Chapter 1
INTRODUCTION

In recent years, the rapid advancement of technology has paved the way for innovative methods of automating routine tasks. One such application is the use of face recognition technology for attendance monitoring in educational institutions, workplaces, and various other environments. This mini project, "Attendance Monitoring System Through Face Recognition," aims to develop a robust and efficient attendance monitoring system using machine learning techniques with Python, leveraging face recognition to ensure accurate and seamless tracking of attendance.

The core functionality of the system is built around the ability to detect and recognize faces in real-time using a webcam. The system employs the OpenCV library for video capture and the face recognition library for face detection and recognition. The face recognition model is trained on a set of pre-registered facial images, which are encoded and stored in a database. These encoded images are used to match faces captured during the attendance process, ensuring accurate identification. The implementation involves several key components such as face detection and recognition, real-time processing, attendance logging, and user interface.

By automating the attendance process, the system not only improves accuracy but also reduces the administrative burden associated with traditional methods. The successful implementation of this project will result in a functional attendance monitoring system that can accurately recognize and record the attendance of individuals in real-time, providing a reliable, efficient, and user-friendly solution adaptable to various environments.

1.1 PROBLEM STATEMENT

The primary problem addressed by this project is the inefficiency and inaccuracy of traditional attendance-taking methods in educational institutions. Manual roll calls are time-consuming and susceptible to human error or manipulation. This project aims to develop an automated attendance system using face recognition technology to ensure accurate, efficient, and reliable attendance tracking. The system should be able to recognize registered students, mark their attendance in real-time, and store the data for future reference. Key challenges include ensuring high accuracy of face recognition under varying environmental conditions and addressing privacy concerns related to the storage and usage of facial data.

1.2 MOTIVATION

The motivation behind developing a face recognition-based attendance system stems from the need for a more efficient, reliable, and non-intrusive method to record attendance. Traditional methods, such as manual roll calls or using ID cards, are time-consuming and prone to errors or manipulation. By leveraging advanced face recognition technology, this system aims to streamline the attendance process, reduce administrative workload, and ensure accurate record-keeping.

1.3 ADVANTAGES

- Efficiency: Automates the attendance process, saving time for both students and instructors.
- Accuracy: Reduces the possibility of errors and manipulation compared to manual methods.
- Non-Intrusive: Students do not need to carry any physical tokens (like ID cards) for attendance.
- Real-Time Monitoring: Provides real-time updates on attendance status, which can be useful for large classes.
- Historical Data: Automatically records attendance data, making it easy to analyze and review past attendance.

1.4 DISADVANTAGES

- Privacy Concerns: Storing facial data may raise privacy issues among students.
- Cost: Initial setup costs for the required hardware (cameras, computers) and software.
- Environment Dependency: The system's accuracy can be affected by lighting conditions and camera angles.
- False Positives: The technology may not be 100% accurate, leading to potential misidentifications.
- Technical Issues: Requires ongoing maintenance and troubleshooting to handle technical glitches.

1.5 21CSMP67 - Mini Project

Work breakdown structure

Week No	Date	Details of work done
Week 1	01-05-2024 to 07-05-2024	Install necessary libraries (e.g., face_recognition, opencv-python, numpy, etc.).
		Initialize Video Capture
		Load Images and Get Face Encoding
		Store Face Encodings and Names
		Initialize Variables
Week 2	08-05-2024 to 15-05-2024	Capture Frame-by-Frame
		Process Frames to Reduce CPU Load
		Resize and Convert Frame
		Encode Faces in the Frame
		Initialize Face Names List
Week 3	16-05-2024 to 23-05-2024	Compare Detected Faces with Known Faces
		Determine Best Match for Each Face
		Mark Attendance for Recognized Faces
		Remove Recognized Faces from the List
		Print Attendance Confirmation
Week 4	24-05-2024 to 31-05-2024	Draw Rectangles Around Faces
		Break Loop on 'q' Key Press
		Release Video Capture
		Close All OpenCV Windows
		Testing and Debugging

Chapter 2

LITERATURE SURVEY

2.1 Face Recognition Technology

Face recognition technology has evolved significantly over the past few decades, becoming one of the most prominent applications of image processing and computer vision. Early methods relied on geometric features and template matching techniques. However, advancements in machine learning, particularly deep learning, have revolutionized face recognition systems. Convolutional Neural Networks (CNNs) have proven to be exceptionally effective, with models like VGG-Face, Face Net, and Deep Face setting new benchmarks for accuracy and reliability. These deep learning models can automatically extract complex features from facial images, enabling high-precision recognition even under challenging conditions such as varying lighting, poses, and occlusions.

2.2 Real-time Video Processing

Real-time video processing is crucial for applications like surveillance, interactive gaming, and automated attendance systems. The advent of powerful GPUs and efficient algorithms has made real-time processing feasible. Techniques such as background subtraction, frame differencing, and optical flow are commonly used for motion detection and tracking. OpenCV, an open-source computer vision library, offers a comprehensive suite of tools for video capture, processing, and analysis. Integration with deep learning frameworks like TensorFlow and PyTorch further enhances its capabilities, allowing for the real-time detection and recognition of objects and faces in video streams.

2.3 Attendance Systems

Traditional attendance systems often rely on manual methods such as roll calls, sign-in sheets, or electronic card swipes. These methods are prone to errors, time-consuming, and vulnerable to manipulation. Automated attendance systems using biometric technologies, such as fingerprint and iris recognition, offer improved accuracy and security. However, face recognition-based attendance systems provide a non-intrusive and user-friendly alternative. Research has shown that these systems can significantly reduce administrative workload and enhance the reliability of attendance tracking in educational institutions, workplaces, and events.

2.4 Machine Learning and Image Processing Libraries

The development of machine learning and image processing libraries has played a pivotal role in the advancement of face recognition technology. Libraries like OpenCV provide essential functions for image manipulation, feature extraction, and video processing. The face recognition library, built on dlib’s state-of-the-art face recognition built with deep learning, is widely used for its simplicity and effectiveness. These libraries enable researchers and developers to implement complex face recognition systems with relatively minimal effort, leveraging pre-trained models and robust algorithms

2.5 CSV Logging and Data Handling

Efficient data handling and logging are critical for the performance and usability of attendance monitoring systems. CSV (Comma-Separated Values) files are a popular choice for logging attendance data due to their simplicity and compatibility with various software applications. Python’s csv module provides convenient methods for reading from and writing to CSV files, facilitating the seamless integration of data logging into attendance systems. This ensures that attendance records are accurately maintained and easily accessible for analysis and reporting.

Chapter 3

SYSTEM ANALYSIS

3.1 Hardware requirements

3.1.1 Computer/Laptop:

- Processor: A dual-core processor (Intel Core i3, AMD equivalent).
- RAM: At least 4 GB.
- Storage: At least 10 GB of free disk space.
- GPU: Integrated graphics (Intel HD Graphics) can work, though processing will be slower compared to dedicated GPUs.
- **Webcam:** A standard USB webcam or built-in laptop camera capable of capturing at least 640x480 resolution

3.2 Software requirements

Operating System: Windows 10 or later, macOS, or a Linux

Python: Python 3.7 or higher.

Python Libraries:

- face_recognition: For face detection and recognition.
- OpenCV: For video capture and image processing.
- NumPy: For numerical operations.
- CSV: For reading and writing CSV files.

Text Editor: Any modern code editor such as Visual Studio Code, PyCharm, or Jupyter Notebook.

3.3 Functional Requirements

Real-Time Face Detection and Recognition

- The system must be able to detect faces in real-time from a video feed.
- It should recognize faces from a preloaded set of known faces.

Attendance Marking

- When a face is recognized, the system must log the name of the person and the current time in a CSV file.
- Each student should be marked present only once per session.

Data Storage

- The system must create a new CSV file for each day to store attendance records.
- Each entry in the CSV file should include the name of the recognized person and the time they were marked present.

User Feedback

- The system should display the name of the recognized person on the video feed when they are marked present.
- Faces detected but not recognized should be marked as "Unknown".

Efficiency

- The system should process every 5th frame to reduce CPU load.

Interface and Interaction

- The system should display the video feed in a window.
- The system should allow the user to terminate the program by pressing the 'q' key.

3.4 Non-Functional Requirements

Performance

- The face detection and recognition should be efficient enough to work in real-time.
- The system should process video frames quickly to provide immediate feedback.

Scalability

- The system should be able to handle multiple known faces efficiently.
- It should be easy to add or remove known faces from the system.

Reliability

- The system should reliably detect and recognize faces under various lighting conditions.
- It should handle cases where no faces are detected without errors.

Usability

- The user interface (video feed window) should be simple and easy to understand.
- The system should provide clear feedback (e.g., displaying names on the video feed).

Maintainability

- The code should be well-documented and modular to facilitate easy maintenance and updates.
- The system should log errors and handle exceptions gracefully.

Accuracy

- The face recognition should have a high accuracy rate to minimize false positives and false negatives.
- The system should use an effective algorithm for face recognition to ensure reliability.

Security

- The system should ensure the privacy of the recognized individuals by securely storing the attendance records.
- Access to the system and its records should be restricted to authorized personnel only.

Portability

- The system should be compatible with different operating systems (e.g., Windows, macOS, Linux).
- It should be able to work with different webcam models and configurations.

Chapter 4

SYSTEM DESIGN

The system design for the Attendance Monitoring System Through Face Recognition" project involves several key components working together to achieve the goal of automated attendance recording. The design encompasses hardware and software components, as well as the flow of data through the system.

4.1 System Architecture

The system architecture is divided into the following modules:

- **Video Capture Module:** This module is responsible for capturing real-time video feed from a webcam. The webcam continuously streams video frames to the system for further processing. The quality and resolution of the video feed are crucial as they directly impact the accuracy of face detection and recognition.

- **Face Detection and Recognition Module:** This module detects faces in the captured video frames and matches them against a pre-existing database of known faces. It employs sophisticated algorithms to ensure accurate and efficient face recognition.

- **Attendance Logging Module:** This module records the attendance of recognized individuals in a CSV file. It maintains a log of attendance, including the name and the time of recognition.

- **User Interface Module:** This module provides real-time feedback on the video feed, indicating recognized individuals and their attendance status. It enhances user interaction and provides visual confirmation of attendance loggings.

4.2 Algorithms Used

The following algorithms and techniques are used in the system:

4.2.1 HOG (Histogram of Oriented Gradients) for Face Detection:

- Histogram of Oriented Gradients (HOG) is a widely used feature descriptor in computer vision, particularly for object detection tasks such as human detection. The core idea behind HOG is to capture and represent the structure and appearance of objects by analysing the distribution of gradient orientations within an image.
- The process begins with the computation of image gradients, which are measures of intensity change in the image. Specifically, HOG calculates the gradient magnitude and direction for each pixel, reflecting how the pixel intensity varies in both horizontal and vertical directions. This information is crucial for understanding the edges and boundaries within the image.

4.2.2 Euclidean Distance:

- Face recognition involves comparing facial encodings using Euclidean Distance. The system computes the distance between the encoding of an unknown face and each known face encoding to find the closest match.
- Euclidean Distance measures the straight-line distance between two points in the feature space. The face with the smallest distance to the unknown face's encoding is considered the best match. This approach is effective in identifying faces based on their feature vectors.

Chapter 5

IMPLEMENTATION

It implements a face recognition-based attendance system using a webcam. It captures video frames, detects faces, compares them with known faces, and records the attendance of recognized individuals in a CSV file.

5.1 APIs Used

- face_recognition: This library simplifies the implementation of face detection and recognition using deep learning models.
- cv2 (OpenCV): This is used for video capture, image processing, and displaying the video feed.
- numpy: Used for numerical operations.
- csv: Used for writing attendance records to a CSV file.
- datetime: Used for timestamping attendance records.

5.2 Working of the Code

1. Initialization:

- The video capture is initialized using OpenCV (`cv2.VideoCapture(0)`), which accesses the default webcam.
- Known faces are loaded from image files, and their face encodings are computed using the `face_recognition` library.
- A list of known face encodings and names is created.
- An empty list `students` is initialized with known face names to keep track of students who haven't been marked present.

2. CSV File Setup:

- The current date is obtained using `datetime.now()` and formatted as a string.
- A CSV file is created with the current date as the filename, and a header row ("Name", "Time") is written.

3. Main Loop:

- The main loop continuously captures frames from the webcam.
- Every 5th frame is processed to reduce CPU load. The frame is resized to a quarter of its original size and converted to RGB.
- Faces are detected in the resized frame, and face encodings are computed.
- For each detected face, the code compares it with known face encodings to find matches.
- If a match is found, the name of the recognized person is appended to the `face_names` list.

4. Attendance Marking:

- If a recognized face is found in the list of students, it is removed from the list (to avoid duplicate entries).
- The current time is recorded, and an entry is written to the CSV file with the name and time.
- The recognized person's name and "Present" are displayed on the video frame.

5. Drawing and Display:

- Rectangles are drawn around detected faces.
- The video frame is displayed using OpenCV's `imshow` function.

6. Exit Condition:

- The loop continues until the 'q' key is pressed.
- Upon exiting, the video capture is released, and all OpenCV windows are closed.

Chapter 6

TESTING

6.1 Images of known faces:

You need to have images of the individuals you want to recognize. Place these images in a folder named "photos". For the given script, you should have the following images:

- photos/jobs.jpg
- photos/shiva.jpg
- photos/shreesha.jpg
- photos/shashank.jpg
- photos/venkatesh.jpg

6.2 Install necessary libraries:

Ensure you have face_recognition, opencv-python, numpy, and other required libraries installed. You can install these using pip:

```
pip install face_recognition opencv-python numpy
```

6.3 Run the script:

Save the script provided as a Python file, for example attendance_system.py, and run it in your Python environment:

```
python attendance_system.py
```

CONCLUSION

The Attendance Monitoring through Face Recognition project demonstrates a significant advancement in automating and streamlining the process of attendance tracking. By integrating facial recognition technology, the system effectively addresses the inefficiencies and inaccuracies associated with traditional attendance methods. The use of real-time video capture and advanced face encoding algorithms ensures that the attendance data is both accurate and timely, providing a reliable record of who is present at any given moment.

This project has successfully implemented a user-friendly interface that simplifies the management of attendance monitoring. The ability to automatically identify and record individuals' presence not only saves time but also enhances administrative efficiency, allowing organizations to focus on other critical tasks. The system’s real-time capabilities further enhance its value by ensuring that attendance records are always up-to-date, which is crucial for environments where attendance can vary frequently.

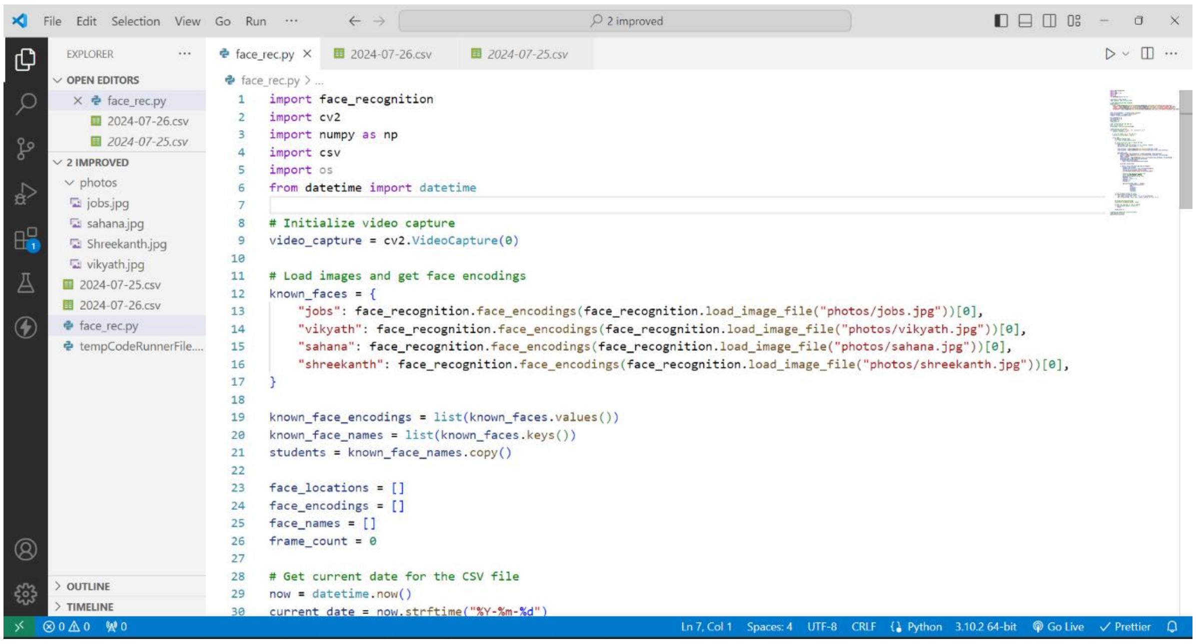
In conclusion, the Attendance Monitoring through Face Recognition project represents a modern solution that leverages cutting-edge technology to improve the accuracy and efficiency of attendance tracking. Its implementation offers a practical example of how facial recognition can be applied beyond security, demonstrating its potential to transform everyday administrative processes. The successful integration of this technology underscores its potential for broader application in various sectors, promising enhanced operational efficiency and accuracy in attendance management.

REFERENCES

[1] Analytics Vidhya – https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python/
[2] FaceOnLive – https://faceonlive.com/face-recognition-attendance-system-project-step-by-step-guide/
[3] Open AI – ChatGPT

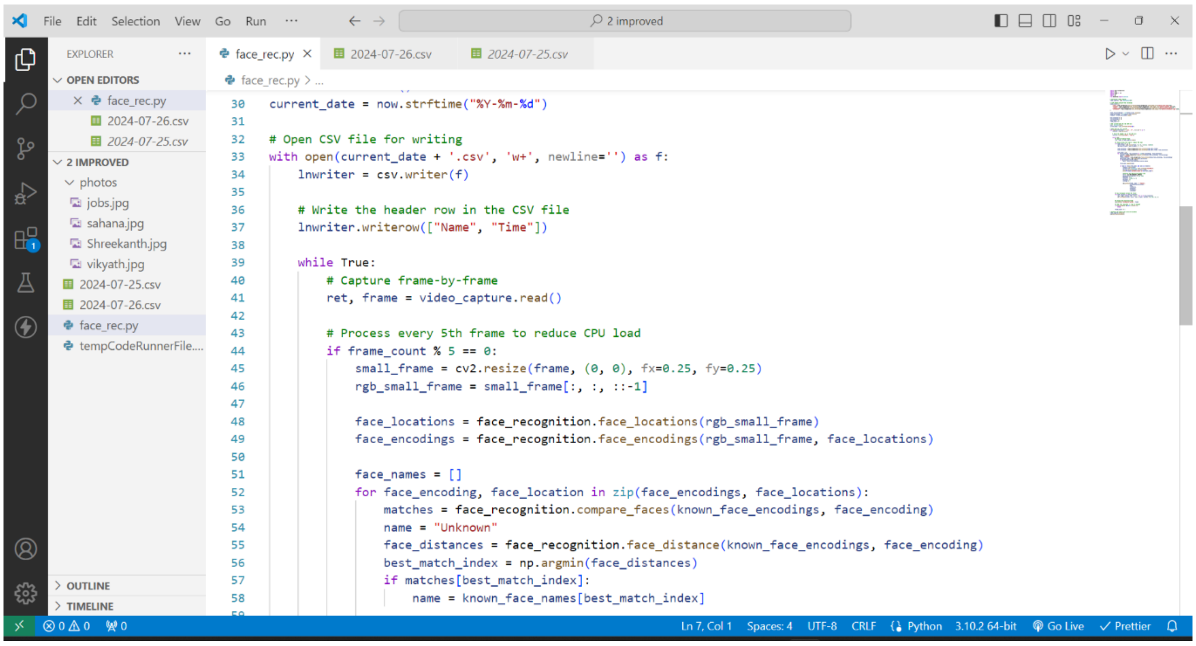
APPENDIX

Code:



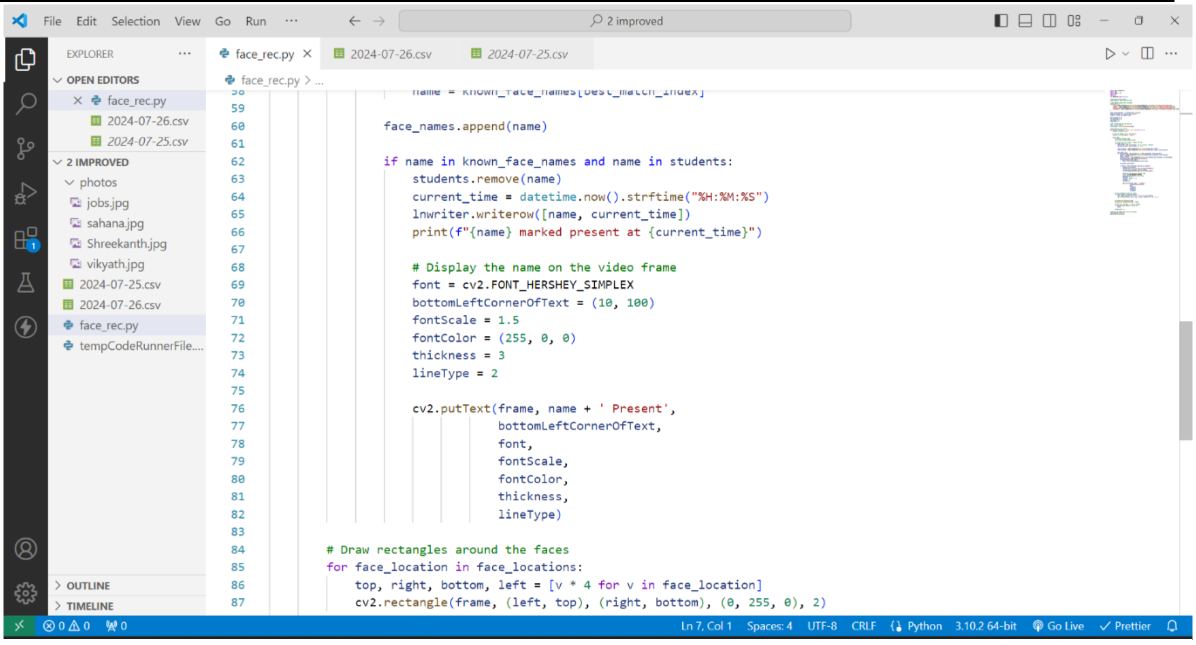
```python
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime

# Initialize video capture
video_capture = cv2.VideoCapture(0)

# Load images and get face encodings
known_faces = {
    "jobs": face_recognition.face_encodings(face_recognition.load_image_file("photos/jobs.jpg"))[0],
    "vikyath": face_recognition.face_encodings(face_recognition.load_image_file("photos/vikyath.jpg"))[0],
    "sahana": face_recognition.face_encodings(face_recognition.load_image_file("photos/sahana.jpg"))[0],
    "shreekanth": face_recognition.face_encodings(face_recognition.load_image_file("photos/shreekanth.jpg"))[0],
}

known_face_encodings = list(known_faces.values())
known_face_names = list(known_faces.keys())
students = known_face_names.copy()

face_locations = []
face_encodings = []
face_names = []
frame_count = 0

# Get current date for the CSV file
now = datetime.now()
current_date = now.strftime("%Y-%m-%d")
```

Figure 1: code for importing library and training datasets



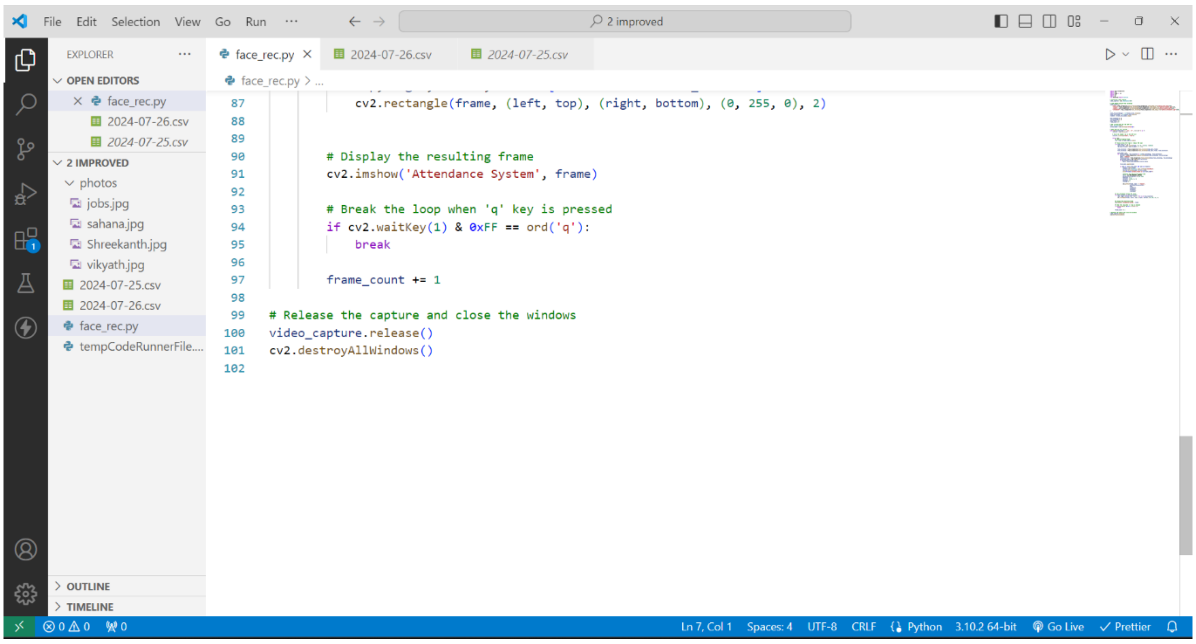
```python
current_date = now.strftime("%Y-%m-%d")

# Open CSV file for writing
with open(current_date + '.csv', 'w+', newline='') as f:
    lnwriter = csv.writer(f)

    # Write the header row in the CSV file
    lnwriter.writerow(["Name", "Time"])

    while True:
        # Capture frame-by-frame
        ret, frame = video_capture.read()

        # Process every 5th frame to reduce CPU load
        if frame_count % 5 == 0:
            small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
            rgb_small_frame = small_frame[:, :, ::-1]

            face_locations = face_recognition.face_locations(rgb_small_frame)
            face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

            face_names = []
            for face_encoding, face_location in zip(face_encodings, face_locations):
                matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
                name = "Unknown"
                face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    name = known_face_names[best_match_index]
```

Figure 2: code for storing data in csv file with date time



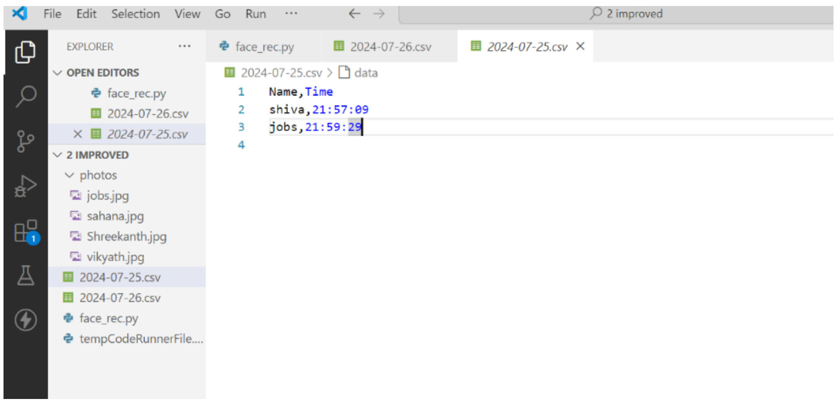
```
                    name = known_face_names[best_match_index]

                face_names.append(name)

                if name in known_face_names and name in students:
                    students.remove(name)
                    current_time = datetime.now().strftime("%H:%M:%S")
                    lnwriter.writerow([name, current_time])
                    print(f"{name} marked present at {current_time}")

                    # Display the name on the video frame
                    font = cv2.FONT_HERSHEY_SIMPLEX
                    bottomLeftCornerOfText = (10, 100)
                    fontScale = 1.5
                    fontColor = (255, 0, 0)
                    thickness = 3
                    lineType = 2

                    cv2.putText(frame, name + ' Present',
                                bottomLeftCornerOfText,
                                font,
                                fontScale,
                                fontColor,
                                thickness,
                                lineType)

        # Draw rectangles around the faces
        for face_location in face_locations:
            top, right, bottom, left = [v * 4 for v in face_location]
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
```

Figure 3: code for comparing the known face to datasets



```
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)


        # Display the resulting frame
        cv2.imshow('Attendance System', frame)

        # Break the loop when 'q' key is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        frame_count += 1

# Release the capture and close the windows
video_capture.release()
cv2.destroyAllWindows()
```

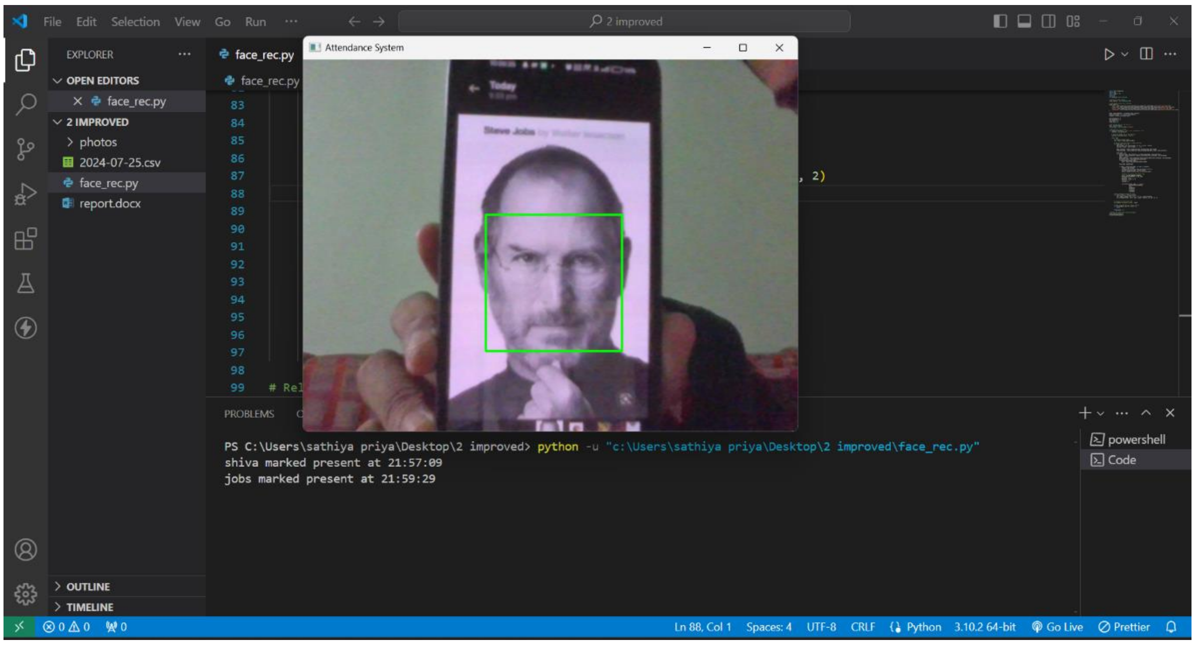
Figure 4: code for displaying the result



```
Name,Time
shiva,21:57:09
jobs,21:59:29
```

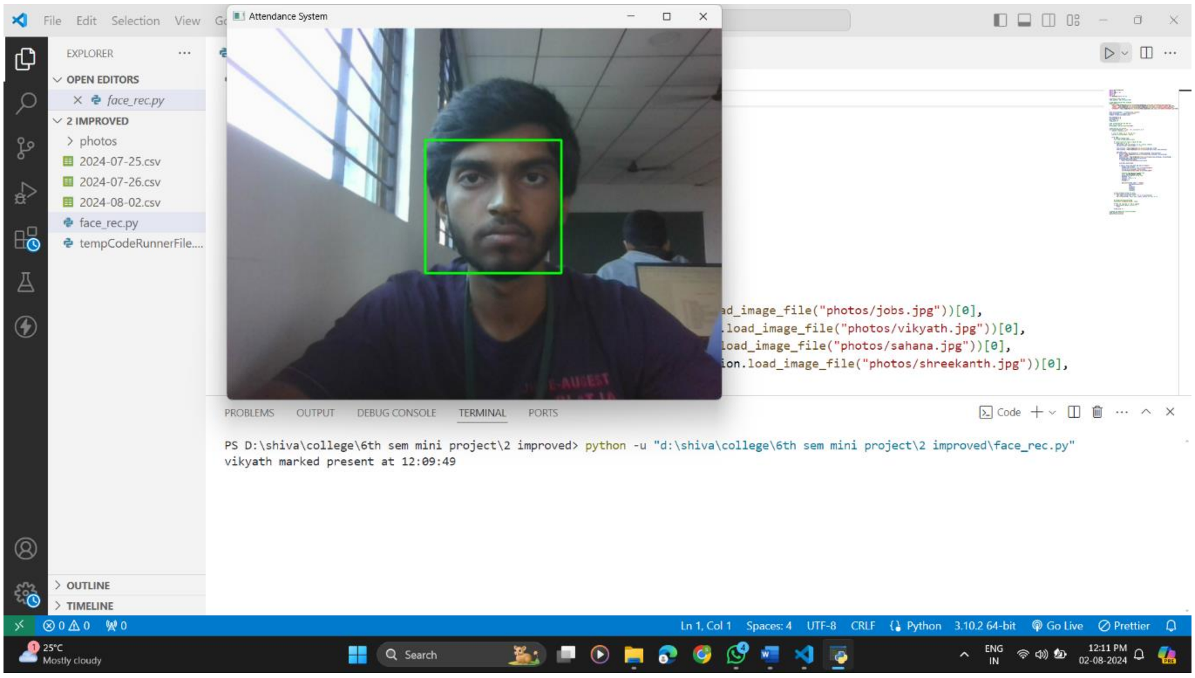
Figure 5: CSV file result

Result:



```
PS C:\Users\sathiya priya\Desktop\2 improved> python -u "c:\Users\sathiya priya\Desktop\2 improved\face_rec.py"
shiva marked present at 21:57:09
jobs marked present at 21:59:29
```

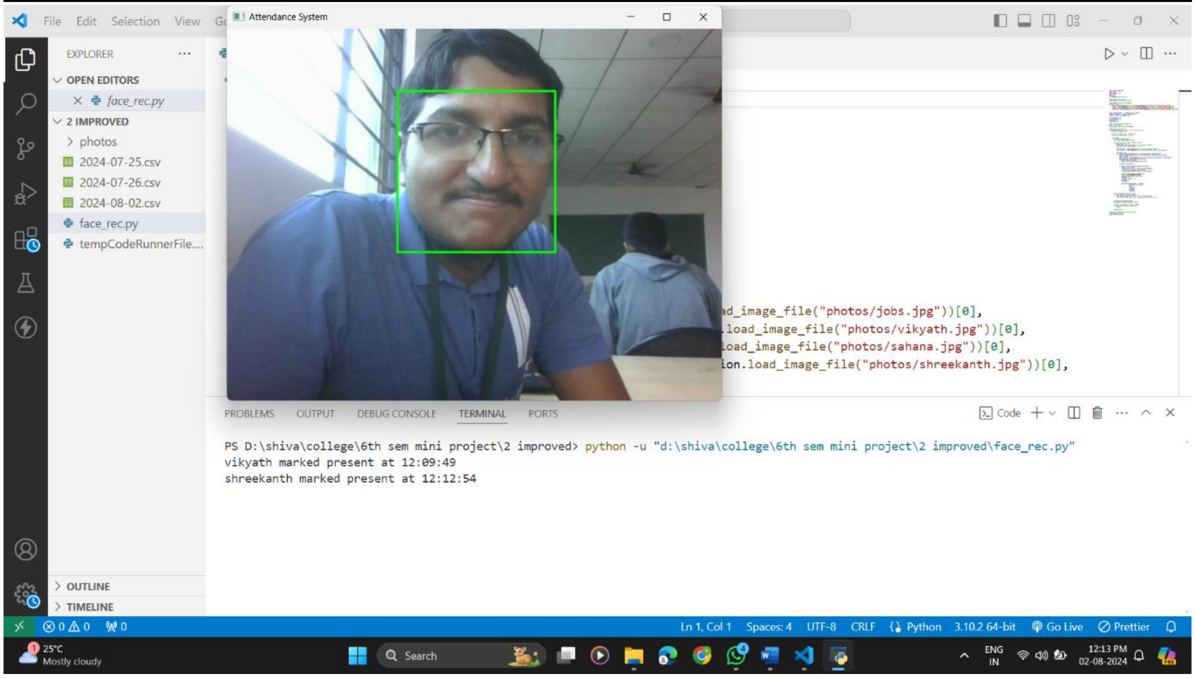
Figure 6: Capturing a face in front of video capture



```
ad_image_file("photos/jobs.jpg"))[0],
load_image_file("photos/vikyath.jpg"))[0],
load_image_file("photos/sahana.jpg"))[0],
ion.load_image_file("photos/shreekanth.jpg"))[0],
```

```
PS D:\shiva\college\6th sem mini project\2 improved> python -u "d:\shiva\college\6th sem mini project\2 improved\face_rec.py"
vikyath marked present at 12:09:49
```

Figure 7: example 1



```
ad_image_file("photos/jobs.jpg"))[0],
.load_image_file("photos/vikyath.jpg"))[0],
load_image_file("photos/sahana.jpg"))[0],
ion.load_image_file("photos/shreekanth.jpg"))[0],
```

```
PS D:\shiva\college\6th sem mini project\2 improved> python -u "d:\shiva\college\6th sem mini project\2 improved\face_rec.py"
vikyath marked present at 12:09:49
shreekanth marked present at 12:12:54
```

Figure 7: example 2