AI-Based Face Attendance System Using Python

Project Based Learning (PBL) Report

for the course

Artificial Intelligence - 20CS31003

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

K NANDA KISHORE REDDI (21R11A05N1)
U VIJAY VARDHAN (21R11A05R3)
P AJAY KUMAR (21R11A05Q5)

Under the guidance of
E MAHENDER

Department of Computer Science and Engineering
Accredited by NBA

Geethanjali College of Engineering and Technology
(UGC Autonomous)
(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

December -2023

TABLE OF CONTENTS

1. Introduction

1.1 Background and Rationale

Attendance tracking is a fundamental aspect of various domains, including education, corporate environments, and event management. Traditional methods, reliant on manual processes and physical registers, often lead to inaccuracies, inefficiencies, and resource-intensive efforts. The advent of facial recognition technology provides an opportunity to address these challenges and usher in a new era of automated, reliable attendance management.

1.2 Evolution of Attendance Systems

Over the years, attendance systems have evolved from manual methods to more sophisticated technologies. Initially, paper-based registers were commonplace, requiring individuals to physically sign in. Later, electronic systems with barcode scanners and RFID cards gained popularity, offering improved efficiency but still susceptible to issues like proxy attendance.

The emergence of facial recognition technology represents a paradigm shift, offering a contactless and secure method for attendance tracking. This technology leverages advanced machine learning models to identify individuals based on their unique facial features, eliminating the need for physical tokens or manual signatures.

1.3 Significance of Facial Recognition

Facial recognition technology, driven by advancements in deep learning and computer vision, has demonstrated remarkable accuracy in identifying individuals. Its non-intrusive nature and the ability to work in real-time make it an ideal candidate for attendance systems. By harnessing the power of artificial intelligence, the proposed system aims to mitigate the shortcomings of traditional methods and provide a seamless, reliable, and automated solution.

1.4 Motivation for the Project

The motivation behind developing an AI-based face attendance system stems from the need for a modern, efficient, and secure method of attendance tracking. The project aims to capitalize on the capabilities of machine learning and facial recognition to enhance accuracy, reduce administrative overhead, and improve the overall user experience. The potential for widespread adoption across various sectors further underscores the significance of this project.

1.5 Objectives of the Project

The primary objectives of the project are:

- To design a robust face attendance system capable of accurately recognizing individuals.
- To implement a secure client-server architecture for handling multiple users and devices.
- To create a user-friendly interface for both administrators and end-users.
- To integrate machine learning models for facial recognition, ensuring adaptability and scalability.

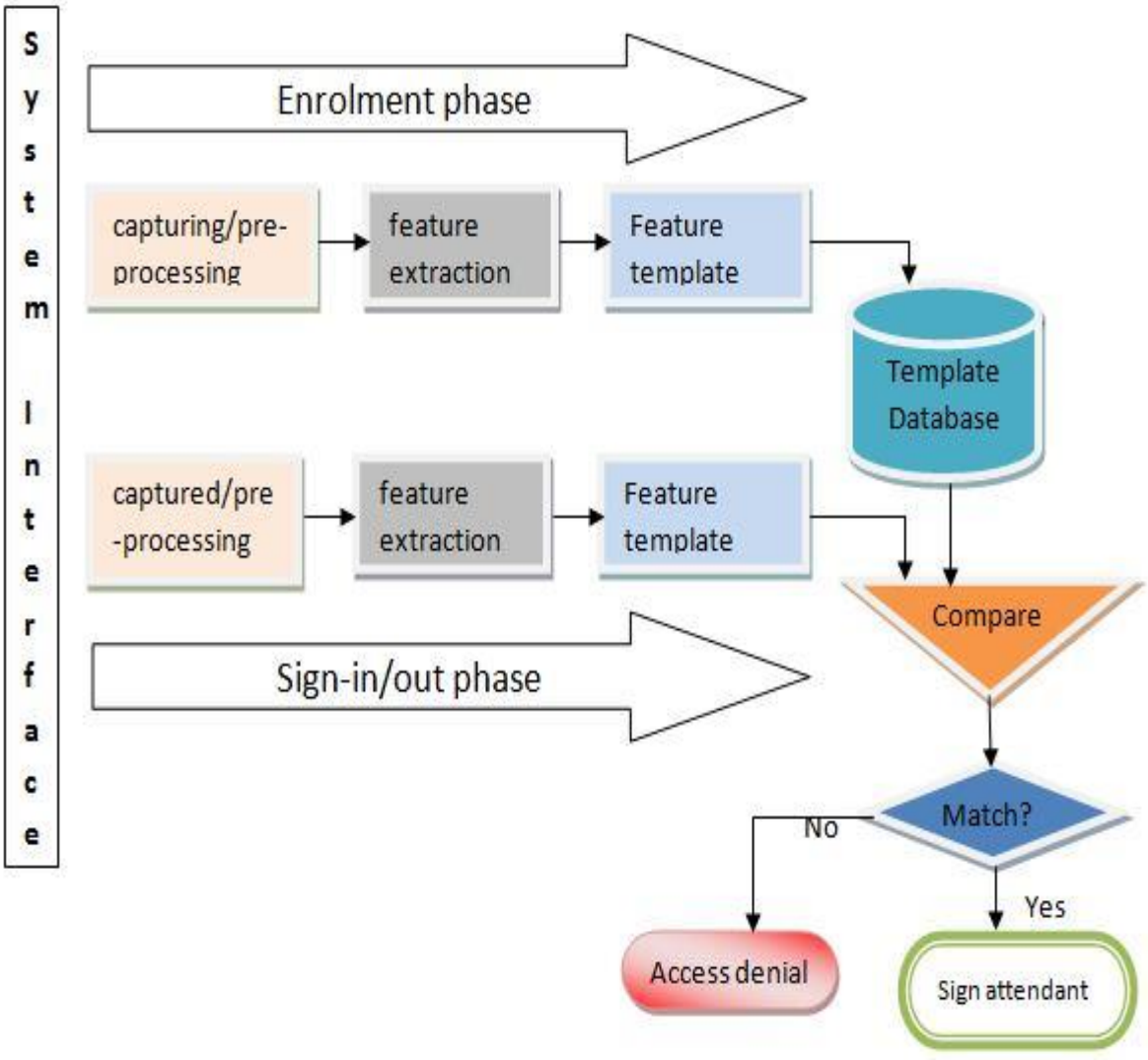
1.6 Scope and Limitations

While the AI-based face attendance system offers numerous benefits, it is essential to acknowledge its scope and limitations. The system is designed to function in controlled environments with proper lighting conditions. Challenges such as variations in facial expressions, occlusions, and privacy concerns may influence the system's performance. Addressing these challenges and understanding the system's limitations are crucial aspects of the project.

2. System Design

2.1 System Architecture

The system follows a client-server architecture, where the client captures facial images, and the server performs facial recognition and attendance tracking. This design ensures scalability, allowing multiple clients to connect to a central server.



2.2 Modules

Data Collection Module:

Responsible for capturing facial images, this module uses OpenCV to access the camera feed and gather the required images for further processing.

Preprocessing Module:

Focused on enhancing image quality, the preprocessing module utilizes OpenCV for image processing techniques such as resizing, normalization, and noise reduction.

Facial Recognition Module:

This module employs a pre-trained deep learning model, such as FaceRecognition or Dlib, to recognize and match faces in the captured images.

Attendance Tracking Module:

The attendance tracking module manages attendance records by updating a secure database. It ensures real-time updates and data integrity.

2.3 Backend Design

The backend, powered by Python, utilizes the tkinter framework to create a UI for the app. The communication between the client and server is established through a secure network connection. In addition to managing data processing and business logic, the backend serves as the backbone of the face attendance system, playing a pivotal role in ensuring the application's robustness. Leveraging Python's versatility, it synchronizes seamlessly with Tkinter, enhancing the user interface's responsiveness.The database, implemented using MySQL, securely stores attendance records. The backend handles tasks such as user verification through facial recognition, attendance record updates, and data validation. It forms a crucial component of the desktop application, orchestrating seamless interactions between the user interface and the database while prioritizing data security and system integrity.

2.4 Working Design

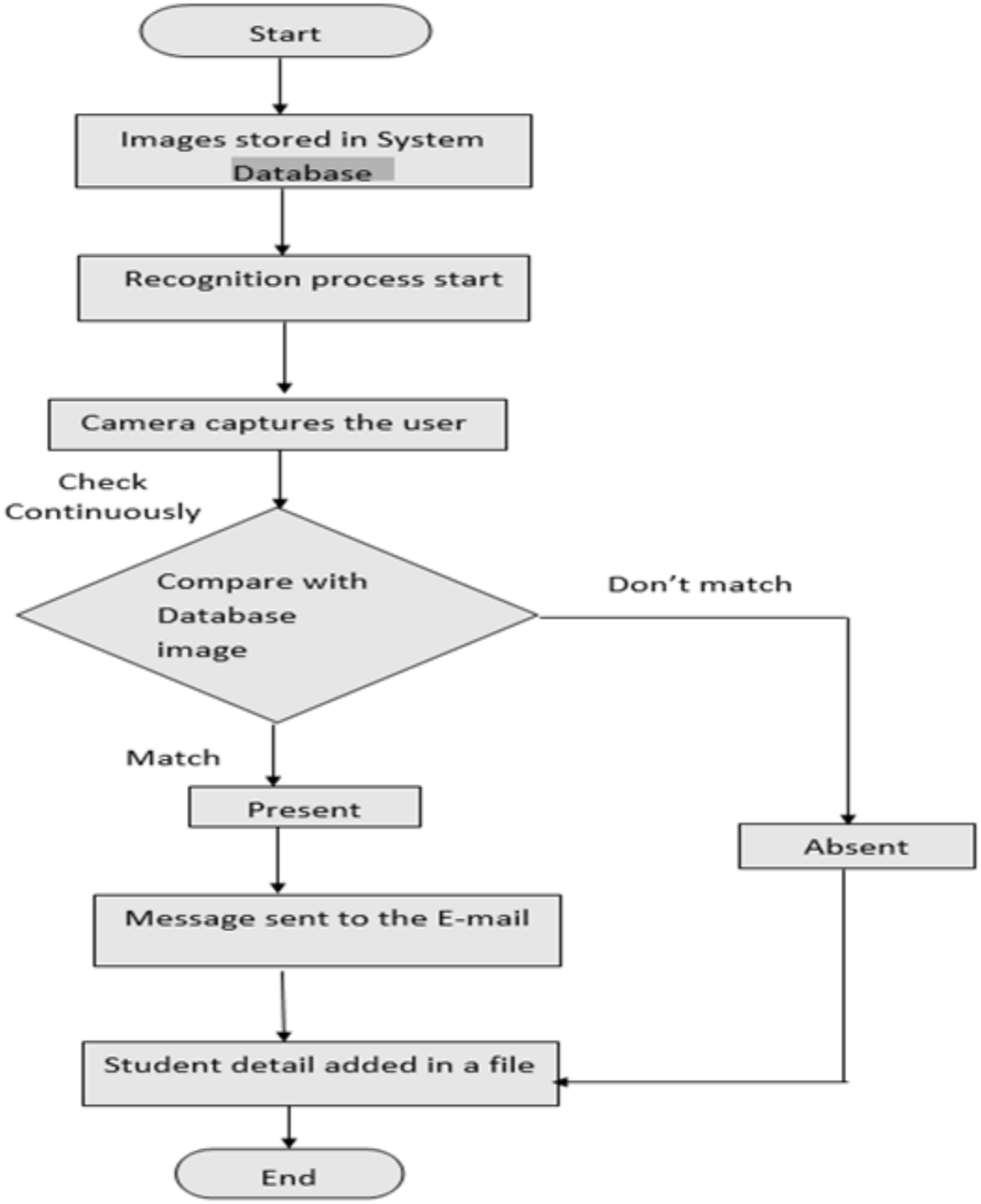


Fig 1 : Work Flow Diagram

3. Implementation

3.1 Modules Implementation

Data Collection Module:

The implementation involves using OpenCV to access the camera feed, capture frames, and store facial images for further processing.

Preprocessing Module:

OpenCV is employed for image preprocessing, applying techniques like resizing, normalization, and Gaussian blur to enhance the quality of facial images.

Facial Recognition Module:

The implementation uses Dlib and OpenCV to perform facial recognition, detecting facial landmarks and matching faces against a pre-trained model.

Attendance Tracking Module:

This module updates the attendance records in the database, ensuring real-time synchronization and data consistency.

3.2 Sample Code

The provided sample code demonstrates the implementation of the facial recognition module using Dlib and OpenCV.

```
import tkinter as tk
from tkinter import messagebox
from tkinter.scrolledtext import ScrolledText
from PIL import Image, ImageTk
import cv2
import face_recognition
import pickle
import time
import mysql.connector
from datetime import datetime, timedelta
import src.attendance

database_config = {
    'host': '',
    'port': 3306,
    'user': 'nandu',
    'password': 'Nk@1412',
    'database': 'FA',
}

mydb = mysql.connector.connect(**database_config)

cursor = mydb.cursor()

class FaceRecognitionApp:
    def __init__(self, root):
```

```
        self.root = root
        self.root.title("Face Recognition App")
        self.my_dict = {}
        self.vid = cv2.VideoCapture(0)
        self.canvas = tk.Canvas(self.root, width=300, height=300, bg="orange")
        self.canvas.pack()

        self.logged_in_users = set()
        self.last_login_time = {}
        self.last_logout_time = {}

        self.image_path = "logo.png"
        self.original_image = Image.open(self.image_path)
        self.resized_image = self.original_image.resize((100, 100), Image.LANCZOS)

        self.logo = ImageTk.PhotoImage(self.resized_image)

        self.logo_label = tk.Label(self.canvas, image=self.logo,bg="orange")
        self.logo_label.grid(padx=10,pady=10,row=0, column=0,sticky=tk.W)

        self.login_status_label = tk.Label(self.canvas, text="", font=("Helvetica",
16),bg="orange")
        self.login_status_label.grid(padx=10,pady=10,row=0, column=0,sticky=tk.E)
        message = "Automated Facial-Biometric Attendance"
        self.login_status_label.config(text=message)

        self.clock_label = tk.Label(self.canvas, font=('calibri', 40, 'bold'),
background='orange', foreground='white')
        self.clock_label.grid(row=0, column=1, padx=30, pady=10, sticky=tk.S)
        self.date_label = tk.Label(self.canvas,  background='orange', font=("Helvetica", 12))
```

```
        self.date_label.grid(row=1, column=1, padx=30, pady=10, sticky=tk.N)

        self.video_frame = tk.Label(self.canvas)
        self.video_frame.grid(padx=10,pady=10,row=2, column=0)

        self.message_text = ScrolledText(self.canvas, height=10, width=30,
wrap=tk.WORD)
        self.message_text.grid(padx=10,pady=10,row=2, column=1)
        self.check = 0
        self.run()

    def run(self):
        time_string = time.strftime('%H:%M:%S %p')
        self.clock_label.config(text=time_string)

        current_date = time.strftime("%Y-%m-%d %A")
        self.date_label.config(text=current_date)

        ret, frame = self.vid.read()
        value = datetime.now()

        if ret:
            frame = cv2.flip(frame, 1)
            user, average_similarity = verify_face(frame)
            if user != "Unknown" and average_similarity >= 50.0 :
                self.updatedb(user)
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            image = Image.fromarray(frame_rgb)
            photo = ImageTk.PhotoImage(image=image)
```

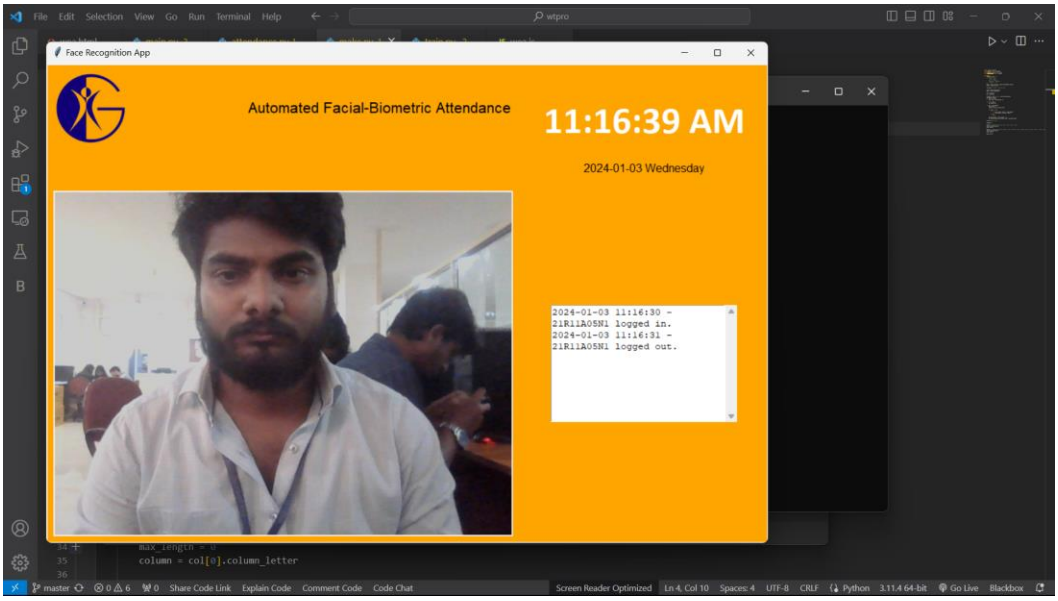
```
            self.video_frame.imgtk = photo
            self.video_frame.configure(image=photo)

        if value.hour == 16 and self.check == 0:
            self.check = 1
            attendance.set()
        self.root.after(10, self.run)

    def updatedb(self,user):
        result = self.find(user,"login")
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        value = datetime.now().time()
        value = datetime.combine(datetime.today(), value)
        if result[0][0] is not None:
            ress = self.find(user,"logout")
            if ress[0][0] is None:
                resultx = datetime.strptime(result[0][0], '%H:%M:%S')
                print()
                time_diff = value - resultx
                print(time_diff)
                if time_diff >= timedelta(minutes=1) and  (user not in self.last_logout_time or
self.last_logout_time[user] is None):
                    self.update(user,"logout")
                    login_message = f"{timestamp} - {user} logged out.\n"
                    self.message(login_message)

                elif time_diff == timedelta(minutes=1):
                    login_message = f"{user} already logged in.\n"
                    self.message(login_message)
            else:
```

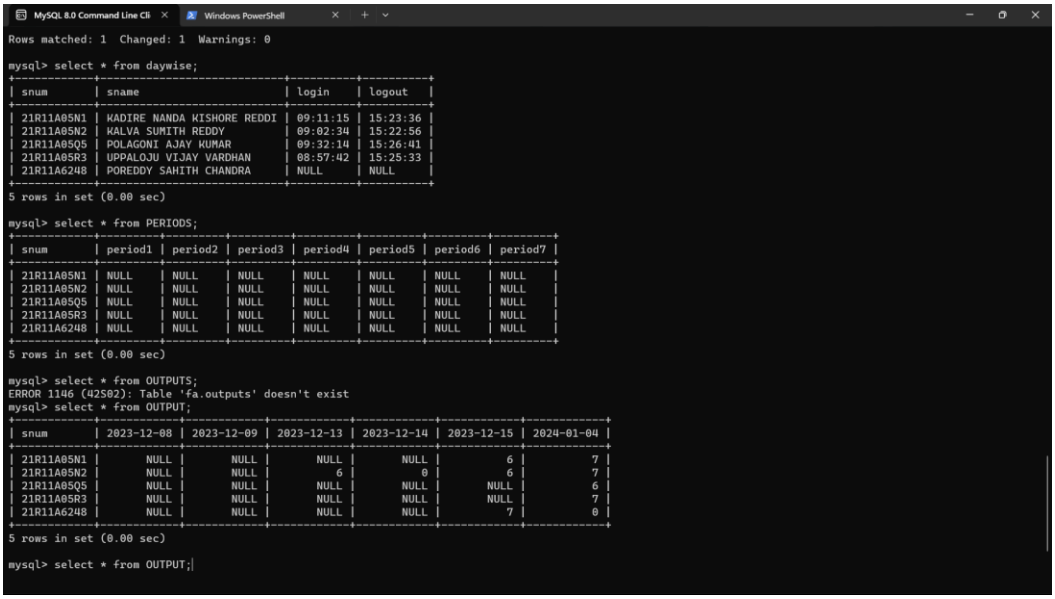
```
            pass
        else:
            self.update(user,"login")
            login_message = f"{timestamp} - {user} logged in.\n"
            self.message(login_message)

    def update(self,user,mode):
        timestamp = time.strftime("%H:%M:%S", time.localtime())
        query = f"UPDATE daywise SET `{mode}` = %s WHERE `snum` = %s"
        cursor.execute(query, (timestamp,user))
        mydb.commit()

    def find(self,user,mode):
        query = f"select {mode} from daywise WHERE `snum` = %s"
        cursor.execute(query, (user,))
        result = cursor.fetchall()
        return result

    def message(self,login_message):
        self.message_text.insert(tk.END, login_message)
        self.message_text.see(tk.END)

    def on_quit(self):
        cursor.close()
        mydb.close()
        self.vid.release()
        self.root.destroy()



def verify_face(frame):
```

```
    with open('encodings1.pkl', 'rb') as file:
        preloaded_encodings = pickle.load(file)

    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, num_jitters=1)

    if not face_encodings:
        return "Unknown", 0.0

    best_user = None
    highest_similarity = 0.0
    for current_face_encoding, current_face_location in zip(face_encodings,
face_locations):
        for user, user_face_encodings in preloaded_encodings.items():
            similarities = face_recognition.face_distance(user_face_encodings,
current_face_encoding)
            average_similarity = (1.0 - sum(similarities) / len(similarities)) * 100
            if average_similarity > highest_similarity:
                highest_similarity = average_similarity
                best_user = user
    return best_user, highest_similarity

def main():
    root = tk.Tk()
    app = FaceRecognitionApp(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

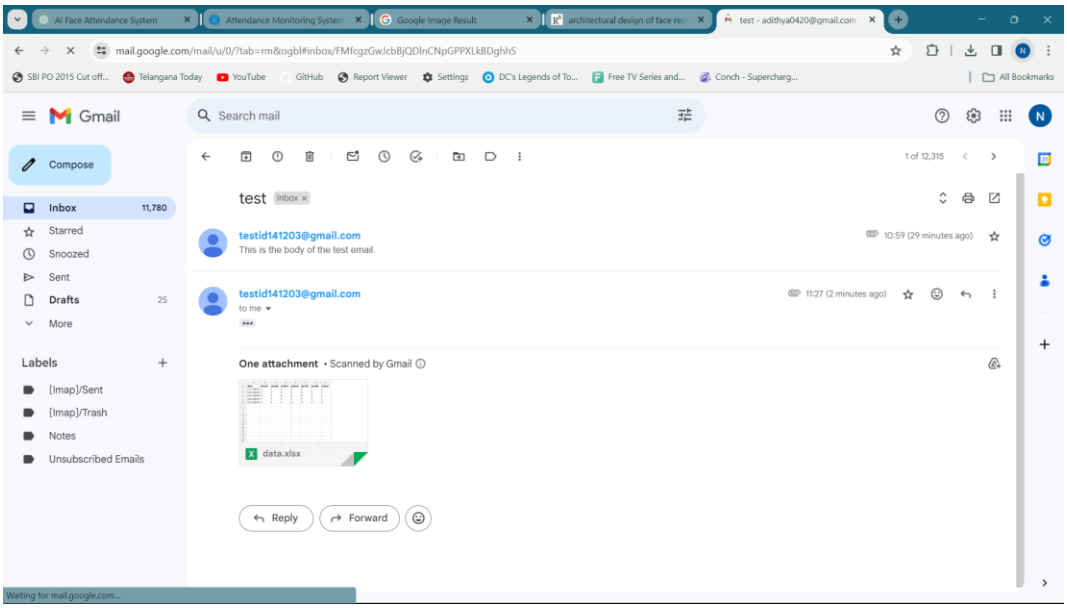
3.3 Sample Output Screenshots/Results

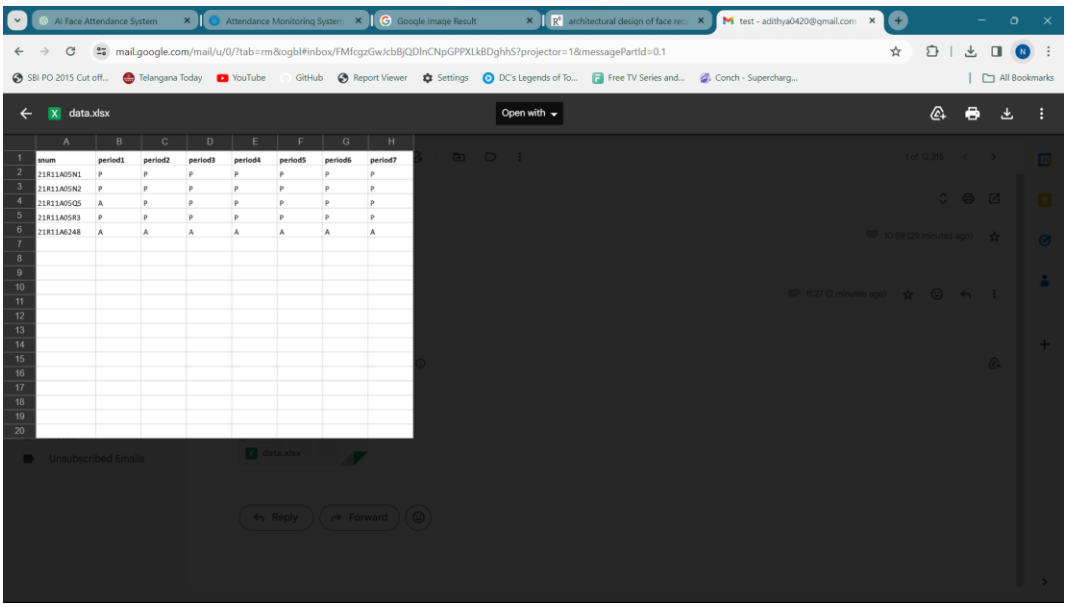
Visual representations of the system in action, including screenshots of the UI, facial recognition results, and real-time attendance updates.





```
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from daywise;
+------------+----------------------------+----------+----------+
| snum       | sname                      | login    | logout   |
+------------+----------------------------+----------+----------+
| 21R11A05N1 | KADIRE NANDA KISHORE REDDI | 09:11:15 | 15:23:36 |
| 21R11A05N2 | KALVA SUMITH REDDY         | 09:02:34 | 15:22:56 |
| 21R11A05Q5 | POLAGONI AJAY KUMAR        | 09:32:14 | 15:26:41 |
| 21R11A05R3 | UPPALOJU VIJAY VARDHAN     | 08:57:42 | 15:25:33 |
| 21R11A6248 | POREDDY SAHITH CHANDRA     | NULL     | NULL     |
+------------+----------------------------+----------+----------+
5 rows in set (0.00 sec)

mysql> select * from PERIODS;
+------------+---------+---------+---------+---------+---------+---------+---------+
| snum       | period1 | period2 | period3 | period4 | period5 | period6 | period7 |
+------------+---------+---------+---------+---------+---------+---------+---------+
| 21R11A05N1 | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    |
| 21R11A05N2 | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    |
| 21R11A05Q5 | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    |
| 21R11A05R3 | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    |
| 21R11A6248 | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    | NULL    |
+------------+---------+---------+---------+---------+---------+---------+---------+
5 rows in set (0.00 sec)

mysql> select * from OUTPUTS;
ERROR 1146 (42S02): Table 'fa.outputs' doesn't exist
mysql> select * from OUTPUT;
+------------+------------+------------+------------+------------+------------+------------+
| snum       | 2023-12-08 | 2023-12-09 | 2023-12-13 | 2023-12-14 | 2023-12-15 | 2024-01-04 |
+------------+------------+------------+------------+------------+------------+------------+
| 21R11A05N1 |       NULL |       NULL |       NULL |       NULL |          6 |          7 |
| 21R11A05N2 |       NULL |       NULL |          6 |          0 |          6 |          7 |
| 21R11A05Q5 |       NULL |       NULL |       NULL |       NULL |       NULL |          6 |
| 21R11A05R3 |       NULL |       NULL |       NULL |       NULL |       NULL |          7 |
| 21R11A6248 |       NULL |       NULL |       NULL |       NULL |          7 |          0 |
+------------+------------+------------+------------+------------+------------+------------+
5 rows in set (0.00 sec)

mysql> select * from OUTPUT;
```


AI Face Attendance System
Attendance Monitoring System
Google Image Result
architectural design of face rec
test - adithya0420@gmail.com
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGwJcbBjQDlnCNpGPPXLkBDghhS
SBI PO 2015 Cut off...
Telangana Today
YouTube
GitHub
Report Viewer
Settings
DC's Legends of To...
Free TV Series and...
Conch - Supercharg...
All Bookmarks
Gmail
Search mail
Compose
Inbox 11,780
Starred
Snoozed
Sent
Drafts 25
More
Labels
[Imap]/Sent
[Imap]/Trash
Notes
Unsubscribed Emails
1 of 12,315
test
Inbox
testid141203@gmail.com
This is the body of the test email.
10:59 (29 minutes ago)
testid141203@gmail.com
to me
11:27 (2 minutes ago)
One attachment • Scanned by Gmail
data.xlsx
Reply
Forward
Waiting for mail.google.com...


AI Face Attendance System
Attendance Monitoring System
Google Image Result
architectural design of face rec
test - adithya0420@gmail.com
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGwJcbBjQDlnCNpGPPXLkBDghhS?projector=1&messagePartId=0.1
SBI PO 2015 Cut off...
Telangana Today
YouTube
GitHub
Report Viewer
Settings
DC's Legends of To...
Free TV Series and...
Conch - Supercharg...
All Bookmarks
data.xlsx
Open with
snum period1 period2 period3 period4 period5 period6 period7
21R11A05N1 P P P P P P P
21R11A05N2 P P P P P P P
21R11A05Q5 A P P P P P P
21R11A05R3 P P P P P P P
21R11A6248 A A A A A A A
Unsubscribed Emails
data.xlsx
Reply
Forward

4. Conclusion

In summary, the AI-based face attendance system emerges as a transformative solution, effectively modernizing attendance tracking processes. The successful integration of facial recognition technology, coupled with a scalable client-server architecture, ensures accuracy, efficiency, and adaptability across various sectors.

The system's contribution extends beyond routine attendance management, introducing a secure and contactless approach that aligns with contemporary needs. Challenges encountered during development provided valuable insights for ongoing improvements, laying the groundwork for future innovations.

As organizations increasingly recognize the benefits of such systems, the project's impact lies in its ability to redefine how we approach security and efficiency in attendance tracking. The journey from concept to implementation highlights the potential of artificial intelligence to enhance fundamental aspects of organizational processes.

In conclusion, the AI-based face attendance system signifies a positive shift in attendance management methodologies, showcasing the power of innovation to meet the evolving needs of diverse sectors. The success of this project serves as inspiration to continue exploring and leveraging cutting-edge technologies for meaningful advancements.

5. References

- OpenCV Documentation
- Dlib Documentation
- Tkinter Documentation
- FaceRecognition GitHub Repository

- ChatGPT
- MySQL Documentation