**Automated Attendance System using Facial Recognition**

**CODE:**

import face\_recognition

import cv2

import numpy as np

import csv

from datetime import datetime

import pyttsx3

# Load known faces

srk\_image = face\_recognition.load\_image\_file("faces/srk.jpg")

srk\_encoding = face\_recognition.face\_encodings(srk\_image)[0]

salman\_image = face\_recognition.load\_image\_file("faces/salman.jpg")

salman\_encoding = face\_recognition.face\_encodings(salman\_image)[0]

known\_face\_encodings = [srk\_encoding, salman\_encoding]

known\_face\_names = ["SRK", "Salman"]

# List of expected students

students = known\_face\_names.copy()

# Initialize video capture from webcam

video\_capture = cv2.VideoCapture(0)

# Get the current date

current\_date = datetime.now().strftime("%Y-%m-%d")

# Open CSV file for writing attendance

csv\_filename = f"{current\_date}.csv"

with open(csv\_filename, "w", newline="") as csvfile:

    csv\_writer = csv.writer(csvfile)

    # Initialize font

    font = cv2.FONT\_HERSHEY\_SIMPLEX

    # Initialize the speech engine

    engine = pyttsx3.init()

    while True:

        \_, frame = video\_capture.read()

        small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        rgb\_small\_frame = cv2.cvtColor(small\_frame, cv2.COLOR\_BGR2RGB)

        # Recognize faces

        face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

        face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

        for face\_encoding in face\_encodings:

            if len(face\_encoding) > 0:  # Ensure there's at least one face encoding

                matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

                face\_distance = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

                best\_match\_index = np.argmin(face\_distance)

                if matches[best\_match\_index]:

                    name = known\_face\_names[best\_match\_index]

                    # Add the text if a person is present

                    if name in known\_face\_names:

                        bottomLeftCornerOfText = (10, 50)

                        fontScale = 1

                        FontColor = (255, 255, 255)

                        thickness = 2

                        lineType = 2

                        cv2.putText(frame, f"{name} present", bottomLeftCornerOfText, font, fontScale, FontColor, thickness, lineType)

                        if name in students:

                            students.remove(name)

                            current\_time = datetime.now().strftime("%H:%M:%S")

                            csv\_writer.writerow([name, current\_time])

                            engine.say("Thank you")

                            engine.runAndWait()

                else:

                    engine.say("Please try again")

                    engine.runAndWait()

        cv2.imshow("Attendance", frame)

        if cv2.waitKey(1) & 0xFF == ord("q"):

            break

# Release video capture and close windows

video\_capture.release()

cv2.destroyAllWindows()

This code is a Python script that uses the `face\_recognition` library and OpenCV to perform facial recognition for attendance purposes. It captures video from a webcam, recognizes faces, and records attendance in a CSV file based on recognized individuals.

Here's a breakdown of the code step by step:

1. **Importing Libraries**:

* `face\_recognition`: A library for face recognition tasks.
* `cv2` (OpenCV): A computer vision library for image and video processing.
* `numpy`: A library for numerical operations in Python.
* `csv`: A module for working with CSV (Comma-Separated Values) files.
* `datetime`: A module for working with dates and times.
* `pyttsx3`: A text-to-speech conversion library.

2. **Loading Known Faces**:

* Load images of known individuals (e.g., SRK and Salman Khan) and generate face encodings using `face\_recognition` library.
* Store these face encodings and corresponding names in lists.

3. **Initializing**:

* Create a copy of `known\_face\_names` list as `students`.
* Initialize video capture from the webcam using OpenCV.
* Get the current date using `datetime` and format it as "YYYY-MM-DD".

4. **Opening CSV File**:

* Create a CSV file with the current date as the filename for recording attendance.
* Initialize a CSV writer to write attendance data into the file.

5. **Main Loop**:

* Enter a loop that continuously captures frames from the webcam.
* Resize the captured frame to a smaller size for faster processing.
* Convert the frame from BGR to RGB color space.

6. **Face Recognition and Attendance**:

* Detect face locations in the resized frame.
* Generate face encodings for the detected faces.
* Compare the detected face encodings with the known face encodings.
* If a match is found for a known face:
* Update the frame with the person's name and "present" text.
* Remove the person's name from the `students` list.
* Record the person's name and current time in the CSV file.
* Use `pyttsx3` to generate speech saying "Thank you."
* If no match is found:
* Use `pyttsx3` to generate speech saying "Please try again."

7. **Displaying the Frame**:

* Show the processed frame with attendance information using OpenCV's `imshow` function.
* If the user presses the 'q' key, exit the loop.

8. **Releasing Resources**:

* Release the video capture and close any OpenCV windows.

This script captures video from the webcam, recognizes faces of known individuals, updates the attendance CSV file, and provides audio feedback using text-to-speech. It's important to note that this code is for educational purposes and may require modifications and optimizations for practical use in real-world scenarios.

CODE DEPLOYMENT

Integrating YOLOv4 into your existing attendance tracker involves modifying the code to perform face detection using YOLOv4 before face recognition. Here's a step-by-step guide on how to integrate YOLOv4 into your code:

1. Preparation:

* Ensure you have a trained YOLOv4 model for face detection.
* Make sure you have the necessary YOLOv4 configuration files and weights.
* Install any required YOLOv4 libraries or frameworks.

2. Modify Your Code:

a. Import the necessary YOLOv4-related libraries and initialize the YOLOv4 model.

```python

# Import YOLOv4-related libraries and initialize the YOLOv4 model

from yolov4.detection import YOLOv4

yolo\_model = YOLOv4(weights="path\_to\_weights", cfg="path\_to\_config", names="path\_to\_names")

```

b. In your video capture loop, perform YOLOv4 object detection to detect faces in each frame.

```python

while True:

\_, frame = video\_capture.read()

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

# Perform YOLOv4 object detection to detect faces

detected\_faces = yolo\_model.detect(frame)

for detection in detected\_faces:

x, y, w, h, class\_id, confidence = detection

# Extract the detected face region from the frame

face\_image = frame[y:y+h, x:x+w]

# Perform face recognition on the detected face\_image

# Compare with known face embeddings and log attendance if a match is found

# You can use your existing face recognition code here

# ... (Rest of the code)

```

3. Adjustment:

* Adjust the YOLOv4 configuration, weights, and class names paths according to your specific setup.

4. Testing and Optimization:

* Test the integrated system thoroughly to ensure that YOLOv4 detects faces accurately and that face recognition works as expected.
* Optimize the system for performance, as running both YOLOv4 and face recognition can be resource-intensive.

5. Attendance Logging:

* Continue with your existing attendance logging and notification logic based on the results of face recognition.

6. Voice Output:

* If you wish to maintain voice output, you can continue using the `pyttsx3` library for providing voice notifications as previously described in the code.

Integrating YOLOv4 into your attendance tracker involves combining face detection and face recognition within your existing video capture loop. It's important to ensure that the YOLOv4 face detection results are passed to your face recognition logic for attendance tracking when a face is detected.

Make sure to configure YOLOv4 correctly, adjust the paths, and optimize the system for performance, as real-time object detection and face recognition can be computationally demanding.

CHANGING THE ATTENDACE SYSTEM

Creating an automated attendance system for your college is a significant project that involves multiple components, including hardware, software, and a database system. Here's a step-by-step guide on how to approach this project:

\*\*1. Identify Requirements and Objectives:\*\*

- Clearly define the requirements of the system, including the number of lectures per day, the number of students, and any specific features or functionalities you want.

\*\*2. Hardware Setup:\*\*

- Set up hardware for attendance tracking. You can use various methods, such as biometric devices (fingerprint or facial recognition), RFID cards, or QR code scanners. The choice of hardware depends on your college's preferences and budget.

\*\*3. Software Development:\*\*

- Develop software to interface with the hardware and manage attendance data.

- Create a backend server that handles attendance tracking and data storage.

- Develop a mobile app for students to check their attendance records.

\*\*4. Database Design:\*\*

- Design a database to store attendance records, student information, lecture schedules, and other relevant data.

- Consider using a relational database management system (RDBMS) like MySQL, PostgreSQL, or a NoSQL database depending on your needs.

\*\*5. Attendance Tracking:\*\*

- Implement the attendance tracking logic based on the chosen hardware method (biometric, RFID, QR code, etc.).

- Ensure that the attendance data is accurately recorded for each lecture.

\*\*6. Authentication and Authorization:\*\*

- Implement user authentication and authorization mechanisms to secure access to attendance records.

- Define user roles (e.g., admin, teacher, student) with appropriate permissions.

\*\*7. Data Storage and CSV Generation:\*\*

- Store attendance data in the database for each lecture, associating students with the lectures they attended.

- Automatically generate CSV files or reports for each lecture containing the attendance data.

\*\*8. Mobile App Development:\*\*

- Create a mobile app for students that allows them to:

- Log in to view their attendance records.

- Check their attendance status for each lecture.

- Receive notifications or alerts for upcoming lectures or important announcements.

\*\*9. Web Interface for Teachers/Admins:\*\*

- Develop a web-based interface for teachers and administrators to manage attendance, generate reports, and monitor student attendance trends.

\*\*10. Integration with Existing Systems:\*\*

- Integrate the new attendance system with other existing college systems, such as student information systems or academic databases.

\*\*11. Testing and Quality Assurance:\*\*

- Thoroughly test the entire system to ensure its reliability and accuracy.

- Conduct user testing with both students and faculty to gather feedback and make improvements.

\*\*12. Deployment:\*\*

- Deploy the system in a production environment and ensure that it can handle the expected load.

\*\*13. Training and Documentation:\*\*

- Provide training for teachers, administrators, and students on how to use the system.

- Create user manuals and documentation for reference.

\*\*14. Maintenance and Support:\*\*

- Set up a maintenance plan for regular updates, bug fixes, and technical support.

\*\*15. Security and Privacy:\*\*

- Implement robust security measures to protect student data and attendance records.

- Comply with data privacy regulations and obtain necessary consents.

\*\*16. Continuous Improvement:\*\*

- Continuously gather feedback from users and stakeholders to improve the system and add new features as needed.

Developing an automated attendance system for a college is a complex project that requires careful planning, development, and testing. It's essential to involve relevant stakeholders, including teachers and students, throughout the process to ensure the system meets their needs and expectations. Additionally, consider working with a team of skilled developers and database administrators to bring this project to fruition successfully.