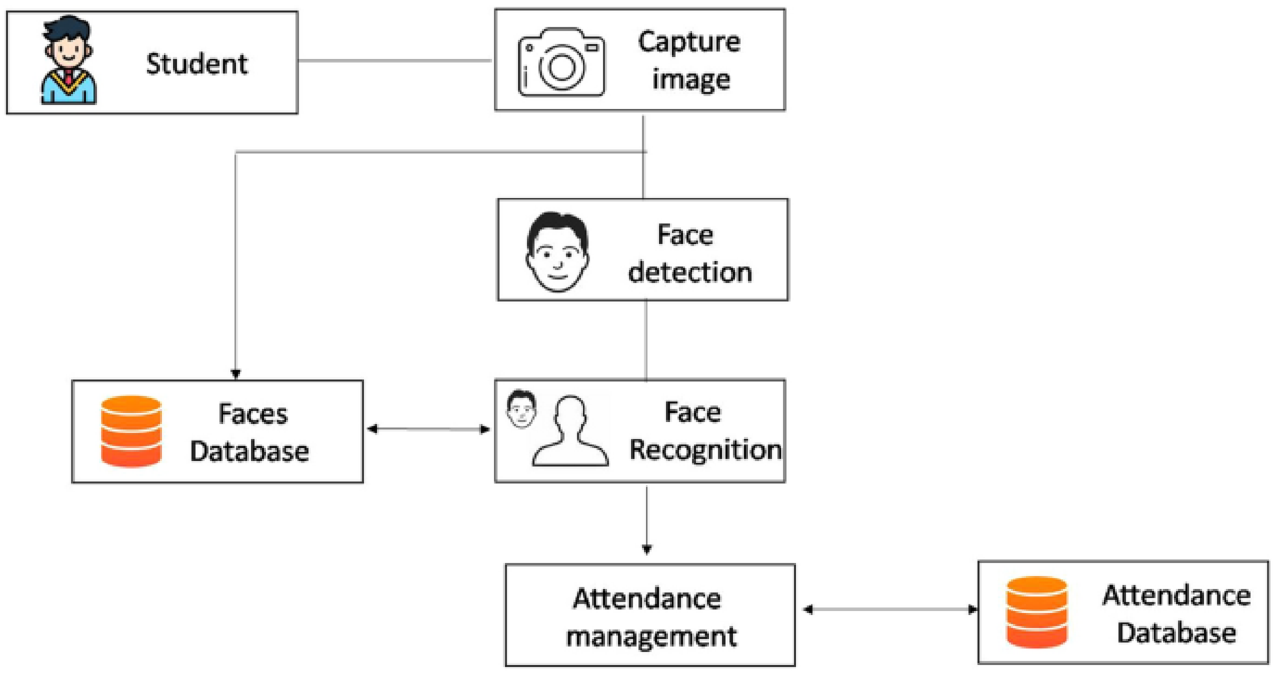

Fig 4.2: Architecture Diagram

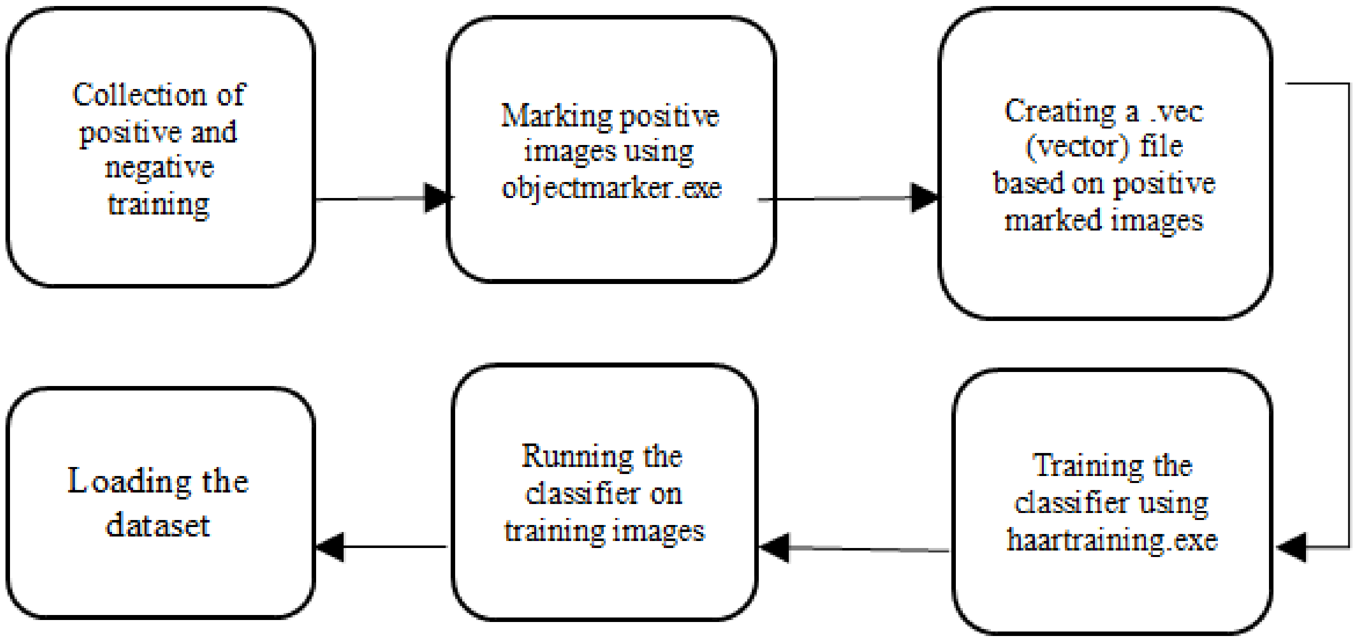
Collection of positive and negative training
Marking positive images using objectmarker.exe
Creating a .vec (vector) file based on positive marked images
Training the classifier using haartraining.exe
Running the classifier on training images
Loading the dataset

Fig 4.3: Training Process in Haar Cascade

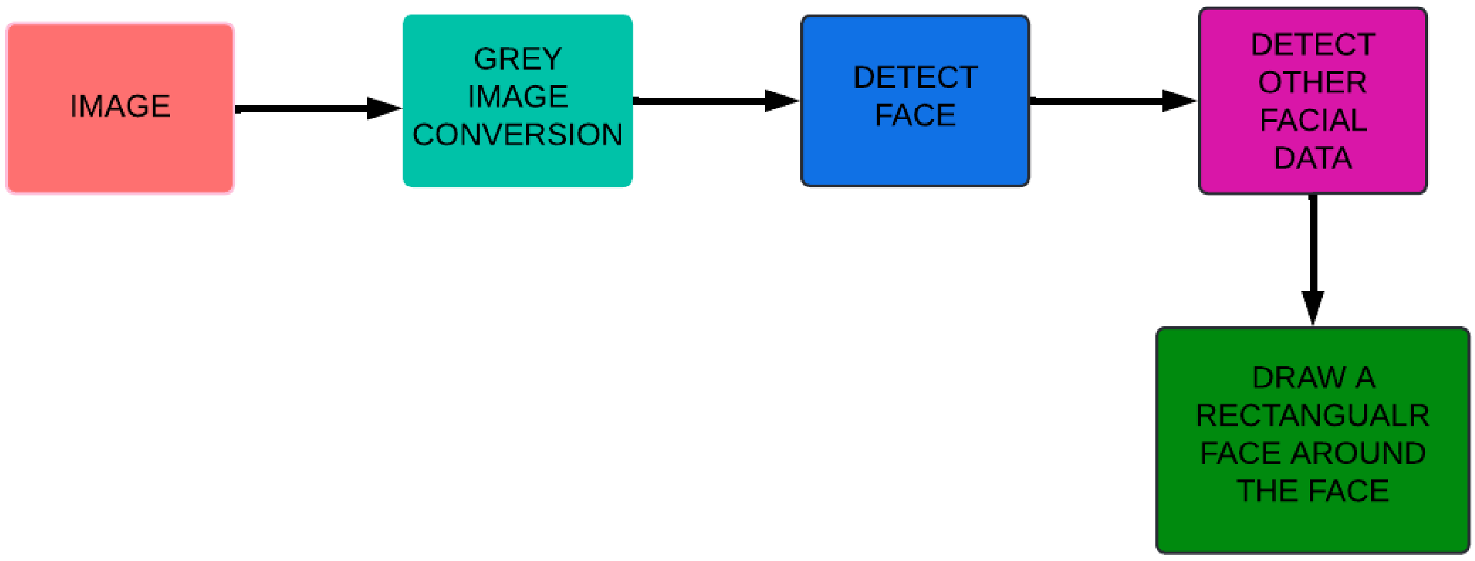

Fig 4.4 Haar Cascade Classifier workflow

PROJECT DESCRIPTION

5.1 OBJECTIVE

The primary objective of this project is to implement a facial recognition-based real-time attendance monitoring system that enhances the efficiency and accuracy of attendance management in educational institutions. This involves utilizing advanced computer vision techniques to automate the process of capturing and recognizing student faces, thereby eliminating the need for manual attendance tracking.

The project aims to create a user-friendly interface using Tkinter that allows educators to seamlessly select subjects and view real-time attendance data. By integrating a reliable image processing module with a well-structured database, the system will ensure accurate attendance records are maintained. The project seeks to improve operational efficiency by automating attendance marking, reducing human errors, and streamlining the reporting process.

A significant focus of this project is to ensure data security and privacy. Sensitive student information will be protected through secure data handling practices and compliance with relevant regulations. By implementing secure data storage solutions using Pandas and CSV files, the project aims to provide an environment where user data is safeguarded against unauthorized access.
Real-time processing is a cornerstone of this system, with the facial recognition module designed to quickly identify students as they enter the classroom. The system will be capable of handling multiple entries simultaneously, ensuring that attendance is marked promptly and accurately. This capability allows educators to manage their time more effectively, concentrating on teaching rather than administrative tasks.

Another key advantage of this project is its scalability and adaptability. The architecture will be designed to accommodate various classroom sizes and configurations, making it suitable for different educational settings. As the demand for remote and hybrid learning environments continues to grow, this system will be positioned to meet evolving educational needs by integrating with online platforms if required.

Overall, the project aspires to provide a robust solution that enhances the educational experience for both students and educators. By automating attendance management through facial recognition technology, the system aims to create a more engaging learning environment while ensuring accuracy, efficiency, and security.

5.2 MODULE DESCRIPTION

The facial recognition-based real-time attendance monitoring system is divided into several key modules, each responsible for specific functionalities that contribute to the overall efficiency and effectiveness of the project. Below is a detailed description of each module:

5.2.1 User Interface Module

- **Description:** This module provides a user-friendly interface for educators to interact with the system. Developed using Tkinter, it allows users to select subjects, initiate attendance marking, and view real-time attendance reports.
- **Key Features:**
  - Easy navigation for teachers.
  - Subject selection options.
  - Display of attendance status and history.

5.2.2 Facial Recognition Module

- **Description:** The core component of the system, this module utilizes OpenCV and pre-trained models to identify and recognize students' faces in real-time. It captures images from a webcam and processes them for attendance marking.
- **Key Features:**
  - Real-time face detection and recognition.
  - Ability to handle multiple students simultaneously.
  - High accuracy rates in face recognition.

5.2.3 Image Processing Module

- **Description:** This module processes images captured by the facial recognition module to enhance the recognition accuracy. It involves techniques such as resizing, normalization, and feature extraction using

Haar Cascade Classifier and LBPH (Local Binary Pattern Histograms) algorithms.

- **Key Features:**
    - Image preprocessing for improved recognition.
    - Conversion of images into suitable formats for analysis.
    - Efficient feature extraction to optimize recognition speed.

5.2.4 Database Management Module

- **Description:** This module handles the storage, retrieval, and management of attendance data. It uses Pandas to create and manipulate CSV files that maintain records of attendance, including timestamps and student details.
- **Key Features:**
    - Secure storage of attendance records.
    - Efficient data retrieval for reporting.
    - Ability to append new attendance records dynamically.

5.2.5 Reporting Module

- **Description:** This module generates attendance reports based on the data collected and stored in the database. It allows educators to view attendance patterns, analyze student presence, and export reports as needed.
- **Key Features:**
    - Generation of daily, weekly, and monthly attendance reports.
    - Visualization of attendance data for easy analysis.
    - Export options for offline review and record-keeping.

5.2.6 Security and Privacy Module

- **Description:** This module ensures the security and privacy of student data throughout the system. It incorporates measures to protect sensitive information from unauthorized access and ensures compliance with data protection regulations.
- **Key Features:**
    - Secure handling of student images and attendance records.
    - Access control mechanisms for user permissions.
    - Encryption of sensitive data to prevent breaches.

5.2.7 Integration Module

- **Description:** This module facilitates the seamless integration of all other modules, ensuring smooth communication and functionality across the system. It coordinates the flow of data and actions between the user interface, recognition, and database management.
- **Key Features:**
  - Coordination of module interactions for efficient operations.
  - Error handling and notifications for operational issues.
  - Real-time updates and synchronization of attendance data.

5.3 IMPLEMENTATION

The implementation of the facial recognition-based real-time attendance monitoring system involves a series of systematic steps aimed at ensuring seamless integration and functionality. Initially, the focus is on defining clear objectives for utilizing technologies such as OpenCV and Tkinter, along with database management using Pandas. Each component is evaluated for its role in supporting the overarching goal of automating attendance marking through facial recognition.

The first critical step is configuring the facial recognition module. This involves setting up the OpenCV environment and training the model using a collection of student images. The Haar Cascade Classifier and Local Binary Pattern Histograms (LBPH) algorithms are employed to enable accurate face detection and recognition. The system is tested with various lighting conditions and angles to ensure robustness and reliability.

Next, the development of the user interface module takes center stage. Utilizing Tkinter, the interface is designed to be user-friendly, allowing educators to select subjects, initiate attendance marking, and view real-time attendance reports. The integration of the user interface with the facial recognition module is carefully orchestrated to facilitate a seamless experience, where educators can easily access the functionalities, they need.

Following the user interface development, the database management module is established to handle the storage and retrieval of attendance data. Pandas is employed to create and manipulate CSV files that maintain records of student attendance, ensuring that data is captured accurately during each session. The system is designed to append new records dynamically, allowing for continuous operation without data loss.

In parallel, the reporting module is developed to generate attendance reports based on the data collected. This module is integrated with the database to provide educators with insights into attendance patterns, enabling effective monitoring and analysis of student presence. Visualization techniques are implemented to enhance the clarity and accessibility of the reports.

Finally, the security and privacy module are integrated to ensure the protection of sensitive student data throughout the system. Access control mechanisms are established to limit user permissions and safeguard against unauthorized access. Additionally, encryption techniques are employed to protect student images and attendance records.

The implementation concludes with thorough testing and validation of all modules to ensure they operate seamlessly together. This includes unit testing for individual components and integration testing for the entire system. Documentation is prepared to guide users through the system's functionalities, and training sessions are conducted to promote user acceptance and proficiency.
By following this structured implementation approach, the project effectively leverages advanced technologies to build a robust, efficient, and secure system for real-time attendance monitoring through facial recognition.

5.4 MAINTENANCE

The maintenance of the facial recognition-based real-time attendance monitoring system is crucial for ensuring the continued efficiency, security, and reliability of the integrated components. A structured maintenance plan is essential for addressing any issues that arise, implementing updates, and optimizing performance. Below are key components of the maintenance strategy:

Maintenance Strategy Key Points:

Regular Monitoring and Performance Analysis:

- Utilize logging mechanisms within the application to continuously monitor the performance of the facial recognition module and attendance tracking functionalities.
- Set up alerts for anomalies or performance degradation, such as issues with face detection or attendance marking accuracy.
- Conduct periodic performance reviews focusing on metrics like processing speed, accuracy rates, and resource utilization.

Security Audits and Data Management:

- Perform regular security audits of data access controls to ensure alignment with best practices for protecting sensitive student information.
- Update security protocols as necessary to reflect changes in project requirements or emerging threats.
- Monitor for unauthorized access attempts to the system and take corrective actions promptly.

Backup and Data Recovery:

- Implement regular backup procedures for attendance records and student data.
- Utilize versioning for maintaining historical records, ensuring data can be recovered if needed.
- Develop a disaster recovery plan outlining steps for restoring the system in case of data loss or corruption, including recovery point objectives (RPO) and recovery time objectives (RTO).

Updates and Optimizations:

- Stay informed about updates to OpenCV and Tkinter that may enhance performance or security features.
- Regularly review best practices for implementing machine learning models to optimize the accuracy and speed of the facial recognition system.
- Refactor and optimize the application code based on performance data and user feedback.

User Support and Training:

- Establish a support channel for ongoing user assistance to resolve any issues or concerns users may encounter.
- Regularly gather user feedback for continuous improvement, helping to identify areas where the system may be enhanced.
- Conduct periodic training sessions to familiarize users with new features, updates, and best practices for using the system effectively.

Documentation Maintenance:

- Keep project documentation up to date, including architecture diagrams, user manuals, and operational procedures.
- Document all changes made to the system, ensuring a clear history of the project's evolution and facilitating future maintenance efforts.

SYSTEM TESTING

6.1 TESTING DEFINITION

The process of identifying and rectifying errors in a facial recognition-based real-time attendance monitoring system is referred to as software testing. This method ensures that the application operates according to the end user's requirements, providing accurate and reliable attendance tracking.

6.2 TESTING OBJECTIVE

The objective of testing this application is to uncover any potential errors and eliminate them to guarantee the system functions as intended. For a project involving facial recognition technology and event-driven architectures, various types of testing are essential for ensuring reliability, performance, and security.

Unit Testing: This involves testing individual components of the facial recognition system, such as specific algorithms for face detection and recognition. By isolating and testing these core functions, developers can ensure that each module behaves correctly and meets the required accuracy levels.

Integration Testing: This type of testing focuses on validating the seamless integration of the facial recognition module with other system components, such as the user interface and the database for storing attendance records. For example, testing whether a successfully recognized face triggers the correct attendance entry and updates the database accordingly.

End-to-End Testing: This testing simulates real-world scenarios to verify that the entire workflow operates smoothly—from capturing images through the webcam to recording attendance in the system. This ensures that all components, including the facial recognition logic, database interactions, and user notifications, function together seamlessly.

Performance Testing: This evaluates how the system performs under various conditions, such as during peak usage times when multiple students' faces are being captured and processed simultaneously. It ensures that the system can scale effectively and maintains a high level of performance without degradation.

Security Testing: This assesses the security measures in place, including access controls for sensitive student data and the encryption of attendance records. Security testing ensures compliance with best practices, protecting against unauthorized access and potential vulnerabilities**.**

6.3 TYPES OF TESTING

6.3.1 Unit Testing

The goal of unit testing is to validate individual components of the facial recognition system, ensuring they operate as expected in isolation. This includes testing internal logic, error handling, and return values of functions responsible for tasks such as face detection and attendance marking. Tools like Python's unit test framework and mock libraries can be utilized for running these tests. Unit testing plays a crucial role in identifying bugs early in the development process, thereby reducing the risk of errors in the production environment. By isolating each function, developers can focus on verifying specific behaviors without interference from external services. This targeted approach ensures that even complex functionalities perform reliably under various conditions. Additionally, unit testing helps maintain code quality and improves the overall stability of the system. Regularly running these tests, especially in a CI/CD pipeline, ensures that any changes or updates to the code do not introduce new issues.

6.3.2 Integration Testing

The objective of integration testing is to ensure that all integrated components, such as the facial recognition module, database interactions, and user notifications, work together seamlessly. This involves testing the end-to-end data flow and interactions between various components, including how the system processes uploaded images and records attendance. Tools like Postman (for API testing) and Python's requests library can be employed for this purpose. Integration tests are designed to validate that integrated software components function correctly as a unified system. This testing is event-driven and focuses on the basic outcomes of interactions between components, demonstrating that the components, although tested individually, work correctly in combination.

6.3.3 Functional Testing

Functional testing validates that the facial recognition system meets its functional requirements. This includes verifying that facial recognition triggers work correctly, database updates are made as expected, and notifications are sent to users. Testing tools such as automated scripts and manual testing techniques are utilized to assess system functionality against predefined criteria. The organization and preparation of functional tests focus on requirements, key functions, and special test cases. In addition, systematic coverage is required to identify business process flows, data fields, and the integrity of successive processes. Before functional testing is complete, additional tests are identified, and the effectiveness of existing tests is evaluated.

6.3.4 Security Testing

Security testing ensures that the facial recognition system adheres to security standards, protecting it from unauthorized access and vulnerabilities. This involves reviewing access controls for sensitive data, verifying secure data transmission, and ensuring that best practices are followed in implementing security measures. Tools such as static analysis tools and penetration testing frameworks can be employed to assess the security posture of the application and identify potential vulnerabilities.

6.3.5 White Box Testing

White box testing is conducted with an understanding of the inner workings and structure of the facial recognition system. This method allows testers to verify the accuracy of algorithms, data processing, and internal logic, ensuring that the software performs as intended.

6.3.6 Black Box Testing

Black box testing involves evaluating the facial recognition system without knowledge of its internal workings or structure. Testers provide inputs (such as uploaded images) and assess the outputs (attendance records), focusing on whether the system meets functional requirements. This testing method helps ensure that the system operates correctly from the end user's perspective.

6.3.8 Unit Testing

Unit testing is a fundamental aspect of the software development lifecycle, focusing on verifying the functionality of individual components within the facial recognition attendance monitoring system. This phase aims to ensure that specific methods and logic in the facial recognition module, database interactions, and notification systems work as intended in isolation.

Test Strategy and Approach

For this project, manual field testing will be conducted alongside detailed functional tests to ensure the correct operation of the facial recognition system across all integrated services, including the facial recognition module, database interactions, and user notifications. Emphasis will be placed on validating proper integration, security, and real-time data processing within the application.

Test Objectives

- Ensure all input fields function correctly and adhere to specified formats, particularly for image uploads and attendance records.
- Verify that event triggers for facial recognition processing activate appropriately based on user interactions.
- Confirm that system responses, including notifications sent to users, occur without delay.

Features to be Tested

- Validate that all input fields follow the correct data format, such as image file types and user identifiers.
- Ensure no duplicate inputs are accepted, particularly for critical actions such as marking attendance.
- Verify that all service integrations, including the processing of uploaded images and the generation of attendance records, work as expected.

6.3.9 Integration Testing

Integration testing will focus on verifying the interactions among the AWS services utilized in the facial recognition attendance monitoring system. This testing will cover the end-to-end data flow, starting from an event in S3 that triggers a Lambda function, followed by sending notifications via SNS, and logging results in CloudWatch. This ensures that all AWS services work together seamlessly and helps identify any interface-related issues.

Test Results
All integration test cases executed successfully, with no defects encountered. The interaction between AWS services was smooth, with proper event-driven triggers functioning as expected.

6.3.10 Acceptance Testing

User Acceptance Testing (UAT) will involve validating that the entire facial recognition attendance monitoring system meets the project's requirements. The focus will be on end-user functionality, including response times for notifications, accurate logging of events in CloudWatch, and the proper functioning of IAM roles and permissions.

Test Results
All UAT test cases passed successfully, confirming that the system operates as expected without defects.

6.4 TEST CASES

S. No	Action	Expected Output	Actual Output	Result
1	Start the attendance system	System initializes without errors	System initialized successfully	PASS
2	Load student data from CSV file	Student data loads correctly	Student data loaded successfully	PASS
3	Take attendance for students	Attendance recorded for present students	Attendance recorded correctly	PASS
4	Display attendance summary	Summary shows total students and attendance	Summary displayed correctly	PASS
5	Search for a student by ID	Correct student details displayed	Correct student details displayed	PASS
6	Export attendance data to CSV	CSV file created with correct data	CSV file created successfully	PASS
7	Handle incorrect student ID input	Error message displayed	Error message displayed	PASS
8	Check attendance for a student	Attendance status (present/absent)	Attendance status displayed correctly	PASS
9	Update student information	Student information updated correctly	Student information updated successfully	PASS
10	Exit the attendance system	System closes without errors	System closed successfully	PASS

Fig 6.1: Representation of Test Cases

CONCLUSION

7.1 SUMMARY

The project aims to create a local attendance management system designed for efficient data processing and notifications. Central to the implementation is the use of event-driven functions that handle tasks such as marking attendance and notifying users. Access control is managed through appropriate security measures to ensure that only authorized users can interact with the system. Data is securely stored, with regular backups in place to maintain data integrity. The system manages notifications effectively, ensuring users receive timely updates regarding attendance status. Overall, the project focuses on building a scalable, reliable, and user-friendly system that meets operational requirements.

7.2 FUTURE ENHANCEMENTS

Future enhancements for this project may include integrating additional features such as a web or mobile application for improved accessibility and user experience. Implementing advanced analytics can provide deeper insights into attendance patterns and trends. Additionally, incorporating machine learning capabilities could enable predictive analytics based on user data, such as forecasting attendance rates. Improving security features through enhanced access controls will further safeguard sensitive information. Establishing a structured feedback mechanism will help continuously improve the system based on user insights, ensuring it evolves to meet changing needs.

APPENDIX

8.1 SCREENSHOTS

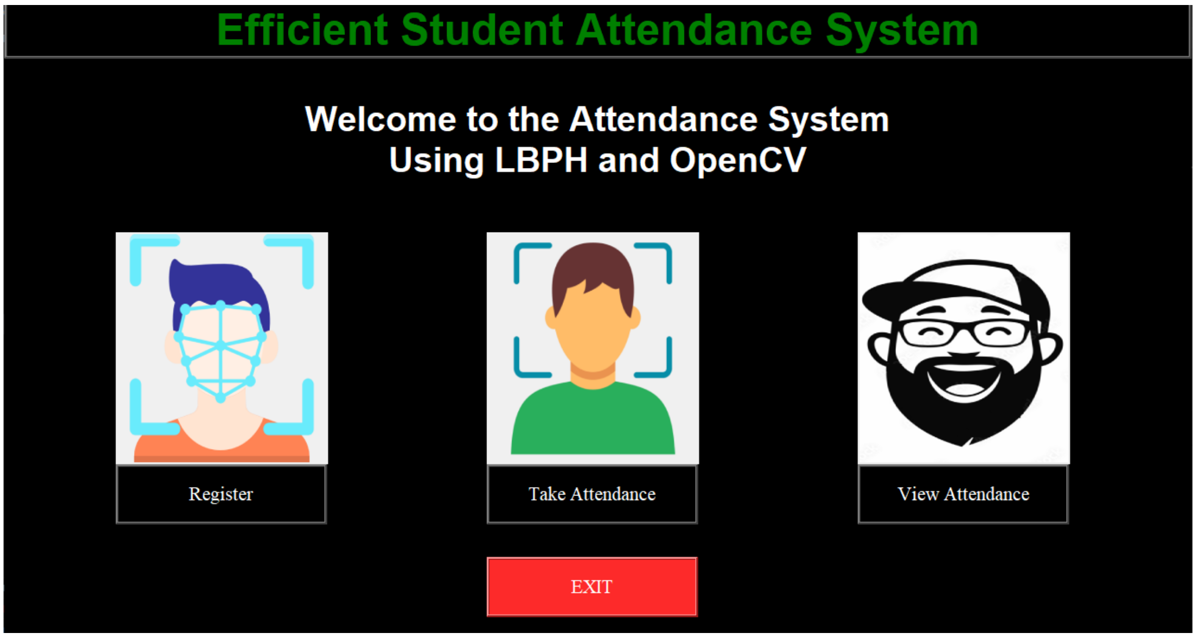
Efficient Student Attendance System
Welcome to the Attendance System
Using LBPH and OpenCV
Register
Take Attendance
View Attendance
EXIT

Fig 8.1 Home Page

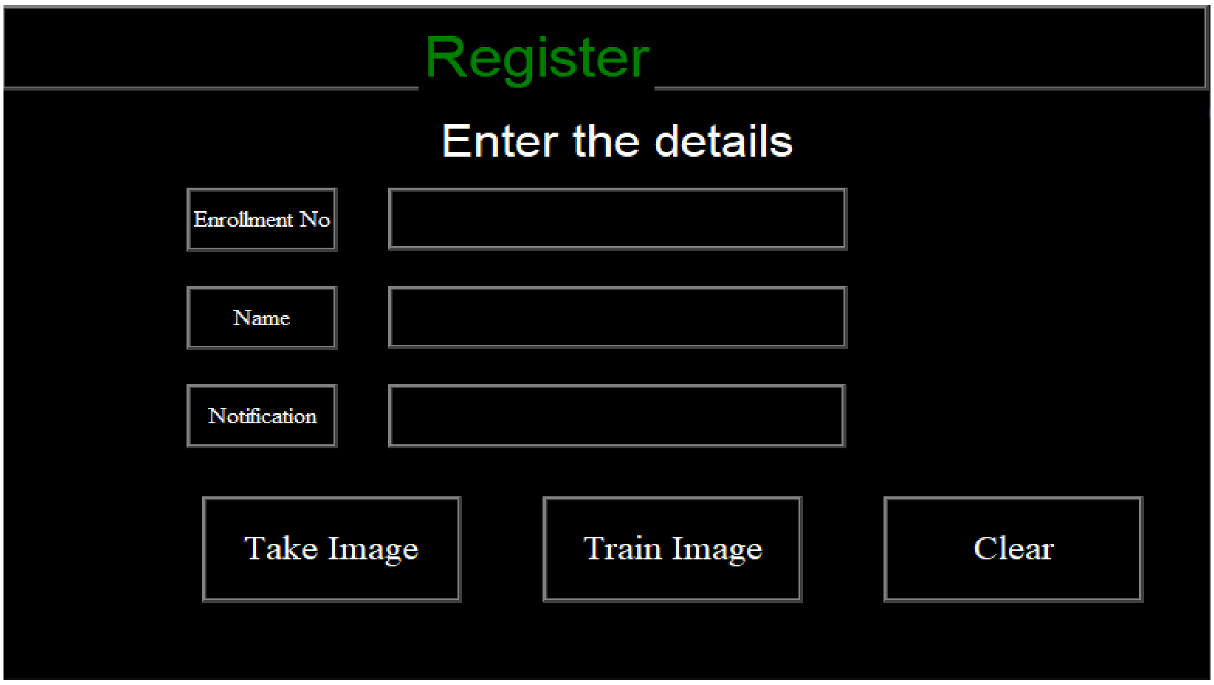

Fig 8.2: Registration Page

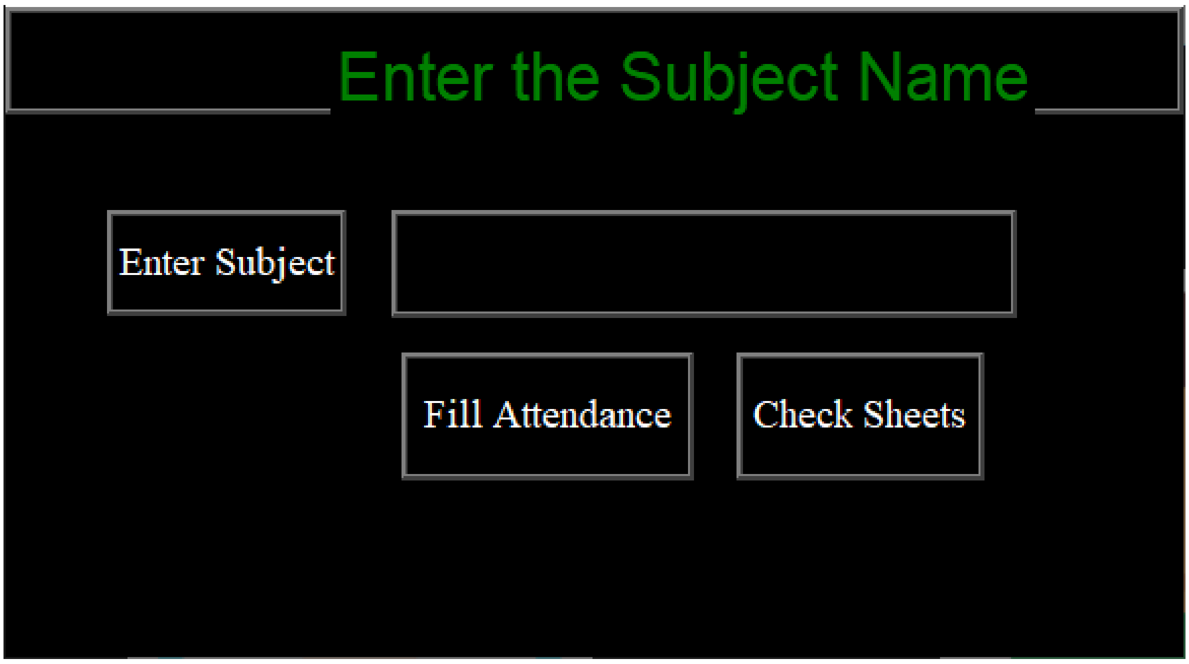

Fig 8.3: Take Attendance

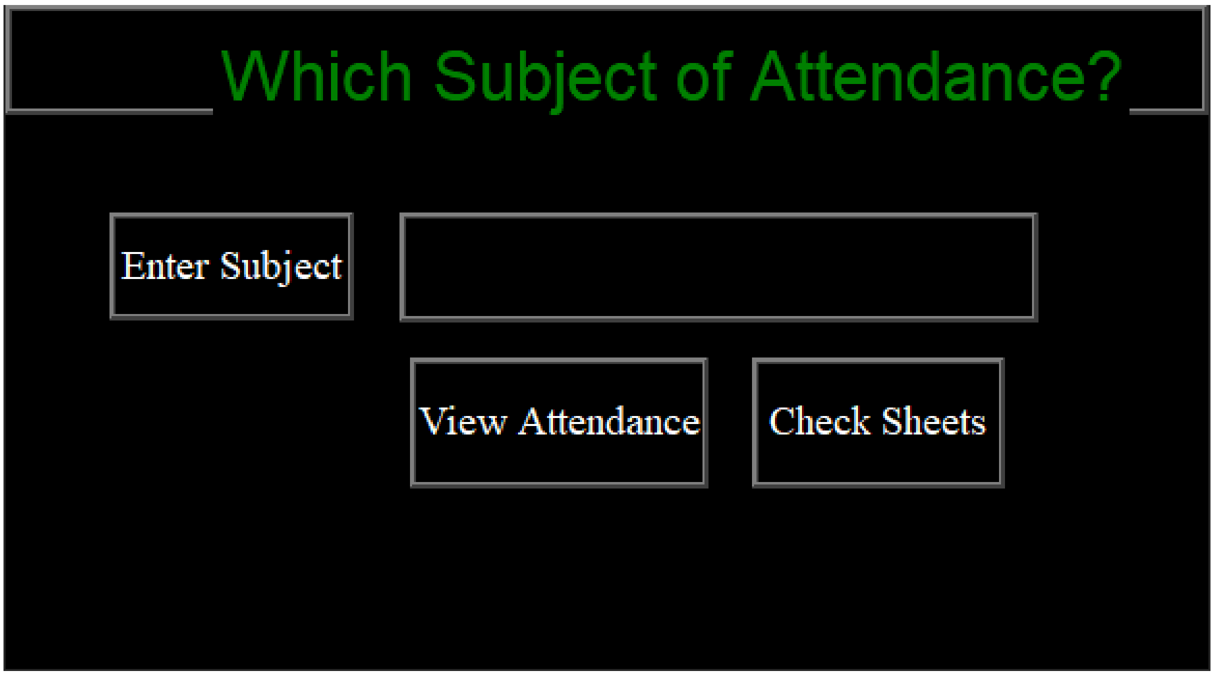

Fig 8.4: Check Attendance

8.2 CODING

attendance.py

```
import tkinter as tk
from tkinter import *
import os
import cv2
import shutil
import csv
import numpy as np
from PIL import ImageTk, Image
import pandas as pd
import datetime
import time
import tkinter.font as font
import pyttsx3
# Project modules
import show_attendance
import takeImage
import trainImage
import automaticAttedance
# Text to Speech Function
def text_to_speech(user_text):
    engine = pyttsx3.init()
    engine.say(user_text)
    engine.runAndWait()
# File paths
haarcasecade_path = "haarcascade_frontalface_default.xml"
trainimagelabel_path = "Trainner.yml"
trainimage_path = "TrainingImage"
if not os.path.exists(trainimage_path):
    os.makedirs(trainimage_path)
studentdetail_path = "/StudentDetails/studentdetails.csv"
attendance_path = "Attendance"
# Main window setup
window = tk.Tk()
window.title("Face recognizer")
window.geometry("1280x720")
window.configure(background="black")
# Function to destroy the warning screen
def del_sc1():
    sc1.destroy()
# Error message for name and number
def err_screen():
    global sc1
    sc1 = tk.Tk()
    sc1.geometry("400x110")
    sc1.iconbitmap("AMS.ico")
    sc1.title("Warning!!")
    sc1.configure(background="black")
    sc1.resizable(0, 0)
    tk.Label(
        sc1,
        text="Enrollment & Name required!!!",
        fg="yellow",
        bg="black",
        font=("times", 20, "bold"),
    ).pack()
    tk.Button(
        sc1,
        text="OK",
        command=del_sc1,
        fg="yellow",
        bg="black",
        width=9,
        height=1,
        activebackground="Red",
        font=("times", 20, "bold"),
    ).place(x=110, y=50)
# Function to validate entry (only digits allowed)
def testVal(inStr, acttyp):
    if acttyp == "1":  # insert
        if not inStr.isdigit():
            return False
    return True
# Load and resize logo
logo = Image.open("UI_Image/0001.png")
logo = logo.resize((50, 47), Image.Resampling.LANCZOS)
logo1 = ImageTk.PhotoImage(logo)
titl = tk.Label(window, bg="black", relief=RIDGE, bd=10, font=("arial", 35))
titl.pack(fill=X)
l1 = tk.Label(window, image=logo1, bg="black")
l1.place(x=470, y=10)
# Title and welcome message
titl = tk.Label(
    window, text="Smart College!!", bg="black", fg="green", font=("arial", 27),
)
titl.place(x=525, y=12)
a = tk.Label(
    window,
    text="Welcome to the Face Recognition Based\nAttendance Management System",
    bg="black",
    fg="yellow",
    bd=10,
    font=("arial", 35),
)
a.pack()
# Load and place images
def load_and_place_image(image_path, x, y):
    img = Image.open(image_path)
    img = ImageTk.PhotoImage(img)
    label = tk.Label(window, image=img)
    label.image = img
    label.place(x=x, y=y)
load_and_place_image("UI_Image/register.png", 100, 270)
load_and_place_image("UI_Image/attendance.png", 980, 270)
load_and_place_image("UI_Image/verifyy.png", 600, 270)
# Function to open the image registration UI
def TakeImageUI():
    ImageUI = tk.Tk()
    ImageUI.title("Take Student Image..")
    ImageUI.geometry("780x480")
    ImageUI.configure(background="black")
    ImageUI.resizable(0, 0)
    titl = tk.Label(ImageUI, bg="black", relief=RIDGE, bd=10, font=("arial", 35))
    titl.pack(fill=X)
    titl = tk.Label(
        ImageUI, text="Register Your Face", bg="black", fg="green", font=("arial", 30),
    )
    titl.place(x=270, y=12)
    a = tk.Label(
        ImageUI,
        text="Enter the details",
        bg="black",
        fg="yellow",
        bd=10,
        font=("arial", 24),
    )
    a.place(x=280, y=75)

    lbl1 = tk.Label(
        ImageUI,
        text="Enrollment No",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl1.place(x=120, y=130)
    txt1 = tk.Entry(
        ImageUI,
        width=17,
        bd=5,
        validate="key",
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 25, "bold"),
    )
    txt1.place(x=250, y=130)
    txt1["validatecommand"] = (txt1.register(testVal), "%P", "%d")

    lbl2 = tk.Label(
        ImageUI,
        text="Name",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl2.place(x=120, y=200)
    txt2 = tk.Entry(
        ImageUI,
        width=17,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 25, "bold"),
    )
    txt2.place(x=250, y=200)
    lbl3 = tk.Label(
        ImageUI,
        text="Notification",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl3.place(x=120, y=270)
    message = tk.Label(
        ImageUI,
        text="",
        width=32,
        height=2,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 12, "bold"),
    )
    message.place(x=250, y=270)
    def take_image():
        l1 = txt1.get()
        l2 = txt2.get()
        takeImage.TakeImage(
            l1,
            l2,
            haarcasecade_path,
            trainimage_path,
            message,
            err_screen,
            text_to_speech,
        )
        txt1.delete(0, "end")
        txt2.delete(0, "end")
    takeImg = tk.Button(
        ImageUI,
        text="Take Image",
        command=take_image,
        bd=10,
        font=("times new roman", 18),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    takeImg.place(x=130, y=350)
    def train_image():
        trainImage.TrainImage(
            haarcasecade_path,
            trainimage_path,
            trainimagelabel_path,
            message,
            text_to_speech,
        )
    trainImg = tk.Button(
        ImageUI,
        text="Train Image",
        command=train_image,
        bd=10,
        font=("times new roman", 18),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    trainImg.place(x=360, y=350)# Buttons on the main window
def create_button(text, command, x, y):
    button = tk.Button(
        window,
        text=text,
        command=command,
        bd=10,
        font=("times new roman", 16),
        bg="black",
        fg="yellow",
        height=2,
        width=17,
    )
    button.place(x=x, y=y)
create_button("Register a new student", TakeImageUI, 100, 520)
create_button("Take Attendance", lambda:
automaticAttedance.subjectChoose(text_to_speech), 600, 520)
create_button("View Attendance", lambda:
show_attendance.subjectchoose(text_to_speech), 1000, 520)
create_button("EXIT", window.destroy, 600, 660)
window.mainloop()
```

automaticAttendance.py

```
import tkinter as tk
```

```
from tkinter import *
import os, cv2
import shutil
import csv
import numpy as np
from PIL import ImageTk, Image
import pandas as pd
import datetime
import time
import tkinter.ttk as tkk
import tkinter.font as font
haarcasecade_path = "haarcascade_frontalface_default.xml"
trainimagelabel_path = "Trainner.yml"
trainimage_path = "TrainingImage"
studentdetail_path = "StudentDetails\\studentdetails.csv"
attendance_path = "Attendance"
def subjectChoose(text_to_speech):
    def FillAttendance():
        sub = tx.get()
        now = time.time()
        future = now + 20
        if sub == "":
            t = "Please enter the subject name!!!"
            text_to_speech(t)
        else:
            try:
                recognizer = cv2.face.LBPHFaceRecognizer_create()
                if not os.path.isfile(trainimagelabel_path):
                    e = "Model not found, please train the model first."
                    Notifica.configure(
                        text=e,
                        bg="black",
                        fg="yellow",
                        width=33,
                        font=("times", 15, "bold"),
                    )
                    Notifica.place(x=20, y=250)
                    text_to_speech(e)
                    return
                recognizer.read(trainimagelabel_path)
                facecasCade = cv2.CascadeClassifier(haarcasecade_path)
                df = pd.read_csv(studentdetail_path)
                cam = cv2.VideoCapture(0)
                font = cv2.FONT_HERSHEY_SIMPLEX
                col_names = ["Enrollment", "Name"]
                attendance = pd.DataFrame(columns=col_names)
                while True:
                    ret, im = cam.read()
                    if not ret:
                        print("Failed to grab frame")
                        break
                    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
                    faces = facecasCade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
                    print(f"Faces detected: {len(faces)}")
                    for (x, y, w, h) in faces:
                        Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
                        print(f"Confidence: {conf}")
                        if conf < 80:
                            aa = df.loc[df["Enrollment"] == Id]["Name"].values
                            tt = f"{Id}-{aa}"
                            attendance.loc[len(attendance)] = [Id, aa]
                            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 4)
                            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
                    cv2.imshow("Filling Attendance...", im)
                    key = cv2.waitKey(30) & 0xFF
                    if key == 27:  # Press 'Esc' to exit
                        break
                attendance = attendance.drop_duplicates(["Enrollment"],
keep="first")
                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(":")
                path = os.path.join(attendance_path, sub)
                if not os.path.exists(path):
                    os.makedirs(path)
                fileName = f"{path}/{sub}_{date}_{Hour}-{Minute}-{Second}.csv"
                attendance.to_csv(fileName, index=False)
                m = "Attendance Filled Successfully for " + sub
                Notifica.configure(
                    text=m,
                    bg="black",
                    fg="yellow",
                    width=33,
                    relief=RIDGE,
                    bd=5,
                    font=("times", 15, "bold"),
                )
                text_to_speech(m)
                Notifica.place(x=20, y=250)
                cam.release()
                cv2.destroyAllWindows()
                root = tk.Tk()
                root.title("Attendance of " + sub)
                root.configure(background="black")
                with open(fileName, newline="") as file:
                    reader = csv.reader(file)
                    for r, col in enumerate(reader):
                        for c, row in enumerate(col):
                            label = tk.Label(
                                root,
                                width=10,
                                height=1,
                                fg="yellow",
                                font=("times", 15, " bold "),
                                bg="black",
                                text=row,
                                relief=tk.RIDGE,
                            )
                            label.grid(row=r, column=c)
                root.mainloop()
            except Exception as e:
                print(f"Error: {e}")
                f = "No Face found for attendance"
```

```
            text_to_speech(f)
            cv2.destroyAllWindows()
subject = Tk()
subject.title("Subject...")
subject.geometry("580x320")
subject.resizable(0, 0)
subject.configure(background="black")
titl = tk.Label(subject, bg="black", relief=RIDGE, bd=10, font=("arial", 30))
titl.pack(fill=X)
titl = tk.Label(
    subject,
    text="Enter the Subject Name",
    bg="black",
    fg="green",
    font=("arial", 25),
)
titl.place(x=160, y=12)
Notifica = tk.Label(
    subject,
    text="Attendance filled Successfully",
    bg="yellow",
    fg="black",
    width=33,
    height=2,
    font=("times", 15, "bold"),
)
def Attf():
    sub = tx.get()
    if sub == "":
        t = "Please enter the subject name!!!"
        text_to_speech(t)
    else:
        os.startfile(f"Attendance\\{sub}")

attf = tk.Button(
    subject,
    text="Check Sheets",
    command=Attf,
    bd=7,
    font=("times new roman", 15),
    bg="black",
    fg="yellow",
    height=2,
    width=10,
    relief=RIDGE,
)
attf.place(x=360, y=170)

sub = tk.Label(
    subject,
    text="Enter Subject",
    width=10,
    height=2,
    bg="black",
    fg="yellow",
    bd=5,
    relief=RIDGE,
    font=("times new roman", 15),
)
sub.place(x=50, y=100)
```

```
    tx = tk.Entry(
        subject,
        width=15,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 30, "bold"),
    )
    tx.place(x=190, y=100)

    fill_a = tk.Button(
        subject,
        text="Fill Attendance",
        command=FillAttendance,
        bd=7,
        font=("times new roman", 15),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    fill_a.place(x=195, y=170)
    subject.mainloop()
```

showAttendance.py

```
import pandas as pd
from glob import glob
```

```
import os
import tkinter
import csv
import tkinter as tk
from tkinter import *

# Paths
attendance_path = "Attendance"

def subjectchoose(text_to_speech):
    def calculate_attendance():
        Subject = tx.get()
        if Subject == "":
            t = 'Please enter the subject name.'
            text_to_speech(t)
            return

        subject_directory = os.path.join(attendance_path, Subject)
        if not os.path.exists(subject_directory):
            t = f"No directory found for the subject: {Subject}"
            text_to_speech(t)
            return

        filenames = glob(os.path.join(subject_directory, f"{Subject}*.csv"))
        if len(filenames) == 0:
            t = f"No attendance files found for the subject: {Subject}"
            text_to_speech(t)
            return

        df = [pd.read_csv(f) for f in filenames]
        newdf = df[0]

        for i in range(1, len(df)):
            newdf = newdf.merge(df[i], how="outer")

        newdf.fillna(0, inplace=True)
        newdf["Attendance"] = 0

        for i in range(len(newdf)):
            newdf["Attendance"].iloc[i] = str(int(round(newdf.iloc[i, 2:-1].mean() * 100))) +
'%'

        newdf.to_csv("attendance.csv", index=False)

        root = tkinter.Tk()
        root.title("Attendance of " + Subject)
        root.configure(background="black")
        cs = os.path.join(subject_directory, "attendance.csv")

        with open(cs) as file:
            reader = csv.reader(file)
            r = 0

            for col in reader:
                c = 0
                for row in col:
                    label = tkinter.Label(
                        root,
                        width=10,
                        height=1,
                        fg="yellow",
                font=("times", 15, " bold "),
                bg="black",
                text=row,
                relief=tkinter.RIDGE,
            )
            label.grid(row=r, column=c)
            c += 1
        r += 1
    root.mainloop()
    print(newdf)

subject = Tk()
subject.title("Subject...")
subject.geometry("580x320")
subject.resizable(0, 0)
subject.configure(background="black")

titl = tk.Label(subject, bg="black", relief=RIDGE, bd=10, font=("arial", 30))
titl.pack(fill=X)
titl = tk.Label(
    subject,
    text="Which Subject of Attendance?",
    bg="black",
    fg="green",
    font=("arial", 25),
)
titl.place(x=100, y=12)

def Attf():
    sub = tx.get()
    if sub == "":
        t = "Please enter the subject name!!!"
        text_to_speech(t)
    else:
        os.startfile(os.path.join(attendance_path, sub))

attf = tk.Button(
    subject,
    text="Check Sheets",
    command=Attf,
    bd=7,
    font=("times new roman", 15),
    bg="black",
    fg="yellow",
    height=2,
    width=10,
    relief=RIDGE,
)
attf.place(x=360, y=170)

sub = tk.Label(
    subject,
    text="Enter Subject",
    width=10,
    height=2,
    bg="black",
    fg="yellow",
    bd=5,
    relief=RIDGE,
    font=("times new roman", 15),
)
sub.place(x=50, y=100)

tx = tk.Entry(
    subject,
    width=15,
    bd=5,
    bg="black",
    fg="yellow",
    relief=RIDGE,
    font=("times", 30, "bold"),
)
tx.place(x=190, y=100)

fill_a = tk.Button(
    subject,
    text="View Attendance",
    command=calculate_attendance,
    bd=7,
    font=("times new roman", 15),
    bg="black",
    fg="yellow",
    height=2,
    width=12,
    relief=RIDGE,
)
fill_a.place(x=195, y=170)
subject.mainloop()
```

takeImage.py

```
import csv
import os
import cv2
import numpy as np
import pandas as pd
import datetime
import time


# Take Image of user
def TakeImage(l1, l2, haarcasecade_path, trainimage_path, message, err_screen,
text_to_speech):
    if (l1 == "") and (l2 == ""):
        t = 'Please Enter your Enrollment Number and Name.'
        text_to_speech(t)
    elif l1 == '':
        t = 'Please Enter your Enrollment Number.'
        text_to_speech(t)
    elif l2 == "":
        t = 'Please Enter your Name.'
        text_to_speech(t)
    else:
        try:
            cam = cv2.VideoCapture(0)
            detector = cv2.CascadeClassifier(haarcasecade_path)
            Enrollment = l1
            Name = l2
            sampleNum = 0
            directory = Enrollment + "_" + Name
            path = os.path.join(trainimage_path, directory)
            os.makedirs(path, exist_ok=True)  # Use os.makedirs with exist_ok=True to avoid
FileExistsError
```

```
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)
        sampleNum += 1
        cv2.imwrite(
            os.path.join(path, f"{Name}_{Enrollment}_{sampleNum}.jpg"),  # Corrected file path
            gray[y: y + h, x: x + w]
        )
        cv2.imshow("Frame", img)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
    elif sampleNum > 50:
        break
cam.release()
cv2.destroyAllWindows()
row = [Enrollment, Name]
with open(
    "StudentDetails/studentdetails.csv",
    "a+",
) as csvFile:
    writer = csv.writer(csvFile, delimiter=",")
    writer.writerow(row)

res = f"Images Saved for ER No: {Enrollment} Name: {Name}"
message.configure(text=res)
text_to_speech(res)
```

```
        except FileExistsError:
            F = "Student Data already exists"
            text_to_speech(F)
```

trainImage.py

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


# Train Image
def TrainImage(haarcasecade_path, trainimage_path, trainimagelabel_path,
message,text_to_speech):
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier(haarcasecade_path)
    faces, Id = getImagesAndLables(trainimage_path)
    recognizer.train(faces, np.array(Id))
    recognizer.save(trainimagelabel_path)
    res = "Image Trained successfully"  # +",".join(str(f) for f in Id)
    message.configure(text=res)
    text_to_speech(res)


def getImagesAndLables(path):
    # imagePath = [os.path.join(path, f) for d in os.listdir(path) for f in d]
    newdir = [os.path.join(path, d) for d in os.listdir(path)]
    imagePath = [
        os.path.join(newdir[i], f)
        for i in range(len(newdir))
```

```
   for f in os.listdir(newdir[i])
]
faces = []
Ids = []
for imagePath in imagePath:
   pilImage = Image.open(imagePath).convert("L")
   imageNp = np.array(pilImage, "uint8")
   Id = int(os.path.split(imagePath)[-1].split("_")[1])
   faces.append(imageNp)
   Ids.append(Id)
return faces, Ids
```

BIBILIOGRAPHY AND REFERENCES

1. Md Sajid Akbar, Pronob Sarker, Ahmad Tamim Mansoor, Abu Musa Al Ashray, and Jia Uddin. Face recognition and RFID verified attendance system. In 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), pages 168–172. IEEE, 2018.
2. Gary Bradski and Adrian Kaehler. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, Inc., 2008.
3. Jason Brownlee. A gentle introduction to computer vision. Machine Learning Mastery, 2019.
4. Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. KNN model-based approach in classification. In OTM Confederated International Conferences on the Move to Meaningful Internet Systems, pages 986–996. Springer, 2003.
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.
6. Michael Herrmann. PyQt5 tutorial 2020: Create a GUI with Python and Qt.
7. Kennedy O. Okokpujie, Etinosa Noma-Osaghae, Olatunji J. Okesola, Samuel N. John, and Okonigene Robert. Design and implementation of a student attendance system using iris biometric recognition. In 2017 International Conference on Computational Science and Computational Intelligence (CSCI), pages 563–567. IEEE, 2017.
8. Sifatnur Rahman, Mahabur Rahman, Md Mijanur Rahman, et al. Automated student attendance system using fingerprint recognition. Edelweiss Applied Science and Technology, 1(2):90–94, 2018.
9. Hemantkumar Rathod, Yudhisthir Ware, Snehal Sane, Suresh Raulo, Vishal Pakhare, and Imdad A. Rizvi. Automated attendance system using machine learning approach. In 2017 International Conference on Nascent Technologies in Engineering (ICNTE), pages 1–5. IEEE, 2017.
10. Adrian Rosebrock. Face detection with OpenCV and deep learning. Web: https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning, 2018.
11. Adrian Rosebrock. Face recognition with OpenCV, Python, and deep learning. Web: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning, 2018.
12. Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 815–823, 2015.
13. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages I–I. IEEE, 2001.