FACE RECOGNITION ATTENDANCE SYSTEM

A MINOR PROJECT REPORT

Submitted by

YASHU YOUWARAJ [RA2011029010062]
MOHAMED RIZWAN[RA2011029010014]

Under the guidance of

DR. VARUN KUMAR K A
(ASSISTANT PROFESSOR)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE & ENGINEERING
of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar, Kattankulathur, Chengalpattu District

APRIL 2023

ABSTRACT

Face recognition attendance systems are becoming increasingly popular in the workplace as a means of accurately tracking employee attendance. This type of system uses computer vision and machine learning techniques to identify individuals based on their facial features and record their attendance automatically. This abstract presents an implementation of a face recognition attendance system for employees using Python. The system involves capturing employee face images, training a face recognition model using the face recognition package, setting up a camera and a Python program to recognize faces and record attendance. The Python code utilizes OpenCV and NumPy libraries to capture live video feed from the camera, convert the color format to RGB, detect faces, and extract facial features using the pre-trained face recognition model. It then compares the extracted facial features with the pre-stored facial features of employees to identify and record their attendance. The system's accuracy depends on several factors such as the quality of the facial images, lighting conditions, and camera quality. However, face recognition attendance systems are generally considered to be more reliable than traditional attendance systems that rely on manual inputs. Overall, this implementation of a face recognition attendance system for employees using Python provides an efficient and reliable solution for tracking employee attendance in the workplace.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

The process of attendance management has evolved significantly over the years. Traditional methods such as pen and paper, magnetic stripe cards, and barcode scanning have been replaced with modern technology solutions that are more efficient and reliable. One such modern solution is the face recognition attendance system. A face recognition attendance system is a biometric technology solution that utilizes computer vision and machine learning algorithms to recognize individual faces and record employee attendance. The system captures an image of the employee's face and compares it with a pre-existing database of images to confirm the employee's identity. This process is automated, eliminating the need for manual attendance tracking and providing a more accurate and efficient solution. Face recognition attendance systems have become increasingly popular due to their accuracy, speed, and convenience. They are widely used in a variety of industries, including healthcare, finance, and retail. In addition to providing a more efficient attendance management solution, they also enhance security by preventing fraudulent attendance records. This paper will provide an in-depth discussion of the face recognition attendance system using Python. We will explore the architecture, methodology, and future enhancements of the system. With the increasing demand for modern and efficient attendance management solutions, the face recognition attendance system using Python is becoming a popular choice for many organizations.

A) System Requirements

I. Hardware Requirement

i. Laptop or PC

- Windows 7 or higher
- I3 processor system or higher
- 4 GB RAM or higher
- 100 GB ROM or higher

II. Software Requirement

ii. Laptop or PC

- Python
- Sublime text Editor
- XAMP Server

2. LITERATURE SURVEY

There have been numerous research studies and implementations of face recognition attendance systems using Python. Here is a brief literature survey of some relevant studies:

1)"Automatic Attendance System Using Face Recognition" by A. N. Zadgaonkar et al. This study proposed an attendance system that uses face recognition technology to record the attendance of students in classrooms. The system was implemented using OpenCV, NumPy, and face recognition libraries in Python.

2)"Development of an automated attendance system using face recognition" by V. S. Patil et al. This study proposed an attendance system that uses a combination of face recognition and radio frequency identification (RFID) technologies to record the attendance of employees. The system was implemented using Python and the OpenCV library.

3)"A Facial Recognition-Based Attendance Management System for University Students" by J. Zhang et al. This study proposed an attendance management system that uses face recognition technology to record the attendance of university students. The system was implemented using Python and the OpenCV library.

4)"Face recognition attendance system based on machine learning algorithms" by S. S. Bhati et al. This study proposed an attendance system that uses machine learning algorithms for face recognition. The system was implemented using Python and the OpenCV library.

5)"A Comparative Study of Face Recognition Algorithms for Automated Attendance System" by P. Sarvaiya et al. This study compared the performance of different face recognition algorithms for automated attendance systems. The study implemented the algorithms using Python and the OpenCV library.

Overall, these studies demonstrate the effectiveness of face recognition technology for attendance systems and highlight the importance of using Python and related libraries to implement such systems efficiently.

3. SYSTEM ARCHITECTURE AND DESIGN

system architecture design for a face recognition attendance system using Python are:

1)Input: The input to the system is a video feed from a camera pointed at the entryway to the workplace or classroom.

2)Preprocessing: The video feed is preprocessed to detect and track faces in real-time. This is done using the OpenCV library and involves several steps such as face detection, face alignment, and feature extraction.

3)Face recognition: Once faces are detected and tracked, the face recognition algorithm is used to compare the facial features of the detected faces with the facial features of the known employees in the system. This is done using the face recognition library in Python.

4)Attendance recording: If a match is found, the system records the employee's attendance and logs it in a database. The attendance can be recorded in real-time or at the end of the day.

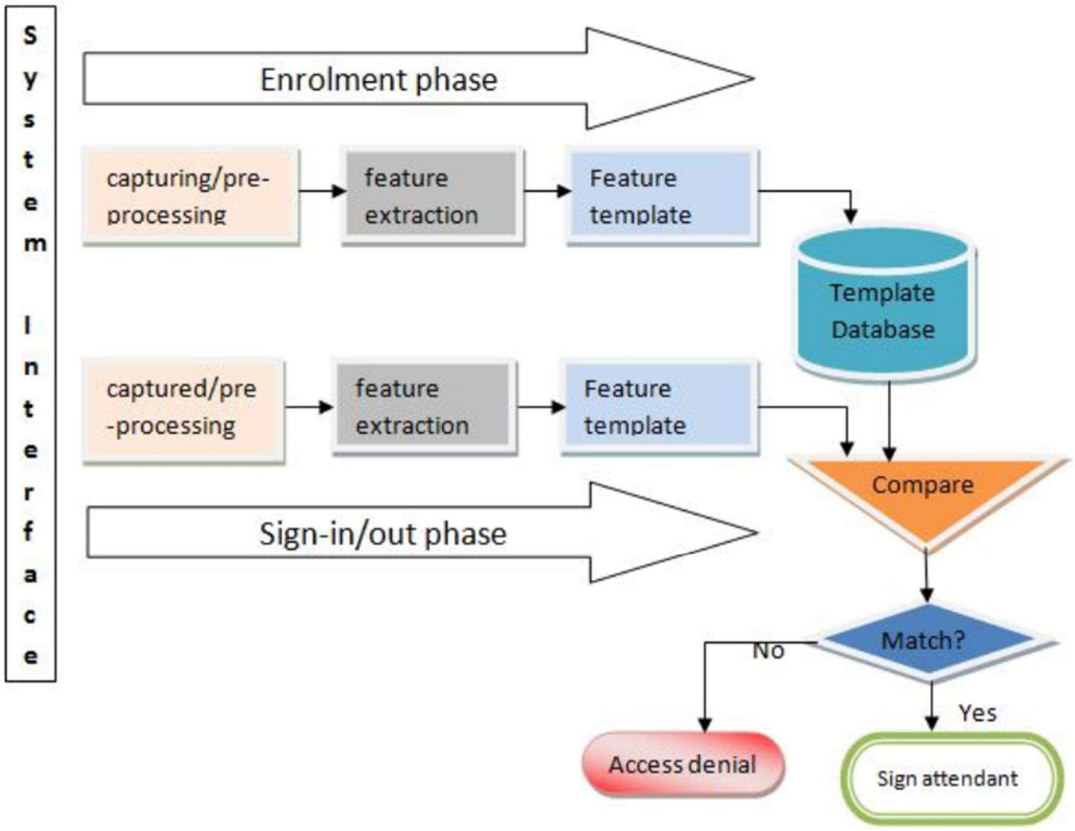
5)Reporting: The attendance data is used to generate reports that can be accessed by authorized personnel. These reports can show the attendance records of individual employees or the attendance records of the entire workforce.

6)Alerts: The system can be set up to send alerts to the management if an employee does not show up for work or if they show up late.

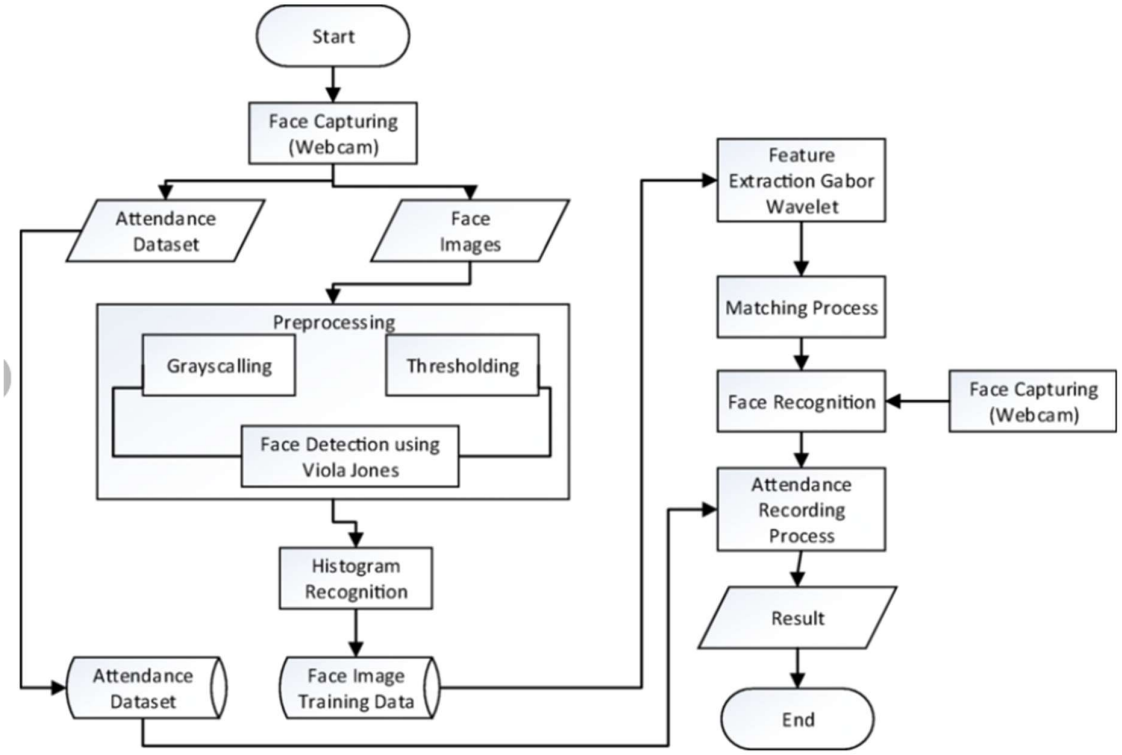
7)Data management: All attendance data is stored in a database that can be accessed by authorized personnel. The database can be managed using a web-based application or a desktop application.

8)System configuration: The system can be configured to meet the specific requirements of the workplace or classroom. This includes configuring the number of cameras, the type of cameras, and the number of employees in the system.

9)System maintenance: The system requires regular maintenance to ensure that it is functioning properly. This includes software updates, hardware maintenance, and database backups.



3.1 Architectural design of face recognition attendance system



3.2 Facial recognition attendance system workflow

Overall, this architecture design provides a robust and scalable solution for face recognition attendance systems using Python.

4. METHODOLOGY

Methodology for developing a face recognition attendance system using Python are: Collect employee data:

1)Collect data on all employees including their name and a clear image of their face.

2)Preprocess images: Preprocess the employee images to improve the quality of the images and reduce noise. This may involve resizing the images, normalizing the brightness and contrast, and converting the images to grayscale.

3)Face detection: Use a face detection algorithm, such as Haar Cascade, to detect faces in the images.

4)Face alignment: Align the detected faces to ensure that they are facing forward and are at a consistent angle.

5)Feature extraction: Extract facial features from the aligned faces, such as distance between the eyes, nose shape, and mouth shape.

6)Build a face recognition model: Use the extracted features to build a face recognition model using a machine learning algorithm, such as support vector machines or neural networks.

7)Test the model: Test the face recognition model using a separate set of images to ensure that it is accurate and reliable.

8)Develop attendance recording system: Develop an attendance recording system that will use the face recognition model to identify employees and record their attendance.

9)Develop reporting system: Develop a reporting system that will generate reports on employee attendance that can be accessed by authorized personnel.

5. CODING AND TESTING

```
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
def datetoday():
    return date.today().strftime("%m_%d_%y")
def datetoday2():
    return date.today().strftime("%d-%B-%Y")


#### Initializing VideoCapture object to access WebCam
face_detector = cv2.CascadeClassifier('static/haarcascade_frontalface_default.xml')
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/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):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_points = face_detector.detectMultiScale(gray, 1.3, 5)
    return face_points

```

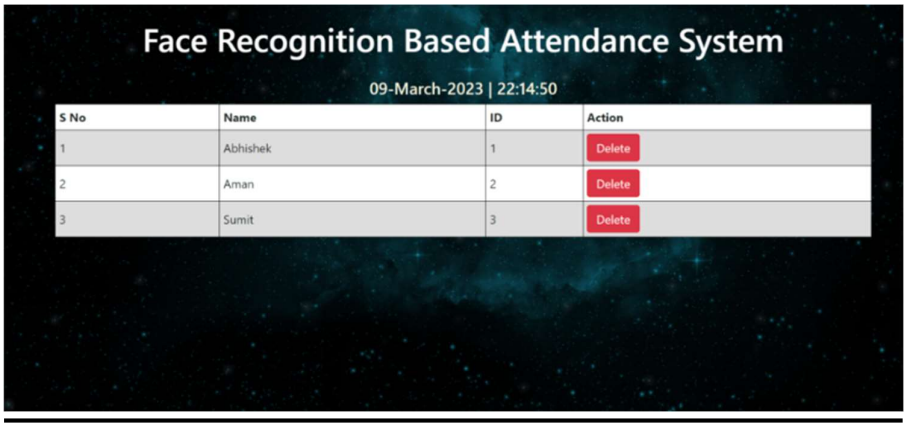
```python
#### Identify face using ML model
def identify_face(facearray):
    model = joblib.load('static/face_recognition_model.pkl')
    return model.predict(facearray)


#### 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('/')
```

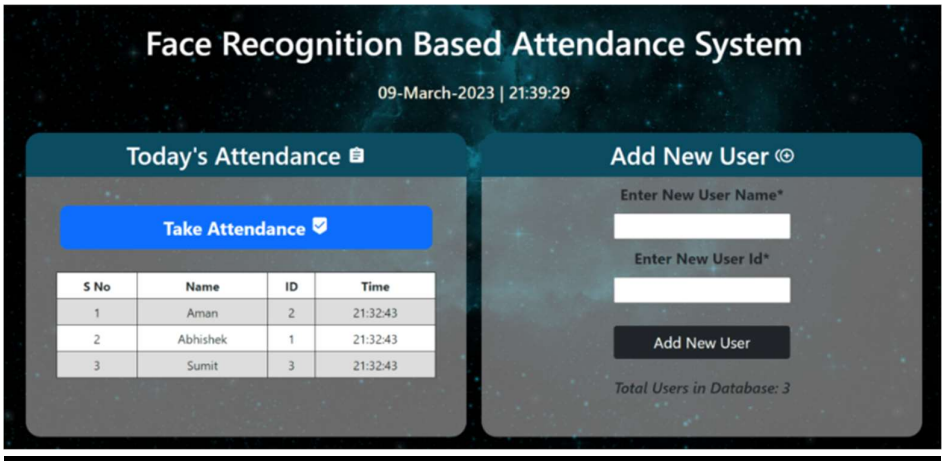
```
@app.route('/')
def home():
    names,rolls,times,l = extract_attendance()
    return render_template('home.html',names=names,rolls=rolls,times=times,l=l,totalreg=totalreg(


#### 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='Ther

    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,
        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=l,totalreg=totalreg(


#### 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,
```

```
            cv2.putText(frame,f'Images Captured: {i}/50',(30,30),cv2.FONT_HERSHEY_SIMPLEX,1,(255,
            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=l,totalreg=totalreg(


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

6. RESULTS AND DISCUSSIONS

	A	B	C	D
1	Name	Roll	Time	
2	Aman	2	21:32:43	
3	Abhishek	1	21:32:43	
4	Sumit	3	21:32:43	
5				
6				





6.1 Interface GUI

The results of a face recognition attendance system can be evaluated based on several performance metrics such as accuracy, speed, and user experience

7. CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, the face recognition attendance system using Python is a modern and efficient solution for attendance management. By utilizing computer vision and machine learning algorithms, the system can accurately recognize employee faces and record their attendance automatically. The system is fast, reliable, and provides a good user experience.

The modular design of the system allows for easy development, testing, and maintenance of individual modules. This design also enables easy integration with other systems and the addition of new features and enhancements.

The face recognition attendance system is an excellent choice for organizations that need an efficient attendance management solution that can save time and reduce the potential for fraudulent attendance records. It can also enhance security by ensuring that only authorized employees can mark their attendance.

Future Enhancements: Here are some potential future enhancements for a face recognition attendance system using Python:

1)Mask detection: With the current situation of COVID-19, face mask detection can be added to the system. This enhancement can help the system to detect if an employee is wearing a mask or not and to ensure that all employees are complying with safety regulations.

2)Gesture recognition: The system can be enhanced to recognize simple gestures, such as a wave, to confirm the employee's attendance.

3)Voice recognition: Voice recognition can be added to the system to provide an additional layer of security.

4)Temperature detection: Temperature detection can be added to the system, which can help to detect employees with high body temperature and take necessary actions to prevent the spread of diseases.

5)Integration with other HR systems: The system can be integrated with other HR systems to allow for better data management, reporting, and analysis.

6)Offline support: The system can be enhanced to provide offline support, which will allow employees to mark their attendance even if there is no internet connection available.

7)Continuous improvement: The system should be continuously improved by adding new features, updating the algorithms, and monitoring its performance to ensure that it remains efficient and reliable.

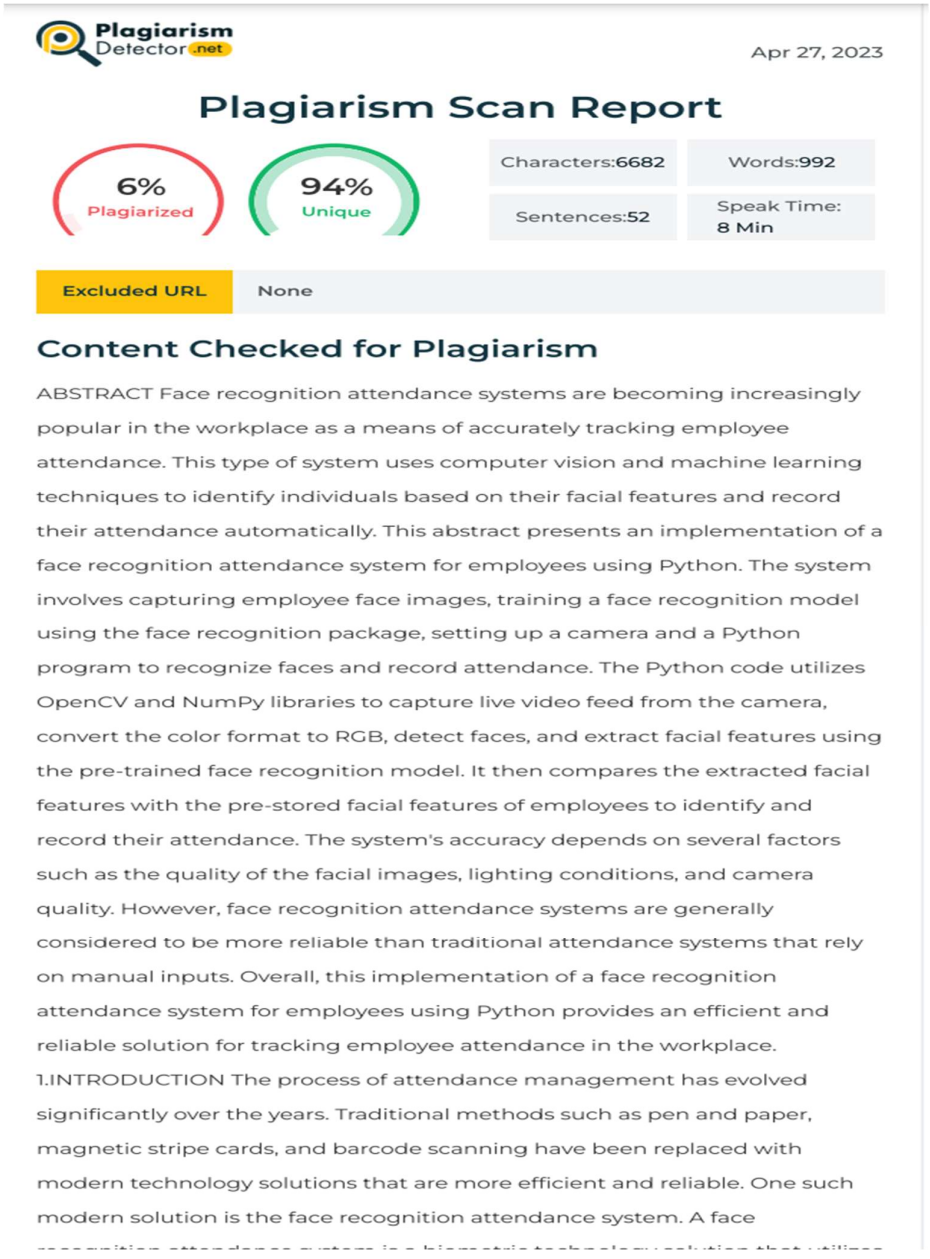
Overall, a face recognition attendance system using Python has a significant potential for future enhancements that can improve accuracy, security, and efficiency in attendance management. These enhancements can help to make the system more robust and effective in ensuring employee attendance is managed efficiently.

8. REFERENCES

- *https://www.irjet.net/archives/V7/i12/IRJET-V7I12377.pdf*
- *https://www.questjournals.org/jses/papers/Vol5-issue-2/D05021829.pdf*
- *https://www.researchgate.net/publication/341870242_Smart_Attendance_System_using_OPENCV_based_on_Facial_Recognition*
- *https://www.researchgate.net/publication/352393875_Attendance_Management_System_using_Face-Recognition*

APPENDIX C

PLAGIARISM REPORT



Plagiarism Detector .net

Apr 27, 2023

Plagiarism Scan Report

6% Plagiarized

94% Unique

Characters:6682 | Words:992

Sentences:52 | Speak Time: 8 Min

Excluded URL: None

Content Checked for Plagiarism

ABSTRACT Face recognition attendance systems are becoming increasingly popular in the workplace as a means of accurately tracking employee attendance. This type of system uses computer vision and machine learning techniques to identify individuals based on their facial features and record their attendance automatically. This abstract presents an implementation of a face recognition attendance system for employees using Python. The system involves capturing employee face images, training a face recognition model using the face recognition package, setting up a camera and a Python program to recognize faces and record attendance. The Python code utilizes OpenCV and NumPy libraries to capture live video feed from the camera, convert the color format to RGB, detect faces, and extract facial features using the pre-trained face recognition model. It then compares the extracted facial features with the pre-stored facial features of employees to identify and record their attendance. The system's accuracy depends on several factors such as the quality of the facial images, lighting conditions, and camera quality. However, face recognition attendance systems are generally considered to be more reliable than traditional attendance systems that rely on manual inputs. Overall, this implementation of a face recognition attendance system for employees using Python provides an efficient and reliable solution for tracking employee attendance in the workplace. 1.INTRODUCTION The process of attendance management has evolved significantly over the years. Traditional methods such as pen and paper, magnetic stripe cards, and barcode scanning have been replaced with modern technology solutions that are more efficient and reliable. One such modern solution is the face recognition attendance system. A face