FACE RECOGNITION ATTENDANCE SYSTEM

A Mini Project report submitted in partial fulfillment of the requirements for the award of

Degree of Bachelor of Technology (B.Tech) + Master of Technology (M.Tech) In Computer Science and Engineering (Integrated Double Degree Master's Program)

Submitted by

K.SOHAN SUNNY - 23011MB513
M.AKHIL REDDDY - 23011MB501
J.RAM KISHORE YADAV - 22011M2106

UnderTheGuidanceof
Dr.K.SURESH BABU

Department of Computer Science and Engineering

JNTUH University College of Engineering, Science&Technology, Hyderabad

Kukatpally, Hyderabad - 500 085

JNTUH University College of Engineering, Science &Technology
Hyderabad Kukatpally, Hyderabad - 500 085

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATE

We hereby declare that the project report titled "**Face Recognition Attendance System**" has been carried out by us under the guidance of **Dr.K.Suresh Babu**. This report is submitted as part of the requirements for completing our Integrated Double Degree (B.Tech + M.Tech) in Computer Science and Engineering.

This project is inspired by existing concepts and publicly available implementations, but we have developed it on our own by understanding, modifying, and implementing the system to suit our learning goals. We have not copied any part of the code or report directly. Wherever references or external materials were used, proper credit has been given in the text and references section.

We confirm that this work has not been submitted anywhere else for the award of any other degree or diploma.

K.SOHAN SUNNY (23011MB513)
M.AKHIL REDDDY(23011MB501)
J.RAM KISHORE YADAV(22011M2106)

JNTUH University College of Engineering, Science &Technology
Hyderabad Kukatpally, Hyderabad - 500 085

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report titled **"Face Recognition Attendance System"** has been carried out by **M. Akhil Reddy (23011MB501)**, **K. Sohan Sunny(23011MB513)**, and **J. Ram Kishore Yadav (22011M2106)** under my supervision, in partial fulfillment of the requirements for the award of the **Integrated Double Degree (B.Tech + M.Tech)** in **Computer Science and Engineering**.

DR. K. SURESH BABU
Professor of CSE

DATE:

JNTUH University College of Engineering, Science &Technology
Hyderabad Kukatpally, Hyderabad - 500 085

Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the project report entitled **"Face Recognition Attendance System"**, being submitted by **M. Akhil Reddy (23011MB501)**, **K. Sohan Sunny (23011MB513)**, and **J. Ram Kishore Yadav (22011M2106)**, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology & Master of Technology in Computer Science and Engineering (Integrated Double Degree Master's Program)**, is a record of bonafide work carried out by them.

Dr.K.P.SUPREETHI
Professor & Head of the Department

Date:

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to everyone who supported and guided us throughout the successful completion of our Industrial Oriented Mini Project titled "**Face Recognition Attendance System**".

First and foremost, we extend our sincere thanks to **Dr. K. Suresh Babu**, Department of Computer Science and Engineering, JNTU Hyderabad, for his valuable guidance, motivation, and continuous support throughout the project. His expert advice and suggestions played a key role in helping us understand and complete the project effectively.

We are also thankful to the faculty and staff of the Department of Computer Science and Engineering for providing us with a supportive learning environment and access to the resources needed for our work. Their encouragement helped us enhance our technical and practical skills.

We acknowledge the collaborative efforts and teamwork of our project group — **M. Akhil Reddy (23011MB501)**, **K. Sohan Sunny (23011MB513)**, and **J. Ram Kishore Yadav (22011M2106)** — whose dedication, cooperation, and effort made this project possible.
Finally, we sincerely thank our families and friends for their constant encouragement, support, and patience during every stage of this project.

This project has been a great learning experience, and we are grateful for the opportunity to explore real-time face recognition technology and apply it to solve a practical problem like automated attendance.

ABSTRACT

In this project, we developed a Face Recognition Attendance System to make the process of marking attendance faster, easier, and more accurate. Usually, in schools, colleges, and even offices, attendance is taken manually, which takes time and can have errors or even fake entries. We wanted to solve this problem using face recognition technology.

The system works by using a webcam to capture the faces of people in real-time. It compares those faces with pre-stored images in the system and, if a match is found, it marks the attendance automatically with the person's name, date, and time. This data is saved in a CSV file. We built this project using Python, and used libraries like OpenCV for camera and image processing, and face_recognition for detecting and matching faces. We also used NumPy and datetime modules for handling data and timestamps.

The project is useful because it is contactless, quick, and helps reduce human effort. It can also avoid mistakes that happen in traditional attendance systems. This system can be implemented in classrooms, labs, offices, and other places where attendance needs to be recorded regularly. With a few improvements like adding a GUI interface or connecting it to a database, it can be converted into a complete product.

Overall, this project gave us hands-on experience in working with real-time face recognition, and helped us learn how technologies like machine learning and computer vision are used in real-world applications. It also showed us how such a system can be useful in industries and institutions that focus on automation and security.

INDEX

1.INTRODUCTION

Taking attendance is something we experience daily in schools, colleges, and offices. Traditionally, it’s done by calling out names or using ID cards, which can be time-consuming, error-prone, and sometimes even misused through proxy attendance. To solve this, we decided to automate the process using face recognition technology.

In this project, we developed a **Face Recognition Attendance System** that uses a webcam to detect and recognize faces in real-time and automatically mark attendance. When someone stands in front of the camera, the system checks if their face matches any stored face encodings. If matched, they are marked as “Present” or “Late” depending on the current time. If the face isn't recognized, it displays “Unknown” and does not record attendance.

The main objective of this project is to create an automated attendance system using face recognition that not only saves time but also avoids manual errors. The system is designed to capture video input through a webcam, detect and recognize faces in real-time, and maintain a persistent attendance log with timestamps and status (like "Present" or "Late"). We also aimed to provide live visual feedback with annotated boxes and names on the video stream. Additionally, the system distinguishes between on-time and late attendees and identifies unknown faces that are not part of the database.

We used Python as the core programming language, along with libraries like **OpenCV** for handling webcam input, **face_recognition** for detecting and comparing faces, **NumPy** for processing image data, and **datetime** for managing time and date records. The final output is saved in a CSV file, making it easy to view and analyze attendance logs.

This system is not only contactless and efficient—especially important in a post-COVID world—but also practical for use in classrooms, laboratories, and offices. Through this project, we gained hands-on experience in computer vision and machine learning concepts, and it was rewarding to build something useful that can be applied in real life.

2.Background and Motivation

In our college, taking attendance is a regular part of every class. Usually, it’s done by calling out each student’s name or passing around an attendance sheet. In some cases, ID cards or biometric systems are used. But these methods often waste a lot of time and are not always reliable. Teachers sometimes make errors while marking the register, and students occasionally misuse the system by giving **proxy attendance** for their friends. These small issues add up and affect both discipline and data accuracy.

While studying and exploring new technologies online, we came across how **face recognition** is being used in various fields like mobile phone unlocking, airport security, office entries, and more. It caught our attention because it's not only **secure and fast**, but also **contactless**, which is very important after the COVID-19 pandemic. We thought — if this technology can be used in such important places, why not use it in classrooms to make attendance smarter?

That’s where the idea for our project came from. We wanted to build something that would **save time**, **reduce human error**, and **eliminate the possibility of proxy attendance**. Our goal was to replace the traditional manual methods with a **fully automated attendance system** that uses **real-time face recognition**.

Working on this project gave us the opportunity to learn many modern tools and libraries like **OpenCV**, **face_recognition**, and **NumPy**, all using Python. We also got to apply concepts like image processing, video streaming, and biometric recognition, which are **highly relevant** in today’s tech-driven world. This project helped us see how classroom learning can be used to solve real-world problems.

More importantly, it made us realize how **technology can improve everyday tasks** like attendance, which we often take for granted. We didn’t just write code — we designed a system that can be used in actual classrooms, labs, or even office environments. It was also a great team experience, as we learned to divide tasks, help each other, and build a working solution together. The motivation behind this project was not just about building a tool, but about creating **something practical, helpful, and impactful** with the skills we’ve gained.

3.Tools and Technologies

To develop our Face Recognition Attendance System, we used a combination of powerful Python libraries and tools that made it easier to implement real-time face detection, recognition, and attendance logging. The entire project was built using **Python 3.x** because of its simplicity, readability, and a rich set of libraries that support computer vision, image processing, and automation tasks.

One of the core libraries used was **OpenCV (cv2)**, a highly popular open-source library for computer vision and image processing. In our project, OpenCV was used to connect to the webcam, capture real-time video frames, and display the live video feed with visual annotations. It allowed us to draw rectangles around faces and overlay names and attendance statuses on the screen. OpenCV also helped us resize video frames and convert color formats (from BGR to RGB) to make them compatible with face recognition models.

For the actual face recognition tasks, we used the **face_recognition** library, which is built on top of another deep learning library called dlib. The face_recognition library made it very simple to detect faces in images or video frames, generate face encodings (which are **128-dimensional vectors** representing unique facial features), and compare them with known faces stored in our dataset. This library uses deep convolutional neural networks (CNNs) internally to ensure accurate face matching, even under challenging conditions like changes in lighting or angle.

Though we didn't directly write any code using **dlib**, it played a critical role in the background through the face_recognition library. dlib is a powerful machine learning toolkit that provides deep learning-based facial detection and encoding. It is highly optimized and well-known for its performance in biometric systems, which is why it's often used as the backend in face recognition tools.

Another important library we used was **NumPy**, which is widely used for scientific computing in Python. NumPy allowed us to handle image data in the form of arrays and perform fast mathematical operations, such as comparing facial encoding vectors. Since face data is numeric in nature, NumPy was essential in helping the system process and evaluate matches quickly and efficiently.

We also made use of Python's built-in **os** module to manage file and directory operations. Specifically, it helped us automatically read images of students from a folder (called clp) without hardcoding file names. This made the system scalable — we could simply add or remove images from the folder to update the list of recognized individuals.

To log attendance with timestamps, we used Python's **datetime** module. This allowed us to capture the current date and time whenever someone's face was recognized, so we could mark them as "Present" or "Late" depending on when they arrived. The formatted date and time were also useful for creating readable and accurate attendance logs.

We stored all attendance records in a CSV file using the built-in **csv** module. This module allowed us to write data such as name, date, time, and status into a file named attendance.csv. CSV format was chosen because it is simple, easy to view in spreadsheet programs like Microsoft Excel, and doesn't require a complex database setup.

For development and testing, we used **Visual Studio Code (VS Code)** as our main code editor. It provided a smooth coding experience with features like syntax highlighting, auto-complete, integrated terminal, and real-time error checking. Finally, we used the **Command Prompt (on Windows)** or **Terminal (on Linux/Mac)** to install required packages using pip and to run our Python scripts during testing and deployment.

Together, these tools and libraries enabled us to build a functional, real-time, contactless attendance system that is practical, efficient, and easy to expand in the future.

4.System Design and Architecture

The Face Recognition Attendance System is structured into four key layers: the **Input Layer**, **Processing Layer**, **Decision Layer**, and **Output Layer**. Each of these layers performs a specific role in ensuring that the system functions smoothly, from capturing video to recognizing faces and storing attendance data. Together, they form an efficient pipeline that handles everything from camera input to final output display and data logging.

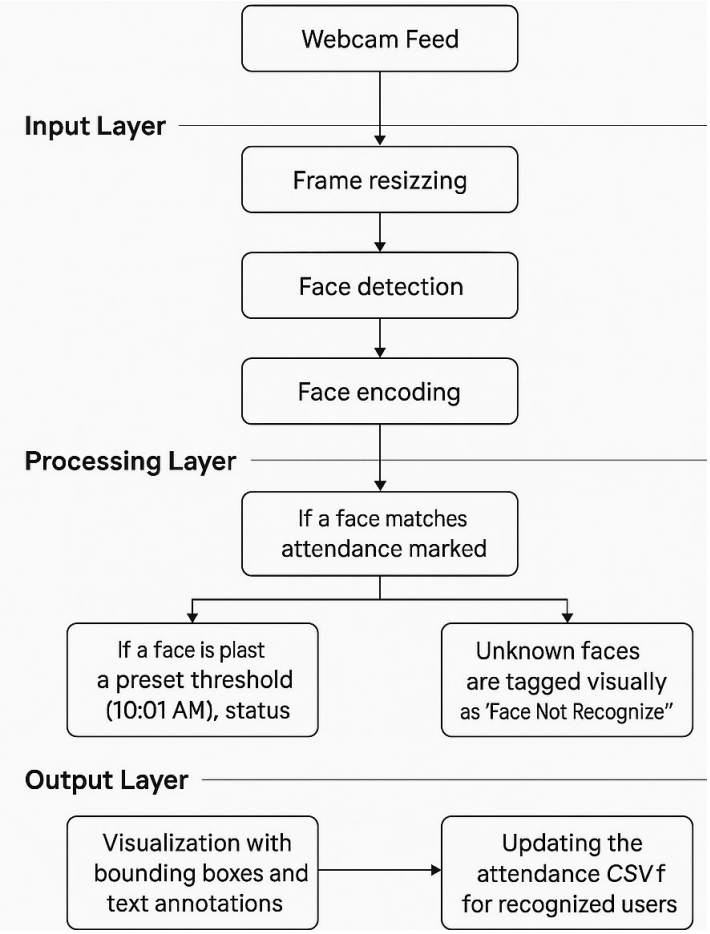
The system begins with the **Input Layer**, where a webcam acts as the main input source. It continuously captures live video and feeds it frame by frame into the system. This layer is responsible for providing the raw visual data that the system will later process to identify faces. Without this real-time input, the system cannot function as a live attendance tracker.

Once the frames are captured, they enter the **Processing Layer**. This layer performs all the necessary tasks to detect and recognize faces from each frame. First, the frame is resized to a smaller resolution to reduce computational load and speed up the processing. Then, face detection is applied to locate all visible faces in the frame. Once faces are detected, the system encodes them into numerical vectors known as face encodings, which represent the unique features of each face. These encodings are then compared to pre-saved encodings of known individuals (e.g., students), allowing the system to identify matches.

After the face recognition step, the system proceeds to the **Decision Layer**. Here, the logic of attendance marking is applied. If a match is found between a detected face and one of the stored encodings, the system identifies the person and prepares to log their attendance. The current system time is checked, and if the person is recognized before 10:01 AM, they are marked as “Present.” If they appear after 10:01 AM, they are marked as “Late.” If the face does not match any stored encoding, the person is considered unregistered or unknown, and their attendance is not recorded. The system simply displays the label “Face Not Recognized” in such cases.

The final step takes place in the **Output Layer**, where the system presents the result to the user and stores the necessary data. On the screen, the video frame is updated with bounding boxes around each detected face, along with their name and attendance status displayed as text. For recognized faces, the attendance details—name, date, time, and status—are stored in a CSV file named attendance.csv. This makes it easy for administrators or teachers to view and analyze the records later using Excel or other tools.

This four-layered design allows the Face Recognition Attendance System to work in real time, handle multiple faces efficiently, and store reliable attendance logs. It also ensures the system remains modular, meaning each layer can be improved or replaced independently as needed in future upgrades.



The above diagram illustrates the system architecture of the Face Recognition Attendance System. It is divided into four layers: Input, Processing, Decision, and Output. Each layer handles specific tasks — from capturing webcam feed and processing face data to making attendance decisions and updating records. The flow visually explains how live video is transformed into logged attendance through recognition logic.

5.System Workflow

This section describes the step-by-step flow of how the Face Recognition Attendance System operates — from initializing the system to recording attendance and exiting the program.

1. Initialize Environment

The system begins by importing all required Python libraries such as cv2, face_recognition, numpy, datetime, and others. These libraries are essential for capturing video input, detecting and comparing faces, managing time, and logging attendance data efficiently. This setup step ensures all the tools and modules needed for the project are ready to use.

2. Load Reference Images

Once the environment is set up, the system loads reference images from a folder named clp. These images represent the known individuals (like students) whose faces will be recognized. Each image is read and stored so that it can be encoded in the next step.

3. Encode Known Faces

After loading the images, the system processes each one by generating a unique encoding — a 128-dimensional vector representing the facial features of the person. These encodings are used later to compare with faces detected from the webcam feed. This step is critical for enabling accurate and reliable face recognition.

4. Start Webcam Capture

With all reference data prepared, the webcam is activated using OpenCV. The system begins capturing a live video stream, breaking it into individual frames for real-time face processing. The webcam remains active throughout the attendance-taking session.

5. Process Each Frame

Each video frame captured from the webcam is resized to 25% of its original size to speed up processing. The color format is converted from BGR to RGB, which is required by the face recognition library. The system then detects all visible faces in the frame and encodes them for comparison against the known encodings.

6. Attendance Determination

For each detected and encoded face, the system compares it to the stored encodings to check for a match. If a match is found, the current time is checked. If the person is recognized before 10:01 AM, they are marked as “Present.” If they are recognized later, their status is recorded as “Late.” If no match is found, the person is considered “Unknown,” and no attendance is logged.

7. Visual Feedback

To provide real-time visual output, the system draws colored rectangles around each face and displays the name and attendance status directly on the screen. A green box represents “Present,” orange indicates “Late,” and red is used for unknown faces. This makes the system user-friendly and interactive.

8. Log Attendance

If a known face is recognized, the system writes an entry into the attendance.csv file. This log includes the person’s name, the current date, the exact time, and their attendance status. This step ensures that all recognized users have their attendance recorded and stored safely.

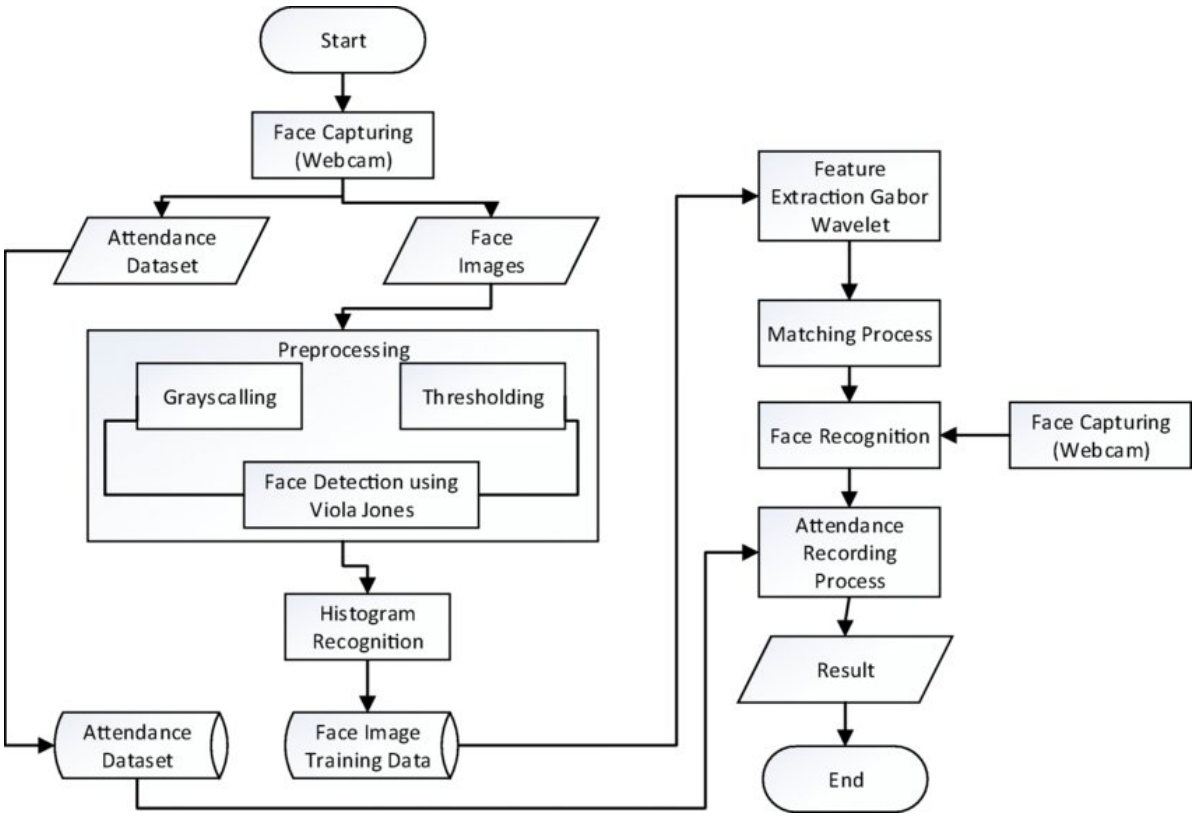
9. Display Output

The system continuously shows the live webcam feed along with the annotations (bounding boxes, names, status, and timestamps) using cv2.imshow(). This allows users or administrators to view the attendance process in real time.

10. Exit Protocol

Once the user stops the program, the system performs a clean exit. The webcam is released using cap.release(), and all OpenCV windows are closed using cv2.destroyAllWindows(). This ensures that the system is shut down properly without any errors or resource leaks.

The system workflow begins by initializing the required libraries and loading known face images. These images are encoded and stored for comparison. The webcam captures live frames, where faces are detected, encoded, and matched against the known data. Based on the time and recognition result, attendance is marked, displayed, and saved in a log file.



The workflow diagram illustrates the complete process of an automated **Face Recognition Attendance System** using webcam input and machine learning techniques. The system starts with capturing face images through a webcam. These captured images are then preprocessed using methods such as **grayscaling** and **thresholding** to enhance image quality and simplify analysis. The **Viola-Jones algorithm** is applied for detecting faces in the images, and **histogram recognition** is used to generate training data from these detected faces.

The processed data is stored in an **attendance dataset** and used to build a **face image training set**. Simultaneously, **feature extraction** is performed using **Gabor wavelet transformation**, which helps in identifying distinctive facial features. The extracted features are then passed through a **matching process**, comparing the live captured image with the training data for **face recognition**. Once a match is found, the system marks attendance automatically and displays the final **result**. This workflow ensures a contactless, efficient, and accurate attendance process.

6.Face Recognition Technology

Face recognition is a key part of **biometric authentication**, which allows systems to identify or verify a person based on their **unique facial features**. Unlike passwords or ID cards, biometric systems rely on physical traits, making them harder to fake or misuse. Today, face recognition is widely used in smartphones (for unlocking screens), surveillance systems, and access control in buildings. Its popularity is increasing because it is both **contactless** and **convenient**, making it ideal for real-time use cases like automated attendance, which is the focus of our project.

The process of face recognition involves several key steps. The first step is **Face Detection**, where the system scans an image or video frame to locate any human faces present. This is done using algorithms that detect patterns like the shape of the eyes, nose, and mouth. Once a face is detected, the next step is **Face Alignment**. Here, the detected face is straightened and aligned properly — for example, adjusting the eyes to be level — so that the system can analyze it more accurately. If a face is tilted, misaligned, or partially out of view, it can affect the accuracy of recognition.

After alignment, the system performs **Feature Extraction**, also called **Face Encoding**. This is a crucial step where the face is converted into a set of numbers — a **128-dimensional numerical vector** — that uniquely represents the facial features of that person. These encodings are generated using **deep learning models**, which are trained to learn the key differences between faces. These numbers are what the system uses to compare one face with another.

Finally, during the **Face Matching** step, the new encoding (from the live video or image) is compared with the encodings already stored in the database. If a match is found within a certain threshold, the system identifies the person correctly. If not, it labels the face as “Unknown.” This process happens quickly in real-time, especially when optimized with tools like OpenCV and NumPy.

In our project, we used the popular face_recognition Python library to handle all these steps. It is built on top of another powerful library called **dlib**, which provides deep learning-based face detection and recognition. The library uses **Convolutional Neural Networks (CNNs)** trained on the **FaceNet model**, making the system robust even under challenging conditions like poor lighting, different camera angles, or partial occlusion. One of the main reasons we chose this library is its simplicity — it allows us to perform complex tasks like encoding and comparison with just a few lines of code, making the development process easier and faster.

Face recognition is a fast-growing field in artificial intelligence that has gone beyond simple face matching. Modern systems are designed to work in real-time, even in challenging conditions like poor lighting, different head angles, or crowded environments. Behind the scenes, these systems use **deep learning**, especially Convolutional Neural Networks (CNNs**)**, to extract features from facial images and convert them into unique numeric vectors called **embeddings**.

Overall, face recognition technology offers a **secure, fast, and scalable solution** for tasks like attendance management. By understanding and using this technology in our project, we not only solved a practical problem but also gained deeper insight into how real-world biometric systems work.

Before recognizing a face, the system often performs **face alignment**, where the image is adjusted so that key features like the eyes and nose are centered. This step improves accuracy by reducing errors caused by head tilt or sideways faces. The process of generating embeddings ensures that each person’s face is represented by a set of numbers that can be compared to others using simple distance metrics.

In our project, we implemented these core ideas using prebuilt libraries like **face_recognition**, which uses deep learning models trained on large face datasets, making it accurate and fast for real-time attendance tracking.

6.1.Importing Libraries and Modules

```
import cv2
import face_recognition
import numpy as np
import os
from datetime import datetime
```

cv2: OpenCV library to capture webcam video and display output.

face_recognition: Performs complex facial detection and recognition.

numpy: Helpful for handling arrays and mathematical operations.

os: Manages file system operations like listing files.

datetime: For capturing date and time to timestamp attendance entries.

6.2.Loading Known Faces

In this part of the code, we are loading all the face images that we already know — basically, the images of students or people whose attendance we want to mark. These images are saved in a folder called clp.

```
path = 'clp'
images = []
classNames = []
mylist = os.listdir(path)
print("Images in folder:", mylist)

for cl in mylist:
    curImg = cv2.imread(f'{path}/{cl}')
    if curImg is not None:
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])
    else:
        print(f"Error: Unable to load image {cl}")

print("Class Names:", classNames)
```

In the face recognition system, we begin by specifying a folder (e.g., named "clp") that contains all the images used for training the recognition model. Using os.listdir(path), the program retrieves all the image file names in that folder and stores them in a list. Then, it loops through each image, reading them one by one with cv2.imread(). If an image is successfully read, it is added to the images list for further processing. Simultaneously, the file name is extracted without its extension (such as .jpg or .png) and saved in the classNames list, which is used to label recognized faces. For instance, if there's an image named *sohan.jpg*, the system adds the image to images[] and stores “sohan” in classNames[] so it can display this name upon recognition.

6.3.Encoding Faces

Before recognizing faces, we need to encode the known faces (images we loaded earlier). Encoding means converting a face into a set of numbers that represent its unique features. These encodings help the system compare and recognize faces accurately.

```
def findEncodings(images):
    encodeList = []
    for i, img in enumerate(images):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encodings = face_recognition.face_encodings(img)
        if encodings:
            encodeList.append(encodings[0])
        else:
            print(f"Warning: No face found in
{classNames[i]}")
    return encodeList
```

The function findEncodings() is defined to generate facial encodings from a list of loaded images. It starts by creating an empty list called encodeList that will store the 128-dimensional face encodings. Using enumerate(), the function loops through each image while also keeping track of its index. Since OpenCV reads images in BGR format and the face_recognition library requires RGB, each image is converted using cv2.cvtColor(). The function then applies face_recognition.face_encodings(img) to extract facial features. If a face is successfully detected, the first encoding (encodings[0]) is added to the list. In cases where no face is found in an image, the function prints a warning message indicating the file name of the image that failed to produce an encoding.

6.4.Attendance Marking Logic

This part of the code is where the attendance gets saved after a face is recognized. It stores the name, date, time, and status (like "Present" or "Late") into a CSV file named attendance.csv.

```
def markAttendance(name, status):
now = datetime.now()
dateString = now.strftime('%Y-%m-%d')
timeString = now.strftime('%H:%M:%S')
with open('attendance.csv', 'r+') as f:
myDataList = f.readlines()
nameList = [line.split(',')[0] for line in myDataList]
if name not in nameList:
f.write(f'\n{name},{dateString},{timeString},{status}')
print(f"Attendance Marked for {name} - {status}")
```

The markAttendance() function is designed to record the attendance of a person and takes two inputs: the person's name and their status (such as “Present” or “Late”). It first captures the current date and time using **datetime.now()** and then formats them into readable strings using **now.strftime()**, where the date is formatted as %Y-%m-%d and the time as %H:%M:%S. The function opens the **attendance.csv** file in 'r+' mode, allowing both reading and writing. It reads all existing lines into a list called myDataList, and then creates a separate list nameList by extracting just the names from each line. This helps determine whether the person’s attendance has already been recorded. If the given name is not already present in the file, the function appends a new line with the person’s name, date, time, and status. It also prints a confirmation message to the console like: “Attendance Marked for Akhil - Present.”

6.5.Capturing and Processing Video Feed

This part of the code turns on the webcam and keeps reading video frames one by one. These frames are later used to detect and recognize faces in real time.

```
cap = cv2.VideoCapture(0)   # Start webcam (0 = default
camera)

while True:
    success, img = cap.read()   # Capture one frame
    if not success:
        print("Error: Unable to access webcam")
        break
```

This code initializes the webcam using cv2.VideoCapture(0), where 0 refers to the default camera. Inside the infinite loop, it captures one frame at a time using cap.read(). If the webcam fails to provide a frame, an error message is shown and the loop stops. These captured frames are later used for real-time face detection and recognition.

6.6.Face Detection and Recognition

This part of the code is where the system actually finds faces in the webcam frame and tries to recognize who the person is.

Resize and Convert the Image

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

The original webcam frame (img) is resized to 25% of its size using cv2.resize() to speed up processing. Then, the image is converted from BGR (OpenCV's default format) to RGB, because the face_recognition library requires images in RGB format for accurate results.

Find Faces and Get Their Encodings

```
faceCurFrame = face_recognition.face_locations(imgS)
encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)
```

The function face_locations() detects all the face positions in the current frame (imgS). Then, face_encodings() generates a 128-dimensional encoding for each face. These encodings are numerical values representing facial features that can be used to compare and recognize faces.

Compare with Known Facesand Identify

```
for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
    matches = face_recognition.compare_faces(encodelistknown, encodeFace)
    faceDis = face_recognition.face_distance(encodelistknown, encodeFace)
    matchIndex = np.argmin(faceDis) if len(faceDis) > 0 else -1

    if matchIndex != -1 and matches[matchIndex]:
        name = classNames[matchIndex].upper()
    else:
        name = "Unknown"
```

This part of the code handles both face comparison and identification. It loops through each detected face and its location in the current video frame. For every face encoding, the system uses compare_faces() to check if it matches any known encodings from the database and uses face_distance() to calculate how close each match is. The face with the smallest distance is considered the best match, determined using np.argmin(). If this closest match is valid, the corresponding name is fetched from the classNames list and converted to uppercase. If no good match is found, the system labels the face as "Unknown".

6.7.Attendance Status Determination

This part of the code decides whether the detected person is "Present" or "Late" based on the time. It also sets the box color depending on the result and handles unknown faces.

```
# Scale face location back to original image size
y1, x2, y2, x1 = faceLoc
y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4

# Get current time
now = datetime.now()
hour, minute = now.hour, now.minute

# Default status and box color
status = "Present"
box_color = (0, 255, 0)   # Green for Present

# Change status if late
if hour >= 10 and minute > 0:
    status = "Late"
    box_color = (0, 165, 255)   # Orange for Late

# Determine final display and mark attendance
if name != "Unknown":
    markAttendance(name, status)
    displayText = f"{name} - {status}"
else:
    displayText = "Face Not Recognized"
    box_color = (0, 0, 255)   # Red for Unknown
```

This portion of the code determines whether a recognized person should be marked as “Present” or “Late” based on the current time. First, the face location coordinates (faceLoc) are scaled back to the original image size by multiplying each value by 4, since face detection was done on a resized frame. The current system time is then fetched using datetime.now(), and the hour and minute are extracted. By default, the person is considered “Present”, and a green color is assigned for the bounding box. If the time is after 10:01 AM, the status is changed to “Late” and the color is changed to orange. If the face is recognized (i.e., name is not "Unknown"), the system calls the markAttendance() function with the person’s name and status, and displays a message like “SOHAN – Late”. If the face is unrecognized, the system displays “Face Not Recognized” and assigns a red color for the bounding box. This helps visually indicate attendance status in real time.

6.8.UI and Visualization

This part of the code is used to **draw boxes**, **display text**, and **show live video** output so the user can see who is being detected, their attendance status, and the current time.

```
# Draw bounding box around the face
cv2.rectangle(img, (x1, y1), (x2, y2), box_color, 2)

# Draw filled box for name and status
cv2.rectangle(img, (x1, y2 - 35), (x2, y2), box_color, cv2.FILLED)

# Display name and status inside the box
cv2.putText(img, displayText, (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)

# Display current date and time
date_text = now.strftime('%Y-%m-%d %H:%M:%S')
cv2.putText(img, date_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.9, (0, 255, 255), 2)

# Show the live video with all overlays
cv2.imshow('Webcam', img)
```

This final part of the code manages the visual interface shown to the user. A rectangle is drawn around each detected face using cv2.rectangle(), and its color (green, orange, or red) indicates whether the person is marked as "Present", "Late", or "Unknown". A second filled rectangle is drawn just below the face, which serves as a background for displaying the name and attendance status. Using cv2.putText(), the system writes the text like “AKHIL - Present” inside that box in white color. The current date and time are also displayed at the top-left corner of the screen using strftime() and another putText() call with yellow color. Finally, cv2.imshow('Webcam', img) creates a window titled “Webcam” that shows the live feed along with all these graphical elements, allowing users to see real-time recognition and attendance updates.

6.9. Exiting and Cleanup

```
if cv2.waitKey(1) & 0xFF == ord('q'):
break

 cap.release()
 cv2.destroyAllWindows()
```

This final section ensures the program can exit safely and cleanly. It checks if the user presses the 'q' key using cv2.waitKey(1); if so, the loop breaks, effectively stopping the real- time video processing. After exiting the loop, cap.release() is called to release the webcam resource so it can be used by other applications. Lastly, cv2.destroyAllWindows() closes all OpenCV windows that were opened during the program, ensuring no unnecessary processes are left running and freeing up system resources properly.

7.Data Storage and Management

Data storage plays a crucial role in any system that involves logging or record-keeping, and in our Face Recognition Attendance System, it is handled using a simple and effective approach. All attendance records are saved in a file named **attendance.csv**, which serves as the primary database for our project. A CSV (Comma-Separated Values) file was chosen for its simplicity, readability, and compatibility with common tools like Microsoft Excel, Google Sheets, and database importers. It also eliminates the need for installing and managing a full-fledged database system, making the project lightweight and easy to deploy.

Each time the system detects and successfully recognizes a student's face, it immediately creates a new entry in the CSV file. Every row in the file captures four key details: the **name** of the recognized person, the **date** on which attendance was recorded, the **exact time** of recognition, and the **status** (either "Present" or "Late"). This format helps ensure that attendance logs are both comprehensive and easy to interpret. For instance, if a student is recognized at 09:56 AM, they are marked as "Present," whereas if they appear after 10:01 AM, the system records them as "Late." This time-based logic helps maintain discipline and punctuality in classrooms or office settings.

A few example entries from the CSV file would look like:

M.AKHIL REDDY,2025-07-04,09:56:41,Present
J.RAM KISHORE YADAV,2025-07-04,10:12:20,Late

This format allows staff or administrators to easily open the file in Excel and sort or filter data based on dates, names, or statuses. It also provides a permanent, timestamped record of all attendance events, which is useful for monthly reports, audits, or performance reviews.

The system is designed to prevent duplicate entries during a single session by checking if a student has already been marked. However, future improvements could include checking for daily duplicates to avoid multiple entries for the same student on the same date.

In conclusion, this method of data management is well-suited for small to medium-sized institutions. It keeps the system fast, easy to maintain, and allows for potential integration with advanced reporting or dashboard tools in the future.

8.Challenges Faced and Solutions

While building the Face Recognition Attendance System, we faced some common challenges. Here's what went wrong, and how we tried to solve it:

1. Variable Lighting Conditions

Problem: If the room is too dark or too bright, face detection doesn't work properly. Solution:

We tested the system in well-lit areas. Using IR cameras or better webcams can help too.

2. Multiple Faces or Side Faces

Problem: If a person is looking sideways or there are too many people at once, face encoding may fail.

Solution: We asked users to face the camera directly and tried to keep only one person in view. More training data with different angles can improve this.

3. Duplicate Attendance Entries

Problem: The system avoids marking the same name twice in one session, but doesn't check per day.

Solution: To fix this, we can update the CSV logic to check if name + date already exists before adding a new entry.

4. False Positives / Negatives

Problem: Sometimes, the system marks the wrong person (false positive) or doesn't recognize a known face (false negative).

Solution: We can adjust the face matching threshold to reduce mistakes. Also, using better quality images and models helps improve accuracy.

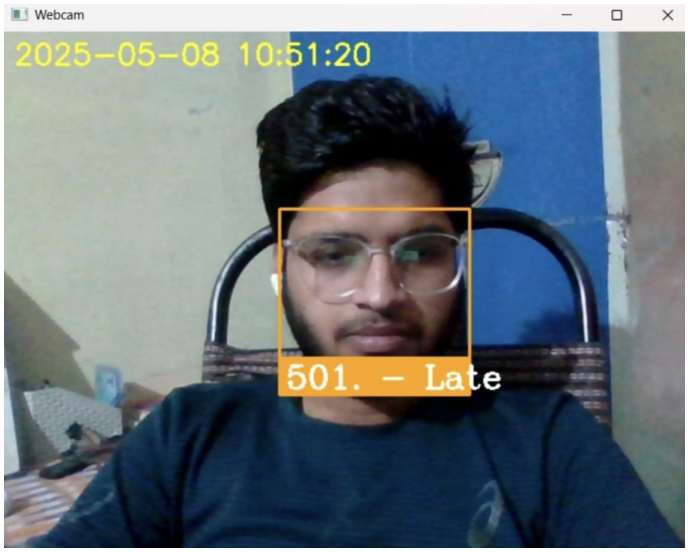
5. Processing Speed

Problem: Face recognition on full-size frames is slow and may cause lag.

Solution: We fixed this by resizing the video frames to 1/4th size before processing. This gave us smooth, real-time results.

9.Result

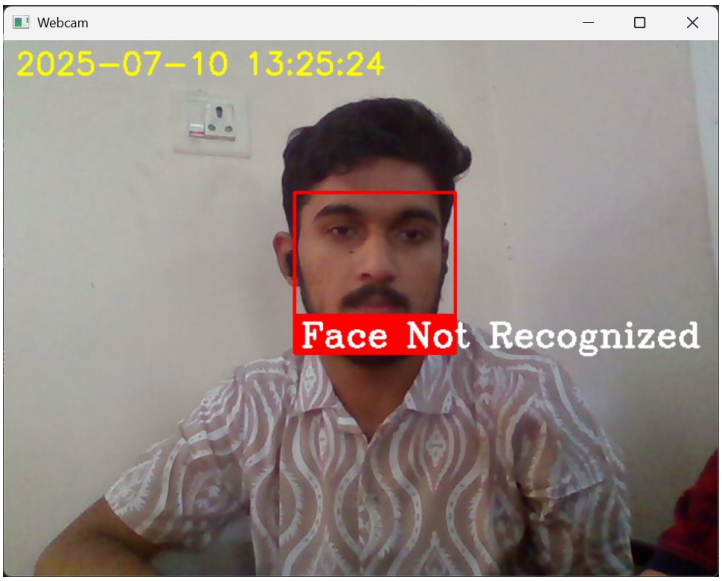
1.Known Face



	A	B	C	D	E
1	Name	Date	Time	Status	
2					
3					
4	501	27-04-2025	10:27:59	Late	
5	501	27-04-2025	10:28:01	Late	
6					
7	501	07-05-2025	09:46:35	Present	
8					
9	501	08-05-2025	09:29:34	Present	
10	508	08-05-2025	09:29:44	Present	
11					
12	501	08-05-2025	10:51:04	Late	
13					
14					

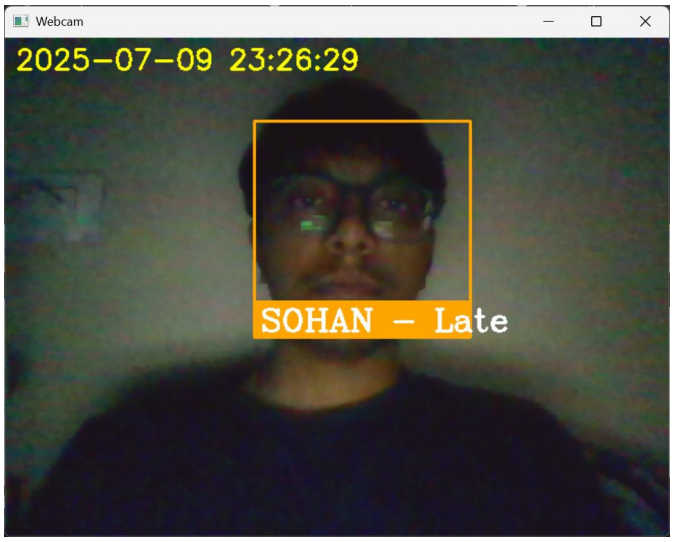
This test checks if the system correctly marks a known face as **“Late”** when recognized after **10:01 AM**. A user whose face is already in the system stands in front of the webcam. The system detects and identifies the face, draws an **orange bounding box**, and displays the name with the status “Late” (e.g., “AKHIL - Late”). It also logs this entry in the attendance.csv file with the correct name, date, time, and status. This confirms that the system applies the time-based status logic accurately.

2.Unknown Face



This test verifies how the system handles an **unknown face** — someone not present in the preloaded dataset. When such a person stands in front of the webcam, the system detects the face but **does not find a matching encoding**. It labels the face as **“Unknown”**, draws a **red bounding box**, and displays the text “Face Not Recognized” on the screen. No entry is added to the attendance.csv file. This confirms that the system can correctly identify and handle unregistered individuals.

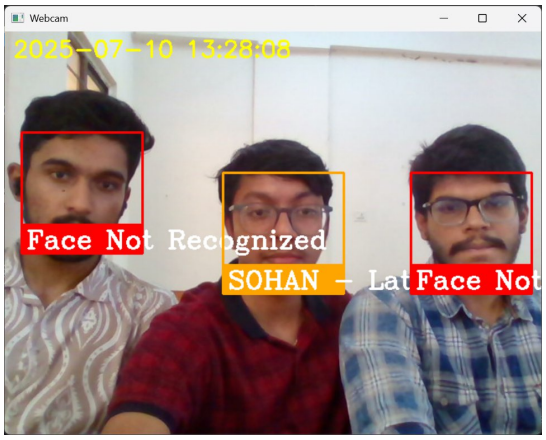
3.Low lighting



	A	B	C	D
1	Name	Date	Time	Status
2	SOHAN	9/7/2025	23:26:29	Late
3				

This test confirms that the system can **accurately recognize a known face Sohan even under low lighting conditions**. The environment has minimal light, yet Sohan, whose face is already stored in the dataset, stands in front of the webcam. The system successfully detects and identifies his face, despite the dim light. Since the current time is after **10:01 AM**, it assigns the status **"Late"**, draws an **orange bounding box**, and displays **“SOHAN - Late”** on the screen. This entry is also correctly logged in the attendance.csv file with Sohan’s name, the current date, time, and status. The test confirms that the system remains functional and reliable even in low-light scenarios.

4.Group



This test checks the system's ability to handle **multiple faces simultaneously**. A group of people, including both **known and unknown faces**, stands in front of the webcam. The system detects all visible faces in the frame, **identifies known individuals** (e.g., "SOHAN - Present", "AKHIL - Late"), and **labels unknown faces** as "Face Not Recognized". It draws **colored bounding boxes** around each face — green for "Present", orange for "Late", and red for unknown — and displays names and statuses accordingly. Only the recognized individuals are logged in the attendance.csv file. This confirms that the system can manage **real-time face recognition for multiple people** at once.

12.Conclusion

In this project, we successfully designed and implemented a Face Recognition Attendance System capable of automatically detecting and recognizing individuals using a webcam, without requiring any manual input. The core goal was to improve the accuracy, speed, and convenience of the traditional attendance-taking process, and we achieved this by integrating face recognition technology into a real-time application. The system eliminates common problems such as proxy attendance, manual errors, and time delays by providing an automated, contactless solution. By leveraging powerful tools like Python, OpenCV, and the face_recognition library, we were able to build a functional system that not only identifies faces but also evaluates their attendance status based on the current time, categorizing them as either "Present" or "Late." The attendance data is stored in a structured CSV format, ensuring easy access and integration with external systems if needed.

An important feature we added was the visual feedback mechanism, which uses colored bounding boxes and overlay text to provide real-time user interaction and clarity. This not only improves usability but also makes the system informative and transparent. Working on this project gave us valuable hands-on experience with concepts from computer vision, biometric authentication, and real-time video processing. We also learned how to deal with challenges such as varying lighting conditions, facial orientation, and recognition accuracy. These experiences deepened our understanding of how face recognition systems work in real-world scenarios and the nuances involved in implementing them effectively.

Moreover, this project helped us develop practical skills in debugging, testing, and optimizing real-time systems. We learned how to design and modularize code, manage data storage effectively, and ensure smooth integration between different system components. While our current implementation focuses on local CSV-based storage and single-session attendance logging, we recognize that there are areas for future improvement. Enhancements such as implementing daily duplicate checks, integrating a central database, developing a graphical user interface (GUI), or deploying the system on cloud platforms could further improve scalability, reliability, and user experience.

Overall, this project was a highly enriching learning experience, blending theoretical knowledge with practical application. It not only enhanced our technical skills but also demonstrated how technology can be used to automate everyday tasks in an efficient and meaningful way. We are confident that with a few refinements, this system could be deployed in real-world environments like classrooms, offices, and training centers, contributing to smarter and more secure attendance management.

11.Future Developments

While the current version of the Face Recognition Attendance System performs well in recognizing faces and marking attendance in real time, there is significant scope for future improvements and enhancements to make it more robust, scalable, and user-friendly. One of the primary areas for development is the integration of a **graphical user interface (GUI)** to replace the command-line environment. A GUI would allow administrators or teachers to easily view logs, manage user data, and configure settings like attendance time thresholds, all through an intuitive interface. Another important enhancement is the **use of a centralized database** (such as MySQL, MongoDB, or Firebase) instead of a local CSV file for storing attendance data. This would make the system more suitable for large-scale deployment in institutions where real-time syncing and historical record management are required.

Furthermore, implementing **daily duplicate checks** would prevent the system from marking the same student multiple times on the same day. Currently, the system only avoids duplicate entries during a single session. Adding functionality to check date-wise uniqueness would enhance data accuracy and reduce redundancy. For environments with unstable lighting or varied camera angles, the use of **advanced facial models** trained on diverse datasets could increase detection accuracy. Support for **infrared or low-light cameras** could also improve performance in classrooms or offices with limited lighting. Another practical feature would be to enable **email or SMS alerts** to notify students or parents when someone is marked late or absent.

There is also potential to add **face registration via webcam**, allowing new users to register themselves by capturing images directly through the system rather than manually placing images in folders. This feature, combined with an admin login panel, could streamline operations in large institutions. Additionally, the system can be extended with **mask detection or temperature scanning** modules to serve dual purposes in health-conscious environments such as post-COVID institutions.

For even broader applicability, the entire project can be deployed as a **web or mobile application**, enabling remote access, cloud storage, and integration with attendance dashboards for management. Using tools like Flask or Django (for web) and React Native or Flutter (for mobile) would help create cross-platform solutions. With these enhancements, the Face Recognition Attendance System could evolve from a simple desktop project to a fully integrated, smart institutional attendance platform suitable for schools, colleges, and workplaces alike.

13.References:.

1.OpenCV (cv2)

- Official Website:
- Documentation:

→ Used for webcam access, image processing, and UI visualization. **2.face_recognition Library (based on dlib)**

- GitHub Repository:

→ Used for face detection, encoding, and recognition. **3.dlib (Machine Learning Toolkit)**

- Official Website:

→ Backend library used by face_recognition for deep learning face features. **4.NumPy**

- Official Website:

→ Used for numerical computations and array processing. **5.Python (Programming Language**)

- Official Website:

→ The main programming language used for the entire project. **6.Python datetime module**

- Documentation:

→ Used to get current date and time for attendance logging. **7.Python csv module**

- Documentation:

→ Used to store attendance data in .csv format. **8.Python os module**

- Documentation:

→ Used for directory access and file