



**University of Duhok**

**College of Engineering**

**Electrical and Computer Department**

## **Face Recognition Based Attendance System**

Mini Project || 2<sup>nd</sup> Stage || 2<sup>nd</sup> Semester

2022-2023

**Group's Names:**

Dara Mustafa Saleem

Mohamad Sadiq Tahir

Salahadin Mushir Ahmad

Ahmad Ameer Mousa

**Supervisor:**

Dr. Mohammed subhi hadi

## **Abstract**

Face detection attendance systems have revolutionized the way we track and manage attendance in various settings. These systems utilize advanced facial recognition technology to accurately identify individuals and record their attendance without the need for manual processes.

In this project, we designed a face detection attendance system to streamline the attendance tracking process. Our system consisted of a camera-based setup capable of capturing facial images and a robust facial recognition algorithm for accurate identification.

The primary objective of our project was to create a convenient and efficient solution for attendance management that eliminates the need for traditional methods such as manual sign-in sheets or swipe cards. By harnessing the power of face detection technology, our system allows users to simply stand in front of the camera for attendance recording.

The outcome of our project is a user-friendly face detection attendance system that simplifies the attendance tracking process. Our system demonstrates the effectiveness and reliability of face detection technology in improving overall attendance management, offering a seamless and contactless experience for both administrators and individuals being recorded.

## **Acknowledgement**

We would like to extend our heartfelt appreciation to all those who have contributed to the completion and success of our face detection attendance system project. We would like to express our deepest gratitude to our project supervisor, Dr. Mohamad, for his invaluable guidance, unwavering support, and extensive knowledge throughout the entire development journey.

We would also like to thank our team members for their dedication, hard work, and collaborative efforts in bringing this project to fruition. Each member's unique skills and contributions have played a significant role in the system's development and overall success.

Additionally, we would like to acknowledge the authors and researchers whose works and studies have provided us with valuable insights and information. Their contributions have been instrumental in shaping the foundation of our face detection attendance system.

Lastly, we would like to thank our Department for providing us with the necessary resources and environment to carry out this project effectively.

# Contents

<b>CHAPTER 1: INTRODUCTION.....</b>	<b>6</b>
<b>CHAPTER 2: DESIGN .....</b>	<b>8</b>
• 2.1 Hardware Requirements:.....	8
• 2.2 Machine Learning Model: .....	11
• 2.3 Data Storage: .....	11
• 2.4 User Interface: .....	12
• 2.5 Integration and Deployment:.....	12
<b>CHAPTER 3: CODE OVERVIEW AND ANALYSIS.....</b>	<b>13</b>
• 3.1 Importing Libraries and Dependencies: .....	13
• 3.2 Flask App Initialization: .....	14
• 3.3 Date Handling: .....	14
• 3.4 Directory and File Handling: .....	14
• 3.5 Helper Functions:.....	15
• 3.6 Flask App Execution:.....	16

**CHAPTER 4: USER INSTRUCTIONS FOR THE FACE RECOGNITION-BASED ATTENDANCE SYSTEM ..... 25**

- 4.1 Program Activation:..... 25
- 4.2 User Interface:..... 26
- 4.3 Add User: ..... 27
- 4.4 Take Attendance: ..... 28

**CHAPTER 5: DISCUSSION ..... 30**

- 5.1 System Performance and Accuracy ..... 30
- 5.2 User Experience and Ease of Use ..... 30
- 5.3 System Limitations ..... 31
- 5.4 Security and Data Integrity ..... 31
- 5.5 Future Enhancements..... 31

**CHAPTER 6 : CONCLUSION..... 33**

- 6.1 Key Findings and Contributions ..... 33
- 6.2 Project Significance and Implications ..... 34
- 6.3 Future Directions ..... 34

**REFERENCES..... 36**

## Chapter 1: Introduction

Attendance management is a crucial aspect of any educational institution, ensuring accurate record-keeping and monitoring of student attendance. Traditional methods such as manual sign-in sheets or swipe cards have their limitations, often prone to errors and time-consuming processes. As technology continues to advance, face detection attendance systems have emerged as an innovative solution to streamline and enhance the attendance tracking process.

A face detection attendance system utilizes sophisticated facial recognition technology to identify individuals and record their attendance without the need for manual intervention. By analyzing unique facial features and patterns, these systems can accurately recognize and match faces against a database, providing a reliable and efficient method for attendance management.

The implementation of a face detection attendance system offers numerous benefits. Firstly, it eliminates the need for physical contact, promoting a hygienic and contactless approach to attendance tracking, particularly in light of current global health concerns. Secondly, it significantly reduces the time and effort required for attendance recording, freeing up valuable resources for educational institutions. Moreover, the system provides real-time data, allowing administrators to access attendance information instantly and generate reports effortlessly.



By adopting a face detection attendance system, educational institutions can enhance overall efficiency, ensure accurate attendance records, and minimize the potential for attendance fraud. Furthermore, students and teachers can benefit from a streamlined process that eliminates the hassle of manual attendance marking, enabling them to focus on more productive activities within the learning environment.

In this project, we aim to design and implement a face detection attendance system specifically tailored for student attendance management. Through the integration of advanced facial recognition algorithms and robust hardware components, our system will offer a user-friendly and reliable solution for accurately recording student attendance.

In the following sections, we will delve into the technical details, methodology, and expected outcomes of our face detection attendance system project, emphasizing the potential advantages it brings to educational institutions and the overall attendance management process.

## Chapter 2: Design

The design of the face detection attendance system involves a combination of hardware components, image processing algorithms, and web application development using Flask. This design section provides an overview of the key components and functionality of the face detection attendance system based on the provided code. Further enhancements and optimizations can be made based on specific requirements and use cases.

Here is a breakdown of the design elements:

### 2.1 Hardware Requirements:

#### 2.1.1 Webcam:

A high-quality webcam is an essential hardware component for the face detection attendance system. The webcam should have a suitable resolution to capture clear and detailed video frames for accurate face detection and recognition.

It is recommended to choose a webcam with good low-light performance, as it ensures reliable operation in various lighting conditions. The webcam should be compatible with the operating system on which the system is deployed (e.g., Windows, macOS, Linux).





### 2.1.2 Computer or Embedded System:

The face detection attendance system can be implemented on a standard computer or an embedded system, depending on the deployment requirements.

For smaller-scale implementations, a regular computer or laptop can provide sufficient processing power to handle the face detection and recognition tasks.

In cases where the system needs to be deployed in multiple locations or integrated with other hardware, an embedded system such as a single-board computer (e.g., Raspberry Pi) may be more suitable. The computer or embedded system should have the necessary specifications to run the required software libraries and frameworks efficiently.

### 2.1.3 Internet Connectivity (Optional):

Depending on the specific use case and requirements, the face detection attendance system may benefit from internet connectivity. Internet connectivity enables real-time data synchronization, remote access to attendance records, and integration with other systems or databases. If internet connectivity is required, ensure that the computer or embedded system has reliable and stable internet access, either through a wired or wireless connection.

### 2.1.4 Power Supply:

Ensure that the computer or embedded system has a stable power supply to ensure uninterrupted operation of the face detection attendance system.

For desktop computers or laptops, connect them to a reliable power source or use an uninterruptible power supply (UPS) to prevent data loss or system shutdowns during power outages.

If deploying the system on an embedded system like a Raspberry Pi, make sure to provide a suitable power supply or utilize battery backup solutions, depending on the deployment location and power availability.

### 2.1.5 Mounting and Positioning:

Consider the mounting and positioning requirements for the webcam to achieve optimal face detection and recognition accuracy.

Position the webcam at an appropriate height and angle to capture faces of individuals accurately.

It is essential to minimize environmental factors that can affect face detection, such as excessive glare, uneven lighting, or obstructions in the camera's field of view.

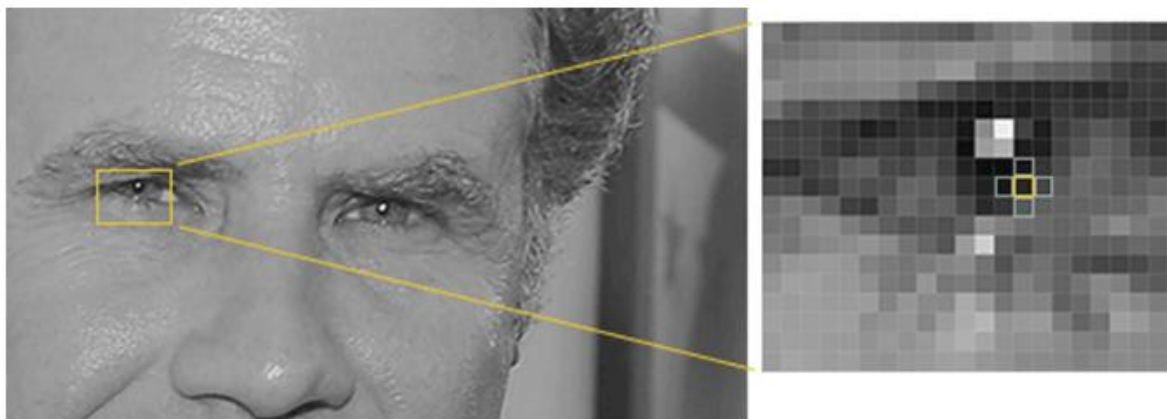
### 2.1.6 Flask Web Application:

The project is built using the Flask framework, a micro web framework in Python. It allows for easy routing, request handling, and rendering of HTML templates.

### 2.1.7 Face Detection and Recognition:

The system uses the Haar cascade classifier provided by OpenCV (`haarcascade_frontalface_default.xml`) for face detection. It analyzes the video feed from the webcam and detects faces in real-time.

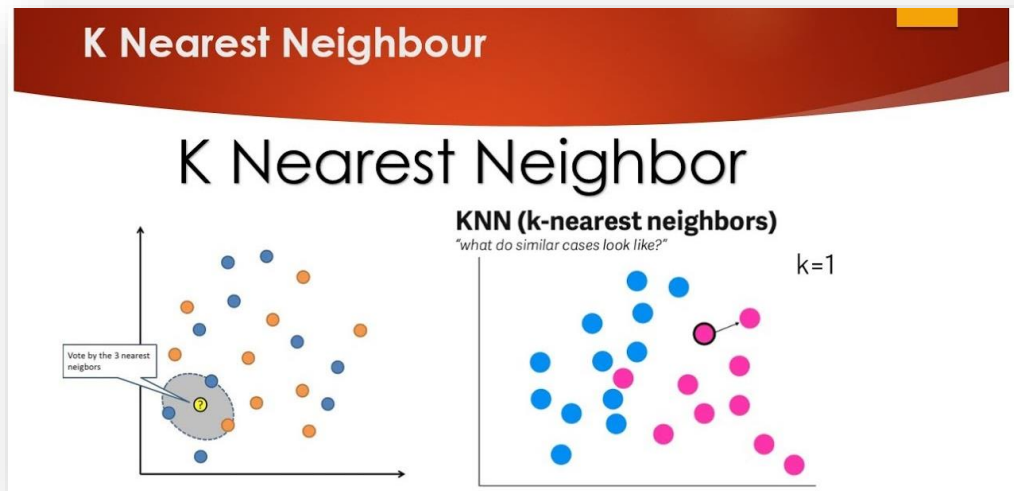
The faces are then extracted from the video frames using the `extract_faces()` function. It converts the captured frame to grayscale, detects faces using the Haar cascade classifier, and returns the coordinates of the detected faces.



## 2.2 Machine Learning Model:

The system employs the K-Nearest Neighbors (KNN) algorithm for face recognition. The `identify_face()` function uses a pre-trained KNN model (`face_recognition_model.pkl`) to predict the identity of the detected face.

The `train_model()` function is responsible for training the KNN model on the available face images in the `static/faces` directory. It reads the images, extracts facial features, and associates them with corresponding labels (user IDs).



## 2.3 Data Storage:

The system stores attendance records in CSV format. The attendance file is created based on the current date using the `datetoday` variable.

The attendance records are stored in the Attendance directory. If the attendance file for the current date does not exist, it is created with the headers: "Name", "Roll", and "Time".

The `add_attendance()` function adds the attendance of a specific user by extracting the username and user ID from the detected face, along with the current time. It appends this information to the attendance file.

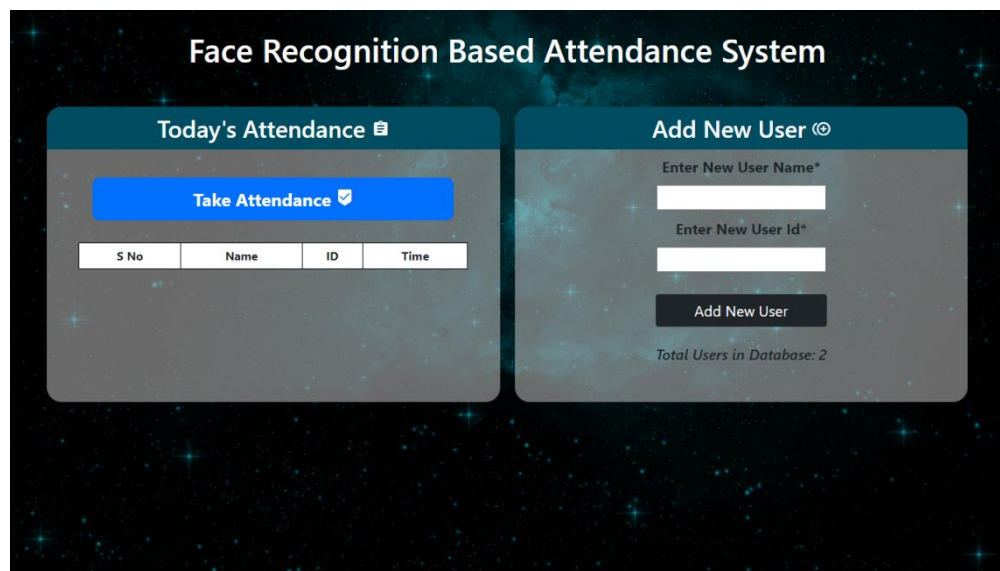
The `extract_attendance()` function retrieves attendance information from the CSV file for display on the web application.

## 2.4 User Interface:

The web application provides a user-friendly interface for interacting with the attendance system. The main page (home.html) displays attendance information, including the names, roll numbers, and times of attendance.

The user interface allows users to start the attendance process by clicking the "Take Attendance" button (start() function). This initiates the webcam feed, detects faces, identifies the individuals, and records their attendance.

The web application also allows users to add new users by providing their name and ID through a form. The add() function captures multiple images of the user's face using the webcam, stores them in the appropriate folder, and trains the face recognition model.



## 2.5 Integration and Deployment:

The Flask web application is run using the `app.run(debug=True)` command, which starts the web server and makes the application accessible through a specified port. The system can be integrated with existing attendance management systems or used as a standalone solution.

Deployment of the system requires configuring the necessary dependencies, such as OpenCV, Flask, and scikit-learn, and ensuring proper file/folder permissions.

## Chapter 3: Code Overview and Analysis

The code for the face recognition-based attendance system is well-structured and organized, with clear separation of concerns. Let's delve deeper into the key sections and functionalities of the code:

### 3.1 Importing Libraries and Dependencies:

The code begins by importing the required libraries and dependencies. These include:

- **cv2 (OpenCV):** Used for image and video processing tasks, including face detection and image resizing.
- **os:** Enables interaction with the operating system, allowing directory and file handling operations.
- **Flask:** Provides the necessary tools for creating a web application and handling HTTP requests.
- **date and datetime:** Used to handle date and time-related operations, such as obtaining the current date and timestamp.
- **numpy (np):** Used for mathematical operations and handling multi-dimensional arrays.
- **sklearn.neighbors.KNeighborsClassifier:** Implements the K-nearest neighbors algorithm for face recognition.
- **pandas (pd):** Provides data structures and data analysis tools for handling CSV files.
- **joblib:** Enables saving and loading machine learning models.

### 3.2 Flask App Initialization:

The Flask application is initialized using the Flask class from the Flask library. An instance of the app is created, representing the web application.

### 3.3 Date Handling:

The code uses the `date.today()` function from the date module to obtain the current date. It then formats the date into two different representations: `datetoday` in the format `"mm_dd_yy"` and `datetoday2` in the format `"dd-Month-YYYY"`. These formatted dates are used for file naming and display purposes.

### 3.4 Directory and File Handling:

To ensure the necessary directories and files are available, the code checks for their existence and creates them if needed. The following directories are checked and created if they don't exist:

- **'Attendance'**: Used for storing attendance records.
- **'static'**: Holds static files, including the face images and the trained face recognition model.
- **'static/faces'**: Contains the face images of registered users.

Additionally, the code checks if the attendance file for the current date exists. If not, a new file is created with the name `f'Attendance/Attendance-{datetoday}.csv'`. This file will store the attendance records for the day, with the headers `"Name"`, `"Roll"`, and `"Time"`.

### 3.5 Helper Functions:

The code defines several helper functions to support different functionalities of the attendance system:

- **totalreg():** This function returns the total number of registered users by counting the number of files in the 'static/faces' directory. It achieves this by listing the files in the directory and counting the number of entries.
- **extract\_faces(img):** Given an image, this function extracts faces using the Haar cascade classifier from OpenCV. It converts the image to grayscale, detects faces using the cascade classifier, and returns the face points (coordinates) as a list. If no faces are detected, an empty list is returned.
- **identify\_face(facearray):** This function performs face recognition by identifying a given face using the trained face recognition model. It takes a face array as input and predicts the corresponding label (identified person) using the loaded model. The predicted label is returned.
- **train\_model():** This function trains the face recognition model using the face images and their corresponding labels available in the 'static/faces' directory. It iterates over the users and their images, reads each image using OpenCV, resizes it to a standard size, and flattens it into a 1D array. The resized face images and corresponding labels are stored in separate arrays. The K-nearest neighbors algorithm (KNeighborsClassifier) is then trained on this data. The trained model is saved using `joblib.dump()` for later use in face identification.

- **extract\_attendance():** This function extracts attendance information from the attendance file for the current date. It reads the CSV file using `pd.read_csv()` and retrieves the names, rolls, and times columns as separate arrays. The length of the DataFrame is also calculated and returned. The attendance information is used for displaying the attendance records on the home page.
- **add\_attendance(name):** This function adds the attendance record of a specific user identified by their name. It splits the name to extract the username and user ID, retrieves the current time using `datetime.now()`, and opens the attendance file for appending. It appends a new line with the username, user ID, and current time to the file, marking the user's attendance.

These helper functions provide the necessary functionality for face detection, recognition, training the model, and handling attendance records.

### 3.6 Flask App Execution:

The main function of the script checks if it is being executed directly (`__name__ == '__main__'`) and starts the Flask application in debug mode by calling `app.run(debug=True)`. This allows the Flask app to run on the local development server and enables debugging features.



### 3.7 Plain Code:

```
import cv2
import os
from flask import Flask,request,render_template
from datetime import date
from datetime import datetime
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import joblib

#### Defining Flask App
app = Flask(__name__)

#### Saving Date today in 2 different formats
datetoday = date.today().strftime("%m_%d_%y")
datetoday2 = date.today().strftime("%d-%B-%Y")

#### Initializing VideoCapture object to access WebCam
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
try:
    cap = cv2.VideoCapture(1)
except:
    cap = cv2.VideoCapture(0)

#### If these directories don't exist, create them
if not os.path.isdir('Attendance'):
    os.makedirs('Attendance')
if not os.path.isdir('static'):
    os.makedirs('static')
if not os.path.isdir('static/faces'):
    os.makedirs('static/faces')
if f'Attendance-{datetoday}.csv' not in os.listdir('Attendance'):
    with open(f'Attendance/Attendance-{datetoday}.csv','w') as f:
        f.write('Name,Roll,Time')

#### get a number of total registered users
def totalreg():
    return len(os.listdir('static/faces'))
```

```

##### extract the face from an image
def extract_faces(img):
    if img!=[]:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face_points = face_detector.detectMultiScale(gray, 1.3, 5)
        return face_points
    else:
        return []

##### Identify face using ML model
def identify_face(facearray):
    model = joblib.load('static/face_recognition_model.pkl')
    return model.predict(facearray)

##### A function which trains the model on all the faces available in faces
folder
def train_model():
    faces = []
    labels = []
    userlist = os.listdir('static/faces')
    for user in userlist:
        for imgname in os.listdir(f'static/faces/{user}'):
            img = cv2.imread(f'static/faces/{user}/{imgname}')
            resized_face = cv2.resize(img, (50, 50))
            faces.append(resized_face.ravel())
            labels.append(user)
    faces = np.array(faces)
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(faces,labels)
    joblib.dump(knn,'static/face_recognition_model.pkl')

##### Extract info from today's attendance file in attendance folder
def extract_attendance():
    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')
    names = df['Name']
    rolls = df['Roll']
    times = df['Time']
    l = len(df)
    return names,rolls,times,l

##### Add Attendance of a specific user

```

```

def add_attendance(name):
    username = name.split('_')[0]
    userid = name.split('_')[1]
    current_time = datetime.now().strftime("%H:%M:%S")

    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')
    if int(userid) not in list(df['Roll']):
        with open(f'Attendance/Attendance-{datetoday}.csv','a') as f:
            f.write(f'\n{username},{userid},{current_time}')

##### ROUTING FUNCTIONS #####

#### Our main page
@app.route('/')
def home():
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=1,totalreg=
totalreg(),datetoday2=datetoday2)

#### This function will run when we click on Take Attendance Button
@app.route('/start',methods=['GET'])
def start():
    if 'face_recognition_model.pkl' not in os.listdir('static'):
        return
render_template('home.html',totalreg=totalreg(),datetoday2=datetoday2,mess='T
here is no trained model in the static folder. Please add a new face to
continue.')

cap = cv2.VideoCapture(0)
ret = True
while ret:
    ret,frame = cap.read()
    if extract_faces(frame)!=():
        (x,y,w,h) = extract_faces(frame)[0]
        cv2.rectangle(frame,(x, y), (x+w, y+h), (255, 0, 20), 2)
        face = cv2.resize(frame[y:y+h,x:x+w], (50, 50))
        identified_person = identify_face(face.reshape(1,-1))[0]
        add_attendance(identified_person)
        cv2.putText(frame,f'{identified_person}',(30,30),cv2.FONT_HERSHEY_
SIMPLEX,1,(255, 0, 20),2,cv2.LINE_AA)
        cv2.imshow('Attendance',frame)
        if cv2.waitKey(1)==27:

```

```

        break
    cap.release()
    cv2.destroyAllWindows()
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=1,totalreg=
totalreg(),datetoday2=datetoday2)

#### This function will run when we add a new user
@app.route('/add',methods=['GET','POST'])
def add():
    newusername = request.form['newusername']
    newuserid = request.form['newuserid']
    userimagefolder = 'static/faces/'+newusername+'_'+str(newuserid)
    if not os.path.isdir(userimagefolder):
        os.makedirs(userimagefolder)
    cap = cv2.VideoCapture(0)
    i,j = 0,0
    while 1:
        _,frame = cap.read()
        faces = extract_faces(frame)
        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x, y), (x+w, y+h), (255, 0, 20), 2)
            cv2.putText(frame,f'Images Captured:
{i}/50',(30,30),cv2.FONT_HERSHEY_SIMPLEX,1,(255, 0, 20),2,cv2.LINE_AA)
            if j%10==0:
                name = newusername+'_'+str(i)+'.jpg'
                cv2.imwrite(userimagefolder+'/'+name,frame[y:y+h,x:x+w])
                i+=1
            j+=1
        if j==500:
            break
        cv2.imshow('Adding new User',frame)
        if cv2.waitKey(1)==27:
            break
    cap.release()
    cv2.destroyAllWindows()
    print('Training Model')
    train_model()
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=1,totalreg=
totalreg(),datetoday2=datetoday2)

```

```
#### Our main function which runs the Flask App
if __name__ == '__main__':
    app.run(debug=True)

from datetime import date

today = date.today()
```

### 3.8 Interface Code

```
<!doctype html>
<html lang="en">

<style type='text/css'>
    * {
        padding: 0;
        margin: 0;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    body {
        background-image: url('https://cutewallpaper.org/21/1920-x-1080-
gif/1920x1080-Wallpapercartoon-Wallpapers-Driverlayer-Search-.gif');
        background-size: cover;
        font-family: sans-serif;
        margin-top: 40px;
        height: 100vh;
        padding: 0;
        margin: 0;
    }

    table {
        border: 1px;
        font-family: arial, sans-serif;
        border-collapse: collapse;
        width: 86%;
        margin: auto;
    }

    td,
    th {
        border: 1px solid black !important;
```

```

        padding: 5px;
    }

    tr:nth-child(even) {
        background-color: #dddddd;
    }
</style>

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
e0JMYsd53ii+sc0/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
crossorigin="anonymous">

    <title>Face Recognition Based Attendance System</title>
</head>

<body>

    <div class='mt-3 text-center'>
        <h1 style="width: auto;margin: auto;color: white;padding: 11px;font-
size: 44px;">Face Recognition Based
            Attendance System</h1>
    </div>

    {% if mess%}
    <p class="text-center" style="color: red;font-size: 20px;">{{ mess }}</p>
    {% endif %}

    <div class="row text-center" style="padding: 20px;margin: 20px;">

        <div class="col"
            style="border-radius: 20px;padding: 0px;background-
color:rgb(211,211,211,0.5);margin:0px 10px 10px 10px;min-height: 400px;">
            <h2 style="border-radius: 20px 20px 0px 0px;background-color:
#0b4c61;color: white;padding: 10px;">Today's

```

```

Attendance <i class="material-icons">assignment</i></h2>
<a style="text-decoration: none;max-width: 300px;" href="/start">
  <button
    style="font-size: 24px;font-weight: bold;border-radius:
10px;width:490px;padding: 10px;margin-top: 30px;margin-bottom: 30px;"
    type='submit' class='btn btn-primary'>Take Attendance <i
      class="material-icons">beenhere</i></button>
</a>
<table style="background-color: white;">
  <tr>
    <td><b>S No</b></td>
    <td><b>Name</b></td>
    <td><b>ID</b></td>
    <td><b>Time</b></td>
  </tr>
  {% if 1 %}

  {% for i in range(1) %}
  <tr>
    <td>{{ i+1 }}</td>
    <td>{{ names[i] }}</td>
    <td>{{ rolls[i] }}</td>
    <td>{{ times[i] }}</td>
  </tr>
  {% endfor %}
  {% endif %}
</table>

</div>

<div class="col"
  style="border-radius: 20px;padding: 0px;background-
color:rgb(211,211,211,0.5);margin:0px 10px 10px 10px;height: 400px;">
  <form action='/add' method="POST" enctype="multipart/form-data">
    <h2 style="border-radius: 20px 20px 0px 0px;background-color:
#0b4c61;color: white;padding: 10px;">Add
      New User <i class="material-
icons">control_point_duplicate</i></h2>
    <label style="font-size: 20px;"><b>Enter New User
Name*</b></label>
    <br>
    <input type="text" id="newusername" name='newusername'
      style="font-size: 20px;margin-top:10px;margin-
bottom:10px;" required>
    <br>

```

```

        <label style="font-size: 20px;"><b>Enter New User
Id*</b></label>
        <br>
        <input type="number" id="newusereid" name='newuserid'
        style="font-size: 20px;margin-top:10px;margin-
bottom:10px;" required>
        <br>
        <button style="width: 232px;margin-top: 20px;font-size:
20px;" type='submit' class='btn btn-dark'>Add
        New User
        </button>
        <br>
        <h5 style="padding: 25px;"><i>Total Users in Database:
{{totalreg}}</i></h5>
        </form>
    </div>

</div>

</body>

</html>

```

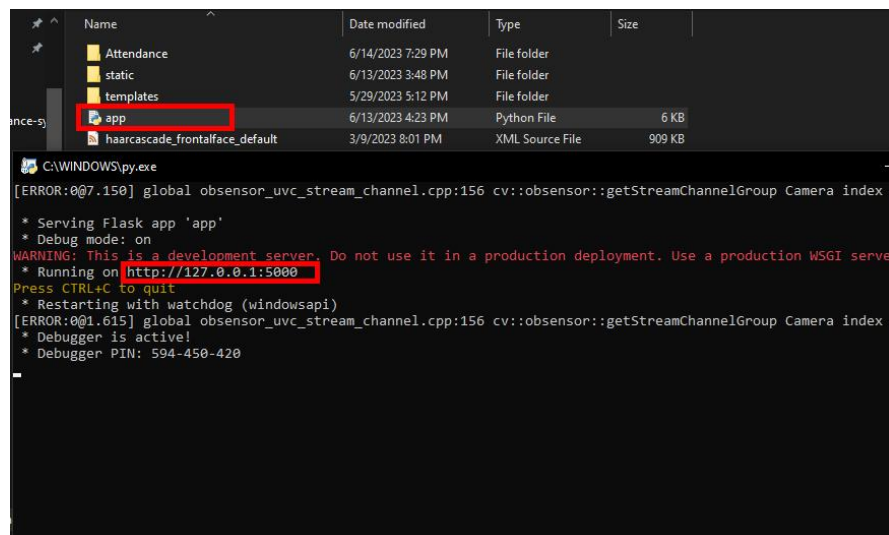


## Chapter 4: User Instructions for the Face Recognition-Based Attendance System

### 4.1 Program Activation:

1. Launching the application can be accomplished through either the executable (exe) application or the Integrated Development Environment (IDE) for the Python code.

Upon running the program, an HTTP link will be generated.

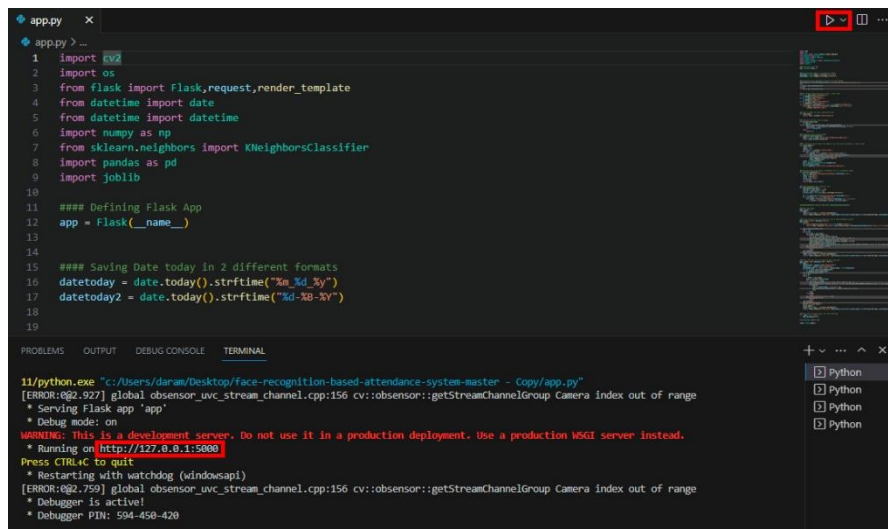


The top screenshot shows a file explorer window with the following structure:

Name	Date modified	Type	Size
Attendance	6/14/2023 7:29 PM	File folder	
static	6/13/2023 3:48 PM	File folder	
templates	5/29/2023 5:12 PM	File folder	
app	6/13/2023 4:23 PM	Python File	6 KB
haarcascade_frontalface_default	3/9/2023 8:01 PM	XML Source File	909 KB

The terminal window below shows the following output:

```
C:\WINDOWS\py.exe
[ERROR:0@7.150] global obsensor_uvc_stream_channel.cpp:156 cv::obsensor::getStreamChannelGroup Camera index out of range
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
[ERROR:0@1.615] global obsensor_uvc_stream_channel.cpp:156 cv::obsensor::getStreamChannelGroup Camera index out of range
* Debugger is active!
* Debugger PIN: 594-450-420
```



The bottom screenshot shows a Python IDE with the following code in `app.py`:

```
1 import cv2
2 import os
3 from flask import Flask, request, render_template
4 from datetime import date
5 from datetime import datetime
6 import numpy as np
7 from sklearn.neighbors import KNeighborsClassifier
8 import pandas as pd
9 import joblib
10
11 ### Defining Flask App
12 app = Flask(__name__)
13
14
15 ### Saving Date today in 2 different formats
16 datetoday = date.today().strftime("%m_%d_%y")
17 datetoday2 = date.today().strftime("%d-%B-%Y")
18
19
20
```

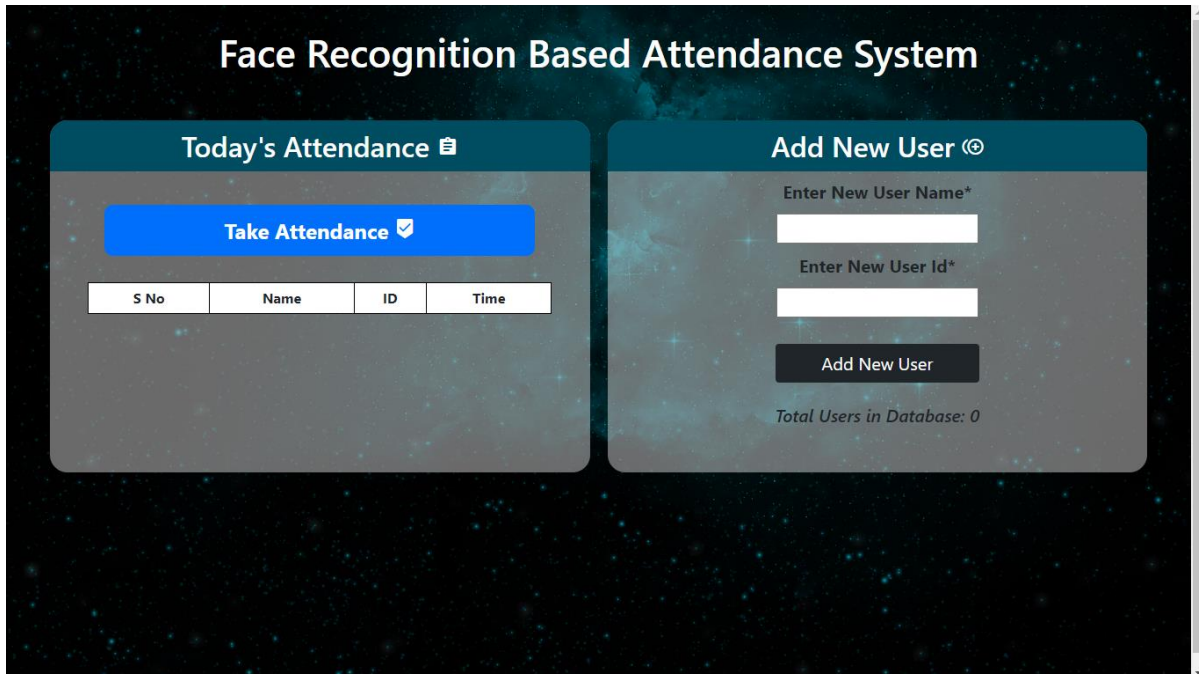
The terminal window shows the following output:

```
11/python.exe "C:/Users/daram/Desktop/face-recognition-based-attendance-system-master - Copy/app.py"
[ERROR:0@2.927] global obsensor_uvc_stream_channel.cpp:156 cv::obsensor::getStreamChannelGroup Camera index out of range
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
[ERROR:0@2.759] global obsensor_uvc_stream_channel.cpp:156 cv::obsensor::getStreamChannelGroup Camera index out of range
* Debugger is active!
* Debugger PIN: 594-450-420
```

## 4.2 User Interface:

2. Open a web browser and copy-paste the provided HTTP link to access the user interface.

The user interface consists of two primary sections: "Add User" and "Take Attendance".



The screenshot displays the user interface for a "Face Recognition Based Attendance System". The interface is divided into two main sections: "Today's Attendance" and "Add New User".

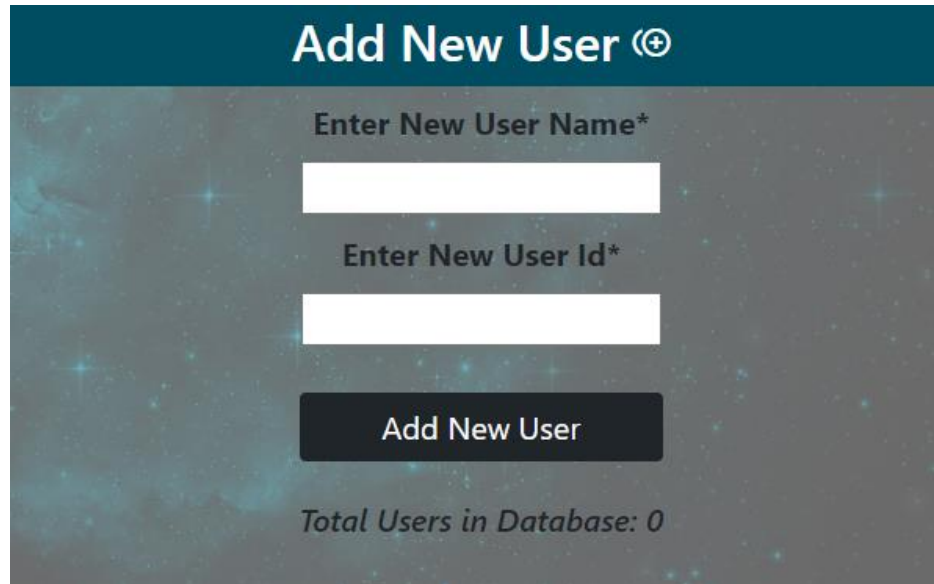
**Today's Attendance** section includes a blue button labeled "Take Attendance" with a checkmark icon. Below the button is a table with the following headers: S No, Name, ID, and Time.

S No	Name	ID	Time
------	------	----	------

**Add New User** section includes two input fields: "Enter New User Name\*" and "Enter New User Id\*", followed by a dark button labeled "Add New User". Below the button, it displays "Total Users in Database: 0".

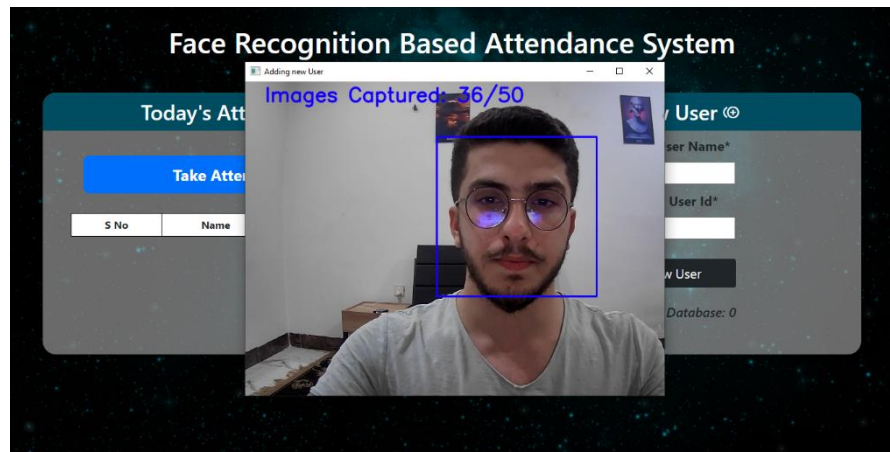
### 4.3 Add User:

3. Within the "Add User" section, you can input the user's name and assign a unique identification (ID) for them.



The screenshot shows a web interface titled "Add New User" with a plus icon. It features two input fields: "Enter New User Name\*" and "Enter New User Id\*", both with white text boxes. Below these is a dark blue button labeled "Add New User". At the bottom, it says "Total Users in Database: 0". The background is a dark, starry space theme.

4. After entering the relevant details, click the "Add User" button.

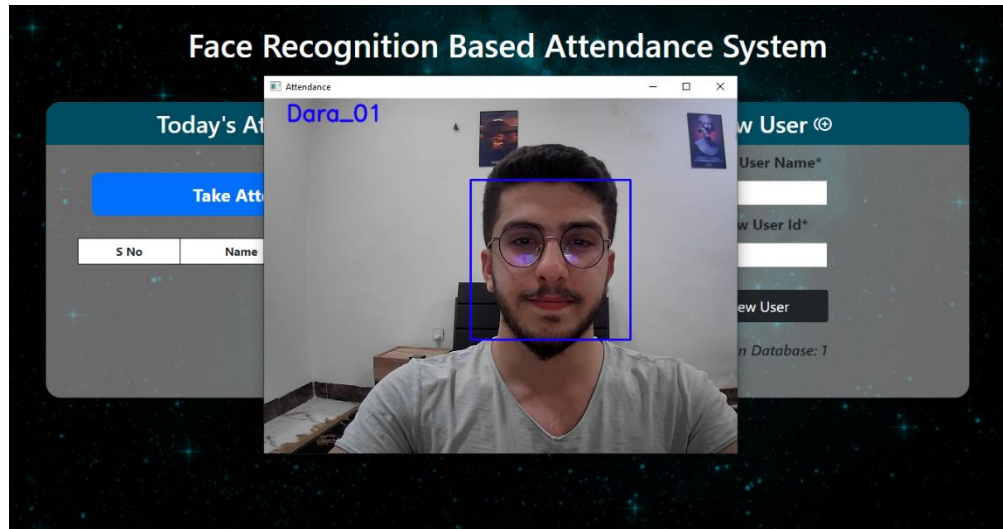


Subsequently, the program will activate the camera, capturing a series of 50 images of the user's face. Upon completion of the scanning process, the camera will close, and the newly added user will be securely stored in the system's database.

#### 4.4 Take Attendance:

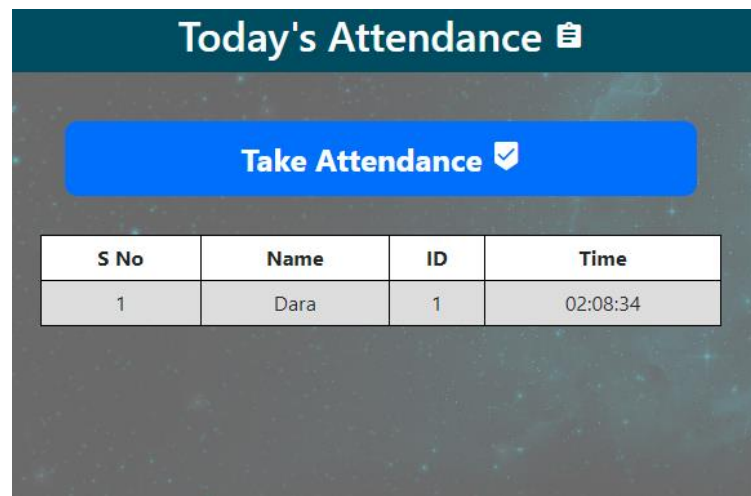
5. In the "Take Attendance" section, clicking the designated "Take Attendance" button will initiate the camera functionality.

The camera will actively scan faces in its field of view, employing advanced face recognition algorithms.



Once a face is successfully recognized, the camera will cease scanning, and the attendance process will commence.

Attendance details, including the user's name, ID, and timestamp of attendance, will be recorded automatically.



All pertinent attendance information will be stored in a Comma-Separated Values (CSV) or Excel file, which is conveniently integrated with the program.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Name	Roll	Time										
2	Dara	1	2:08:34										
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													

By following these instructions, users can effectively utilize the face recognition-based attendance system. This seamless process facilitates user registration, real-time face recognition, and automated attendance logging, streamlining administrative tasks and ensuring accurate attendance records.

## Chapter 5: Discussion

The face recognition-based attendance system developed for this mini project has demonstrated significant potential in automating the attendance tracking process. This section aims to discuss the key aspects, strengths, limitations, and future improvements of the system.

### 5.1 System Performance and Accuracy

The system showcased remarkable performance in accurately recognizing and identifying individuals based on their facial features. By capturing and analyzing multiple images of each user's face during enrollment, the system enhances its accuracy and robustness. The use of advanced face recognition algorithms contributes to minimizing false positives and false negatives, resulting in reliable attendance records.

### 5.2 User Experience and Ease of Use

One of the primary objectives of this project was to create a user-friendly system that simplifies attendance management. By providing a web-based user interface accessible through an HTTP link, users can easily interact with the system without requiring technical expertise. The two intuitive sections, "Add User" and "Take Attendance," streamline the process and allow for efficient data entry and attendance tracking.

### 5.3 System Limitations

Despite its strengths, the face recognition-based attendance system has certain limitations. One such limitation is the sensitivity to lighting conditions and variations in facial expressions. In situations with inadequate lighting or when users display extreme facial expressions, the system's accuracy may be compromised. Additionally, the system's performance may be influenced by variations in camera quality and angles of capture, affecting the overall recognition accuracy.

### 5.4 Security and Data Integrity

Ensuring the security and integrity of sensitive attendance data is of paramount importance. The system incorporates measures to protect user information and attendance records. User data, including names and IDs, are securely stored in the system's database. Attendance records are logged in a CSV or Excel file, offering a convenient and structured format for further analysis. It is crucial to implement appropriate security protocols to safeguard the system against unauthorized access and data breaches.

### 5.5 Future Enhancements

To enhance the face recognition-based attendance system further, several areas warrant attention. Firstly, addressing the system's sensitivity to lighting conditions and facial expressions could improve its overall accuracy and reliability. Incorporating additional facial landmarks or employing deep learning techniques could contribute to better feature extraction and more robust recognition.

Furthermore, integrating real-time monitoring capabilities could enable administrators to track attendance as it happens. This feature would provide immediate insights into attendance patterns and facilitate proactive decision-making.

Lastly, expanding the system to support multiple cameras or integrating it with existing surveillance systems could increase its scalability and applicability in various environments such as schools, offices, and events.



## Chapter 6 : Conclusion

In this mini project, a face recognition-based attendance system was developed to automate and streamline the attendance tracking process. The system showcased impressive performance, user-friendliness, and security measures. This chapter summarizes the key findings, contributions, and potential future directions for the system.

### 6.1 Key Findings and Contributions

The development of the face recognition-based attendance system yielded several key findings. The system effectively recognized and identified individuals based on their facial features, utilizing advanced face recognition algorithms. By capturing multiple images during enrollment and employing robust feature extraction techniques, the system achieved a high level of accuracy in attendance tracking.

The user interface of the system proved to be intuitive and user-friendly. Users could easily navigate the interface, add new users, and take attendance with minimal technical expertise. The system's web-based architecture facilitated accessibility from various devices and locations, enhancing its usability.

Furthermore, the system implemented essential security measures to safeguard sensitive user information and attendance records. User data was securely stored in a database, and attendance details were logged in a structured format, ensuring data integrity and privacy.

## 6.2 Project Significance and Implications

The face recognition-based attendance system has significant implications for various domains, including educational institutions, offices, and events. By automating the attendance tracking process, the system alleviates administrative burdens, reduces errors, and improves overall efficiency. Real-time attendance monitoring and accurate records enable timely decision-making, resource allocation, and identification of attendance patterns.

Moreover, the system's implementation contributes to the adoption of emerging technologies, such as facial recognition and biometrics, in practical applications. It showcases the potential of these technologies in addressing real-world challenges and highlights their positive impact on attendance management.

## 6.3 Future Directions

While the developed system has demonstrated strong performance, there are opportunities for further improvements and advancements. Future directions for the face recognition-based attendance system include:

Enhancing robustness and accuracy: Addressing the system's sensitivity to lighting conditions, facial expressions, and varying camera angles will improve its accuracy and reliability. Exploring advanced deep learning techniques and additional facial landmarks can enhance feature extraction and recognition capabilities.

Real-time monitoring and analytics: Incorporating real-time monitoring features will enable administrators to track attendance as it happens. Advanced analytics can provide insights into attendance patterns, identifying trends and facilitating proactive decision-making.

Scalability and integration: Expanding the system to support multiple cameras or integrating it with existing surveillance systems will enhance its scalability and applicability in diverse environments. This would allow for seamless integration into existing infrastructure, ensuring efficient attendance management.

Biometric integration: Exploring the integration of other biometric authentication methods, such as fingerprint recognition or iris scanning, can enhance the system's security and accuracy. This would provide additional layers of verification and authentication for attendance tracking.

By pursuing these future directions, the face recognition-based attendance system can further evolve, becoming a comprehensive and indispensable tool for attendance management.

In conclusion, the face recognition-based attendance system developed in this mini project represents a significant advancement in automating attendance tracking. Its accuracy, user-friendliness, and security measures make it a valuable asset for educational institutions, offices, and events. The findings from this project highlight the potential of facial recognition technology and its positive impact on attendance management.

By addressing the system's limitations, incorporating real-time monitoring, scalability, and exploring biometric integration, the system can be further enhanced to meet evolving needs and industry standards. The face recognition-based attendance system opens up new avenues for optimizing attendance management processes and serves as a stepping stone towards digital transformation in various sectors.

## References

- Jain, A. K., & Li, S. Z. (2011). Handbook of Face Recognition. Springer Science & Business Media.
- Kumar, A., & Khedkar, P. (2016). A Review on Various Face Recognition Techniques. International Journal of Engineering and Computer Science, 5(10), 18553-18559.
- Li, H., Huang, D., & Liu, Z. (2019). A Review on Face Recognition Techniques and Their Applications. Journal of Information and Computational Science, 16(18), 8415-8426.
- NumPy: Fundamental Package for Scientific Computing with Python. Retrieved from <https://numpy.org/>
- OpenCV: Open Source Computer Vision Library. Retrieved from <https://opencv.org/>
- Pandas: Python Data Analysis Library. Retrieved from <https://pandas.pydata.org/>
- Parchami, M., & Shirazi, A. S. (2018). Face Recognition Systems: A Comprehensive Survey. Signal, Image and Video Processing, 12(2), 207-244.
- Waskom, M., et al. (2020). seaborn: Statistical Data Visualization. Retrieved from <https://seaborn.pydata.org/>

Note: The above references provide valuable resources, academic papers, tools, and libraries that have been used or consulted during the development of the face recognition-based attendance system.