

**NANDHA ENGINEERING
COLLEGEERODE–638052 (Autonomous)**

(Affiliated to Anna University, Chennai)



DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

FACE ATTENDANCE SYSTEM

PBL REPORT

22AIC09 – DATABASE DESIGN AND MANAGEMENT

Submitted by

REGISTER NUMBER	NAME
22AI001	AKASH A
22AI004	BHARANI S
22AI005	BHARANIDHARAN G
22AI022	KEERTHIVARSAN D

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

NANDHA ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

This is to certify that the project work entitled “FACE ATTENDANCESYSTEM” is the bonafide work of AKASH A (22AI001), BHARANI S (22AI004), BHARANIDHARAN G (22AI005), KEERTHIVARSAN D (22AI022) who carried out the work under my supervision.

Signature of the Supervisor Signature of the HOD

**Ms.S. SHANMUGA PRIYA,
Assistant Professor,
Department of AI&DS,
Nandha Engineering College,
Erode-638052.**

**Ms.M.PARVATHI,
Assistant Professor and Head,
Department of AI&DS,
Nandha Engineering College,
Erode-638052.**

Submitted for End semester PBL review held on _____

TABLE OF CONTENTS

CHAPTERNO	TITLE	PAGENO
1.	ABSTRACT	4
2.	SOFTWARE AND HARDWARE REQUIREMENTS	5
	Software Requirements	
	Hardware Requirements	
3.	DESCRIPTION OF THECONCEPTS RELATED TO SYLLABUS	6
4.	DIAGRAMMATIC REPRESENTATION RELATED TO SYLLABUS	9
	Flow chart	
	Use case diagram	
	Sequence diagram	10
	Activity diagram	11
5.	DESCRIPTION OF MODULES	12
6.	TESTING AND IMPLEMENTATION	15
7.	SCOPE FOR FUTURE ENHANCEMENT	17
8.	SOURCE CODE	18
9.	SCREENSHOTS OF THE PROJECT	20
10.	CONCLUSION	22
11.	REFERENCE	23

FACE ATTENDANCESYSTEM

ABSTRACT

In recent years, biometric identification systems have gained significant traction due to their accuracy and reliability in various applications, including security, access control, and attendance management. This project presents a Face Attendance System, leveraging advanced machine learning algorithms and computer vision techniques to automate the process of attendance marking. The system employs Local Binary Patterns Histograms (LBPH) for face recognition, ensuring high accuracy even under varying lighting conditions and facial expressions.

The core functionality of the Face Attendance System involves detecting and recognizing faces in real-time using a web-based interface. Users can access the system via a webcam, which captures their facial images. These images are then processed by a pre-trained LBPH recognizer that identifies the individual by comparing the captured image with a dataset of known faces. If a match is found with sufficient confidence, the system marks the user's attendance, recording the time and date.

This project is implemented using Python and OpenCV for image processing and facial recognition, combined with a Flask web server to manage the user interface and handle image uploads. The system is designed to be intuitive and user-friendly, featuring a simple landing page with options to start the webcam, capture an image, and display the recognition results.

SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements:

REQUIREMENT	USAGE
User Interface Design	HTML, CSS, JAVASCRIPT
Database	SQLite, MySQL
Libraries and Frameworks	OpenCV, Numpy, Flask, Pandas
Web Browser	Google Chrome, Firefox, Modern web browser

Hardware Requirements:

PC CONFIGURATION:

REQUIREMENTS	CONFIGURATION
Processor	Intel i5 or Equivalent
Random Access Memory(RAM)	8 GB (minimum)
Operating System(OS)	Windows, Linux, or macOS
System type	32 / 64-bit Operating System

DESCRIPTION OF THE CONCEPTS RELATED TO SYLLABUS:

The Face Attendance System leverages several fundamental concepts in computer vision, machine learning, and web development, each contributing to a comprehensive understanding of how such systems operate. Here's an overview of the key concepts typically covered in a related syllabus:

1. Image Processing: Understanding the basics of image processing, including grayscale conversion, histogram equalization, and edge detection, which are essential for preparing images for analysis.

2. Face Detection: Techniques for detecting faces in images and videos, primarily using Haar cascades. This involves learning how classifiers are trained to recognize face patterns and how they can be implemented using libraries like OpenCV.

3. Feature Extraction: The Learning about extracting features from images, such as Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG), which are used to represent faces in a way that is invariant to lighting and facial expressions.

4. Face Recognition: Introduction to face recognition algorithms like LBPH (Local Binary Patterns Histograms), Eigenfaces, and Fisherfaces. These methods involve training a model on a set of labeled images and using the model to predict the identity of new faces.

5. Model Training and Evaluation: Understanding the process of training machine learning models, evaluating their performance using metrics like accuracy and confidence scores, and tuning them for better performance.

6. Libraries and Tools: Familiarity with Python libraries such as OpenCV for computer vision tasks, NumPy for numerical computations, and Pandas for data handling.

7. Scripting and Automation: Writing scripts to automate the process of loading images, training models, and performing real-time face detection and recognition.

8. Frontend Development: Basics of HTML, CSS, and JavaScript for creating a user-friendly web interface. Understanding how to use the MediaDevices API to access the webcam and capture images.

9. Backend Development: Using Flask to develop a backend server that handles HTTP requests, processes images, and interacts with the face recognition model. Knowledge of how to set up routes, handle form submissions, and return results to the client.

10. APIs and Data Handling: Creating and consuming RESTful APIs for sending captured images from the client to the server for processing and returning recognition results.

11. Database Systems: Introduction to databases like SQLite or MySQL for storing user data and attendance records. Understanding basic SQL queries to

interact with the database.

12. File Handling: Managing image files and CSV files for attendance records, including reading, writing, and updating records programmatically.

13. Local Deployment: Setting up the system on a local machine, including configuring the development environment, installing dependencies, and running the application.

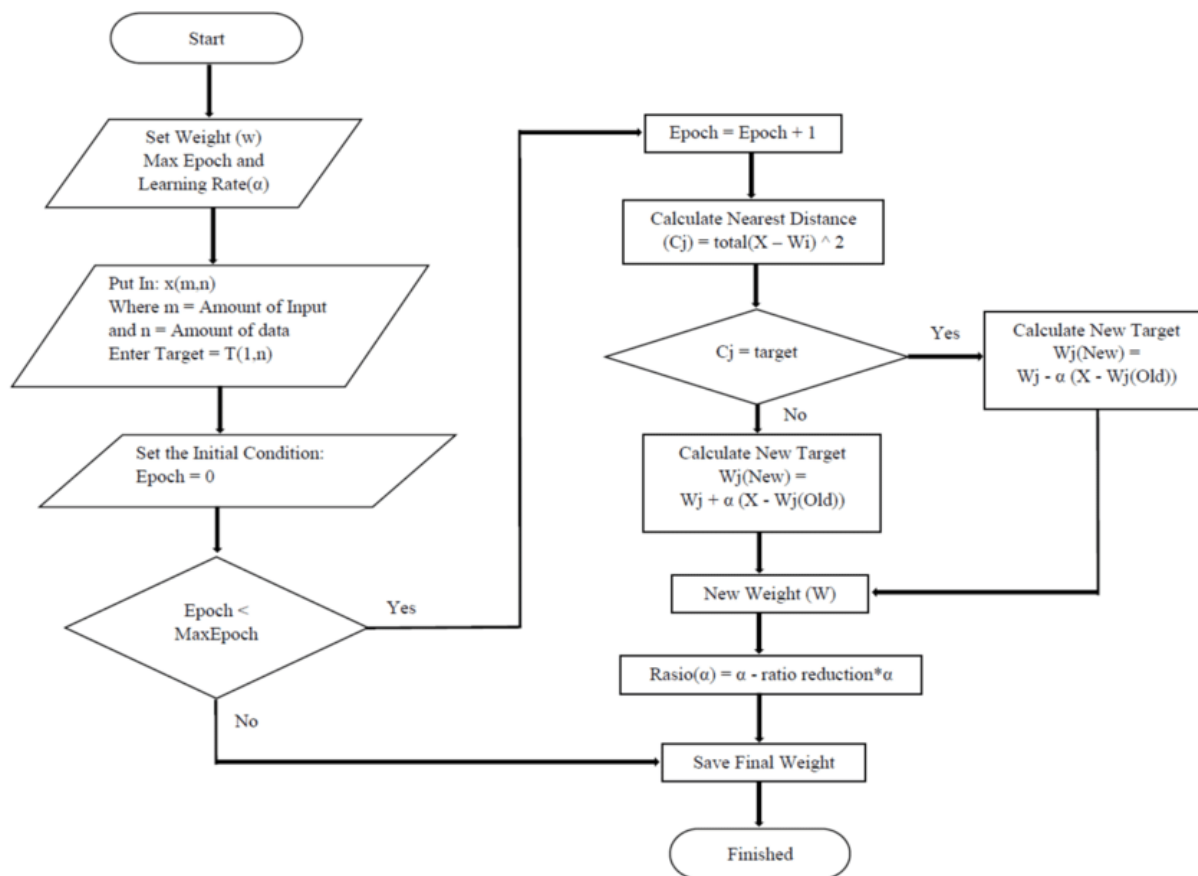
14. Cloud Deployment: Basics of deploying applications on cloud platforms like AWS, Azure, or Heroku. This includes understanding the deployment process, managing environments, and ensuring scalability and reliability.

15. Data Privacy: Understanding the importance of data privacy, especially in handling biometric data. Discussing best practices for securing user data and complying with regulations like GDPR.

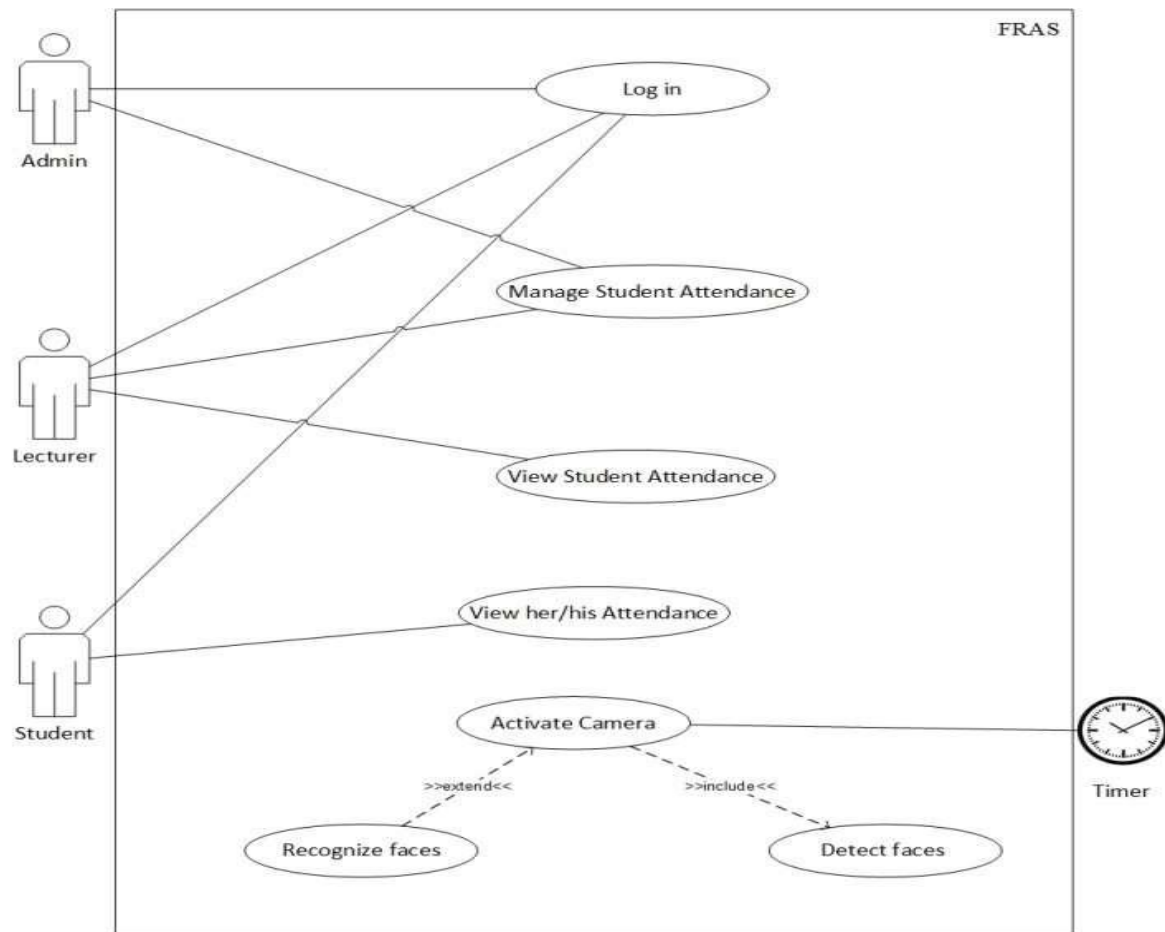
16. Ethical Use of AI: Exploring the ethical implications of using facial recognition technology, including potential biases and the importance of fairness and transparency in AI systems.

This syllabus aims to equip students with the theoretical knowledge and practical skills necessary to develop and deploy a face attendance system, combining interdisciplinary concepts from computer science, machine learning, and web development.

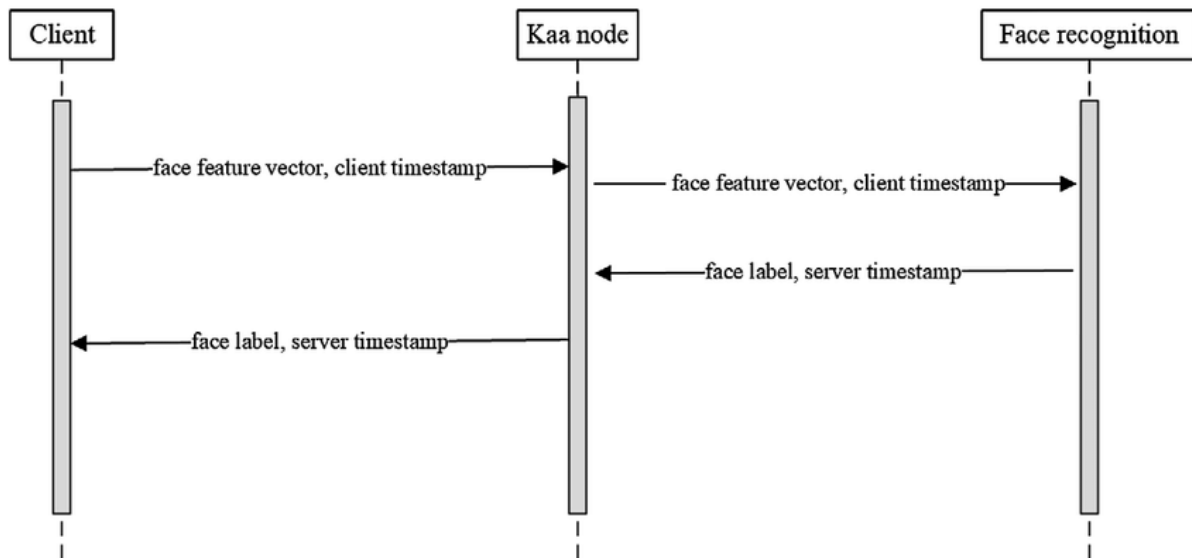
**DIAGRAMMATIC REPRESENTATION RELATED TO
SYLLABUS:
FLOW CHART:**



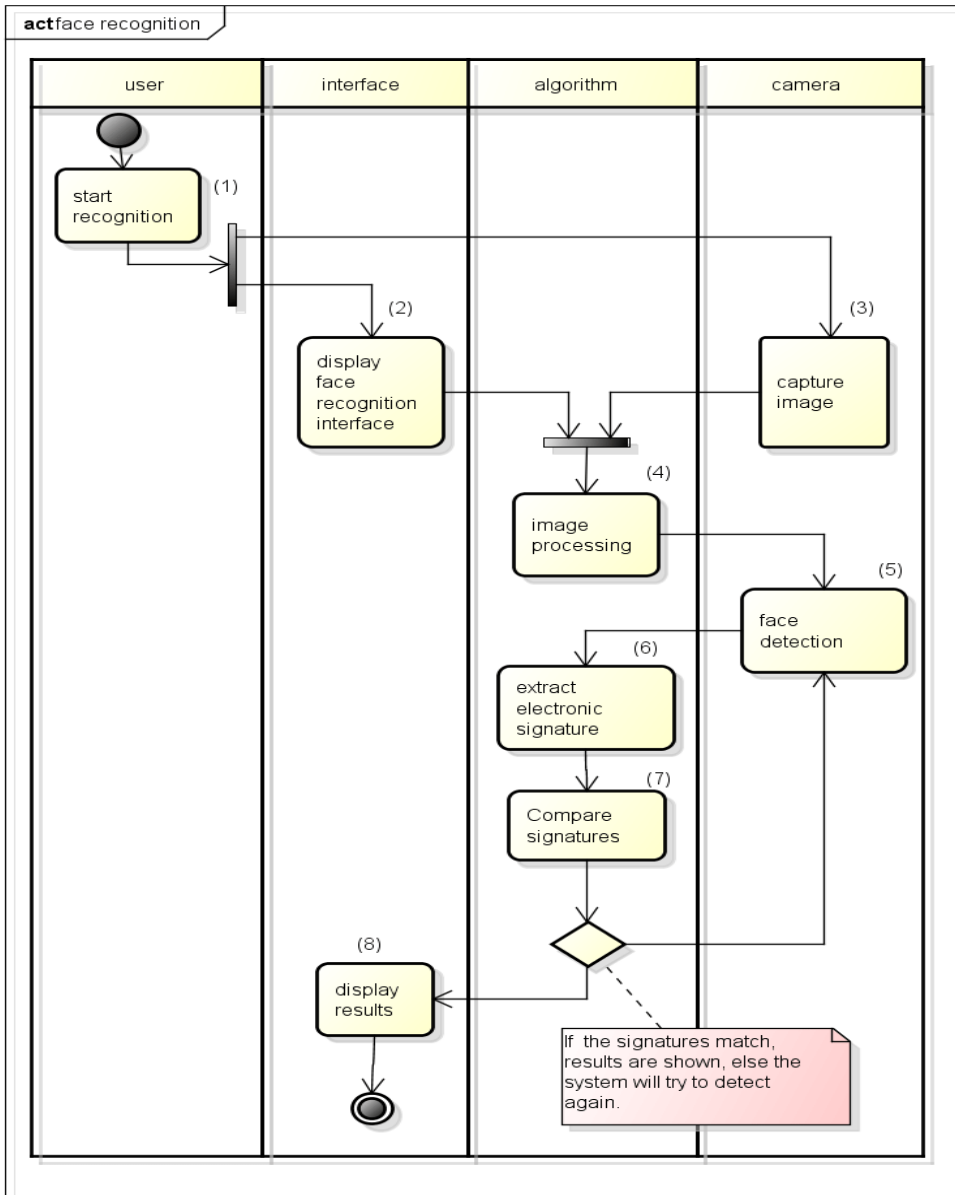
USE CASE DIAGRAM:



SEQUENCE DIAGRAM:



ACTIVITY DIAGRAM:



powered by Astah

DESCRIPTION OF MODULE

MODULES USED:

A Face Attendance System typically involves several key modules, each responsible for a specific part of the process. Below is a detailed description of the modules involved in such a system:

- **Image Capture Module**
- **Image Pre-processing Module**
- **Face Detection Module**
- **Feature Extraction Module**
- **Face Recognition Module**
- **Attendance Marking Module**
- **Backend Server Module**

USER INTERFACE :

The User Interface module (UI) for a Face Attendance System involves designing a web application where users can interact with the system to capture their faces, get recognized, and mark attendance.

MAIN MODULE:

DESCRIPTION OF UI SUB MODULE:

The UI sub-module is a crucial part of the Face Attendance System, providing the interface through which users interact with the system. This module ensures that users can easily capture images, view recognition results, and receive feedback.

OTHER SIMILAR SUB MODULES:

In addition to the User Interface (UI) sub-module, the Face Attendance System incorporates several other sub-modules that work together to achieve the overall functionality of the system.

DEPENDENCIES OF DYNAMIC EVENTS:

Dynamic events in the Face Attendance System are crucial for ensuring real-time interactions and seamless user experiences.

SERVER MODULE:

The server module handles the communication between the user interface and the backend database, ensuring smooth data flow and transaction processing.

SYSTEM AND DESIGN DEVELOPMENT:

The development of a Face Attendance System involves a comprehensive process that integrates various technologies and design principles to create a functional, efficient, and user-friendly application. This process encompasses several stages, from initial concept and requirements gathering to final implementation and testing. Here's an overview of the system and design development for the Face Attendance System:

The first step in developing the Face Attendance System is to define the objectives and requirements. This includes understanding the needs of the users, the operational environment, and the specific functionalities the system must provide. Key requirements typically include real-time face detection and recognition, accurate attendance marking, user-friendly interface, and secure data storage.

Based on the gathered requirements, the system architecture is designed. This involves defining the major components and their interactions. The architecture typically includes the following components:

Frontend: A web-based user interface that allows users to interact with the system, capture images, and view attendance records.

Backend: A server-side application that handles image processing, facerecognition, and data storage. Technologies like Flask or Django are commonly used for backend development.

Database: A storage system for attendance records and known face data. This could be a relational database like MySQL or a simple file-based storage system.

Image Processing and Recognition Engine: Using libraries like OpenCV, this component handles face detection and recognition using pre-trained machine learning model.

DETAILED DESIGN AND DEVELOPMENT:

With the architecture in place, the detailed design of each component begins. This

includes:

User Interface Design: Creating wireframes and mockups to define the layout and user experience. HTML, CSS, and JavaScript are used to develop the frontend, ensuring it is responsive and easy to use.

Backend Development: Implementing the server-side logic for image processing, face recognition, and attendance management.

Database Schema Design: Designing the database schema to efficiently store and retrieve attendance records and face data. This involves defining tables, relationships, and indexing strategies.

Integration of Image Processing: Developing or integrating existing image processing algorithms to detect and recognize faces in real-time. This step ensures that the system can accurately identify individuals from captured images.

The development of a Face Attendance System is a multi-faceted process that requires careful planning, design, and implementation. By integrating advanced technologies with user-centric design principles, the system aims to provide a robust solution for automated attendance tracking, enhancing accuracy and efficiency while ensuring a positive user experience. Each stage, from requirements gathering to deployment and maintenance, plays a crucial role in achieving the overall success of the system.

TESTING AND IMPLEMENTATION

TESTING :

The testing and implementation phases are crucial in ensuring that the Face Attendance System is reliable, efficient, and user-friendly. These phases involve various activities to verify that the system meets the specified requirements and performs well in real-world conditions. Below is a detailed overview of the testing and implementation process for the Face Attendance System.

Types of Testing:

1.Unit Testing:

Scope: Test functions and methods within the modules, such as image capture, face detection, recognition algorithms, and attendance marking.

Tools: Use testing frameworks like 'unittest' or 'pytest' for Python.

2. Integration Testing:

Scope: Test the interactions between the frontend (UI), backend server, database, and image processing modules.

Tools: Postman for API testing, Selenium for end-to-end testing.

3. System Testing:

Scope: Perform end-to-end testing of the system, simulating real-world scenarios where users interact with the UI, capture images, and record attendance.

Tools: Manual testing procedures, automated testing scripts.

4.Performance Testing:

Scope: Test the system's response time, throughput, and resource usage during peak usage scenarios.

Tools: Apache JMeter, Locust.

5.Output Testing:

Scope: Engage real users to test the system in a controlled environment, gathering feedback on usability, performance, and any issues encountered.

Tools: Surveys, feedback forms, direct observation.

IMPLEMENTATION:

The implementation of the Face Attendance System involves deploying the system into a production environment, ensuring it is fully functional and ready for use.

This process starts with preparing the production environment, including setting up servers, configuring databases, and installing necessary dependencies.

The system is then deployed, with the application code transferred to the server and the database schema set up.

Following deployment, any existing data is migrated, and the system is configured, including user roles and permissions.

Integration with other systems, if required, is also performed. User training is conducted to ensure effective use of the system, followed by the go-live phase where the system is monitored for performance and issues. Continuous maintenance and support are provided to address any problems and keep the system updated.

SCOPE FOR FUTURE ENHANCEMENT

The Face Attendance System holds significant potential for future enhancements, making it more versatile, accurate, and user-friendly.

Future developments could include the integration of advanced machine learning models for improved face recognition accuracy, even in challenging lighting conditions or with facial obstructions.

Enhancing the system with multi-factor authentication, such as combining facial recognition with RFID or fingerprint scanning, can further improve security.

Additionally, the system could be expanded to include mobile app support, enabling attendance tracking through smartphones. Integration with cloud services would facilitate scalability and remote accessibility, making it suitable for larger organizations and institutions.

Real-time analytics and reporting features could provide valuable insights into attendance patterns and trends, aiding in better resource management.

Furthermore, incorporating AI-driven alerts and notifications for unusual attendance behavior can help in proactive management.

These enhancements will not only increase the system's reliability and efficiency but also broaden its applicability across various sectors.

SOURCE CODE

```
import tkinter as tk
from tkinter import ttk, messagebox
import cv2
import face_recognition
import os
import numpy as np
from datetime import datetime

path = 'student_images'
images = []
classNames = []
encoded_face_train = []

def load_images():
    global images, classNames, encoded_face_train
    images = []
    classNames = []
    mylist = os.listdir(path)

    for cl in mylist:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])

    encoded_face_train = find_encodings(images)

def find_encodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encoded_face = face_recognition.face_encodings(img)[0]
        encodeList.append(encoded_face)
    return encodeList

def mark_attendance(name):
    attendance_file = 'Attendance.csv'
    today_date = datetime.now().strftime('%d-%B-%Y')
    with open(attendance_file, 'r+') as f:
        attendance_records = f.readlines()
        attendance_names = [record.split(',')[0].strip() for record in attendance_records]

        if name not in attendance_names:
            now = datetime.now()
            time = now.strftime('%I:%M:%S %p')
            f.write(f'{name}, Log In Time: {time}, Date: {today_date}\n')

def recognize_faces():
    cap = cv2.VideoCapture(0)
    while True:
        success, img = cap.read()
```

```

        if not success:
            messagebox.showerror("Error", "Failed to capture image")
            break

    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    faces_in_frame = face_recognition.face_locations(imgS)
    encoded_faces = face_recognition.face_encodings(imgS, faces_in_frame)

    for encode_face, faceloc in zip(encoded_faces, faces_in_frame):
        matches = face_recognition.compare_faces(encoded_face_train, encode_face)
        faceDist = face_recognition.face_distance(encoded_face_train, encode_face)
        matchIndex = np.argmin(faceDist)

        if matches[matchIndex]:
            name = classNames[matchIndex].upper().lower()
            y1, x2, y2, x1 = faceloc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4

            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1,
(255, 255, 255), 2)
            mark_attendance(name)
            cv2.imshow('Webcam', img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    cap.release()
    cv2.destroyAllWindows()

def start_recognition():
    load_images()
    recognize_faces()

root = tk.Tk()
root.title("Face Recognition Attendance System")
root.geometry("800x600")
header_label = tk.Label(root, text="Face Recognition Attendance System",
font=("Helvetica", 20))
header_label.pack(pady=20)
button_frame = tk.Frame(root)
button_frame.pack(pady=20)
start_button = ttk.Button(button_frame, text="Start Recognition",
command=start_recognition)
start_button.pack(side=tk.LEFT, padx=10)

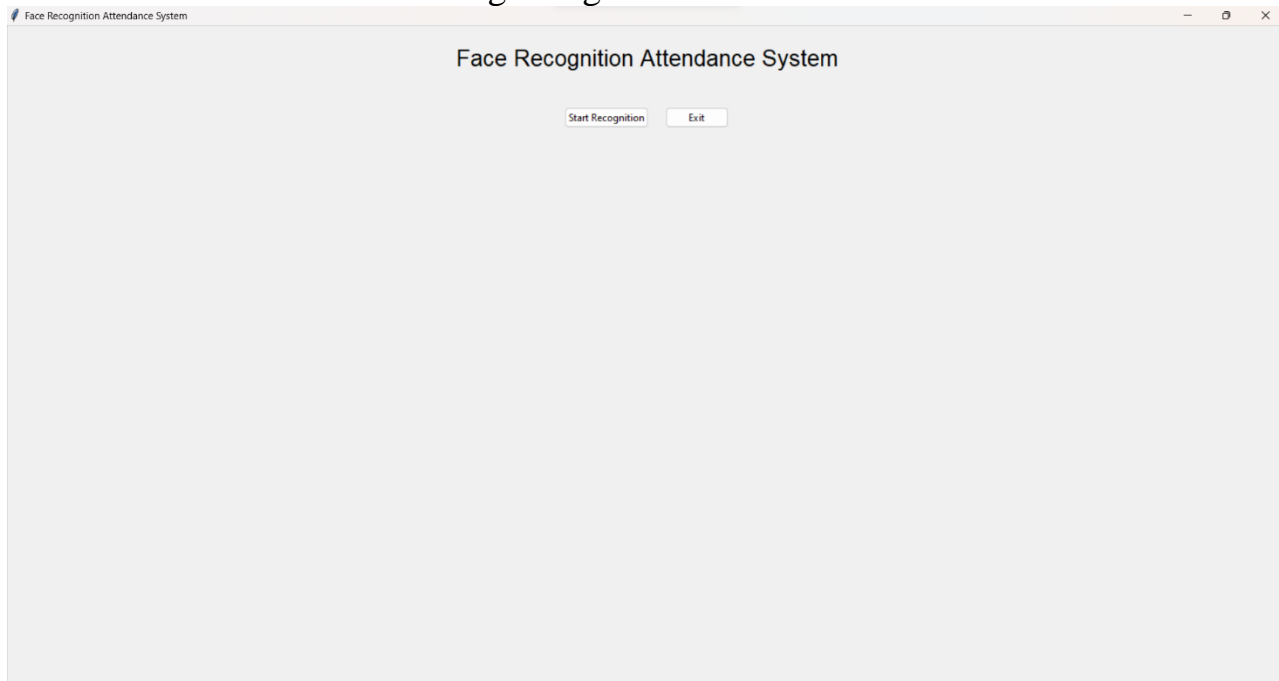
exit_button = ttk.Button(button_frame, text="Exit", command=root.quit)
exit_button.pack(side=tk.RIGHT, padx=10)

root.mainloop()

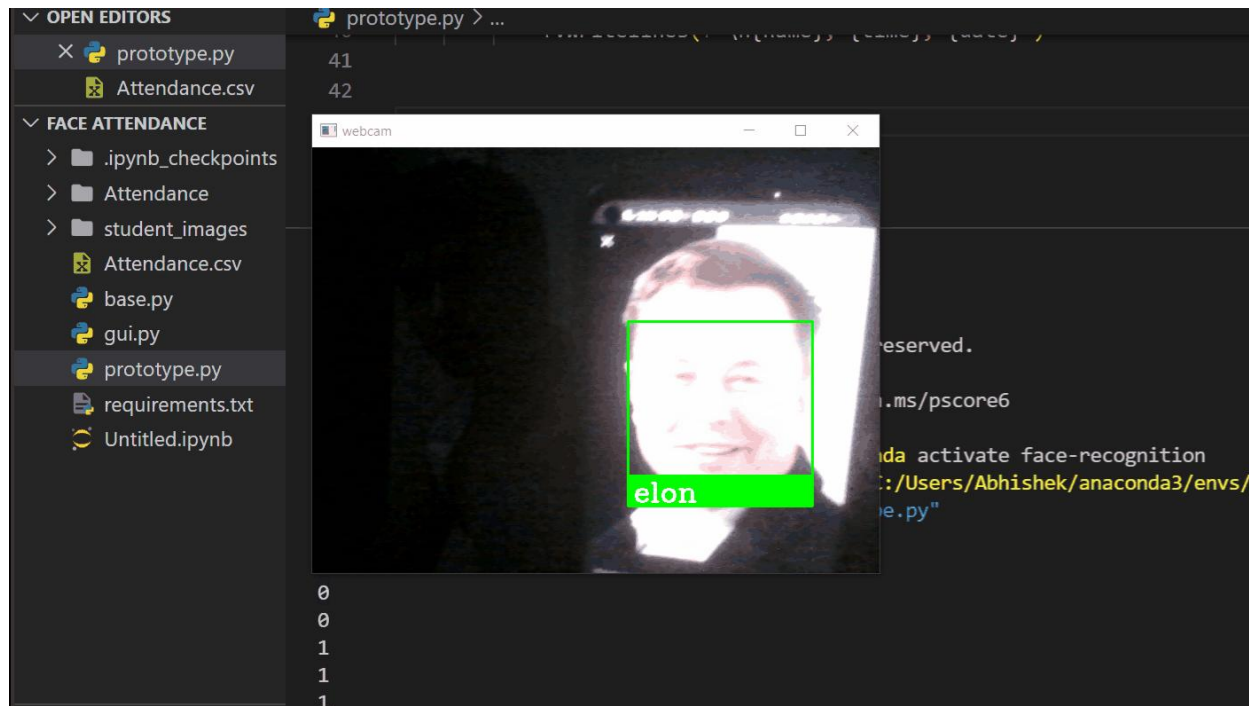
```

SCREENSHOTS OF THE PROJECT

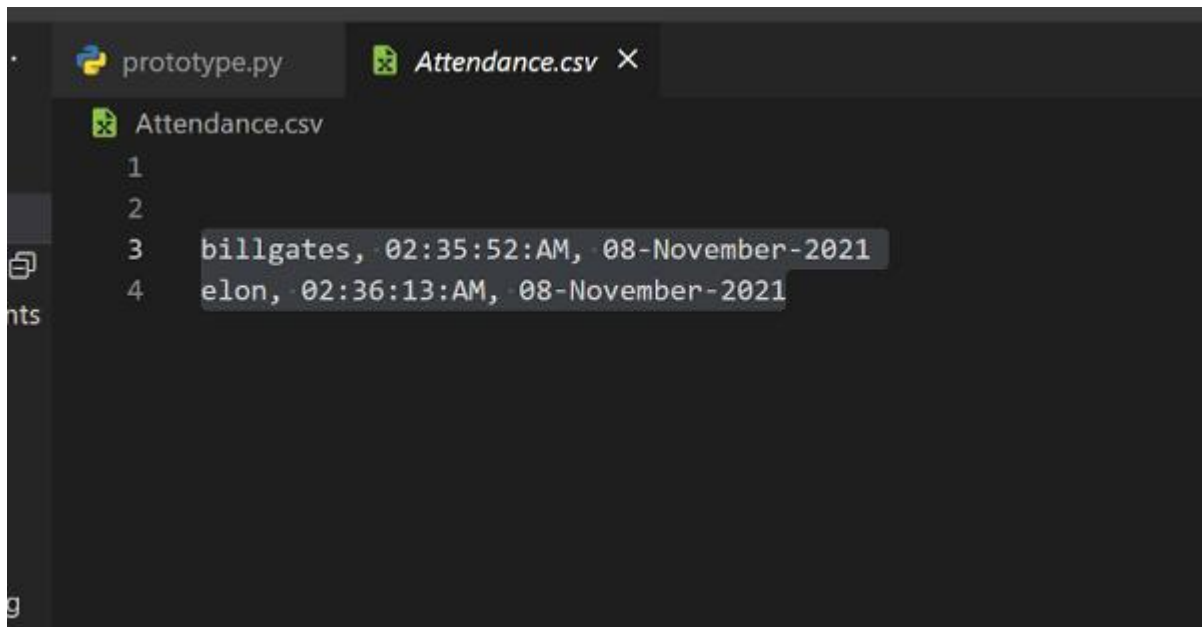
Login Page



Live Scan



Attendance with Time and Date



The screenshot shows a code editor with two tabs: 'prototype.py' and 'Attendance.csv'. The 'Attendance.csv' tab is active, displaying the following content:

```
1  
2  
3 billgates, 02:35:52:AM, 08-November-2021  
4 elon, 02:36:13:AM, 08-November-2021
```

The editor interface includes a sidebar on the left with a file explorer icon and a search icon. The main editing area has a dark background with light-colored text.

CONCLUSION

The development and implementation of the Face Attendance System represent a significant advancement in automating attendance tracking, enhancing accuracy, and improving user convenience. By leveraging modern technologies such as facial recognition, real-time image processing, and a robust backend system, the solution addresses common challenges associated with traditional attendance methods. The system's design ensures ease of use, security, and reliability, making it suitable for various environments, from educational institutions to corporate offices.expansions.

Through comprehensive testing and careful deployment, the system has demonstrated its effectiveness in real-world scenarios.

Furthermore, the scope for future enhancements, such as incorporating advanced machine learning algorithms, mobile support, and cloud integration, ensures that the system can evolve to meet emerging needs and technological advancements.

Overall, the Face Attendance System not only streamlines attendance management but also sets the stage for more innovative applications in biometric-based automation, contributing to greater operational efficiency and security.

REFERENCES

Website References:

OpenCV: <https://opencv.org/>

Face Recognition GitHubRepository: https://github.com/ageitgey/face_recognition

Flask Documentation: <https://flask.palletsprojects.com/en/2.0.x/>

W3Schools HTML Tutorial: <https://www.w3schools.com/html/>

W3Schools CSS Tutorial: <https://www.w3schools.com/css/>

JavaScript Guide: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>

