📄 **Project Report: Face Recognition-Based Attendance System.**

📌 **Project Title:**

Face Recognition-Based Attendance System using Python.

📌 **Project Objective:**

The objective of this project is to develop an automatic attendance system using **Face Recognition** that can:

- Detect faces in real-time using a webcam.
- Recognize registered faces.
- Mark attendance automatically.
- Export attendance records in **CSV and PDF format**.

📌 **Project Tools & Technologies:**

- **Programming Language:** Python – version 3.8 & above.
- **Libraries Used:**
    - face_recognition (for face detection and recognition).
    - opencv-python (for webcam handling and image processing).
    - numpy (for array and matrix operations).
    - pandas (for handling CSV files).
    - fpdf (for generating PDF reports).
- **Hardware:** Basic Laptop/PC with Webcam.

📌 **Working Process:**

🔹 **Step 1: Image Dataset Preparation :**

- A folder named **dataset** is created.
- Each student's image is saved inside this folder.
- The image filename must be the **student's name** (Example: Alam.jpg,Rohit.jpg).

🔹 **Step 2: Face Encoding (train.py)**

- The train.py script reads all images from the dataset folder.

- It generates **unique face encodings** for each image using the face_recognition library.
- All encodings and names are saved into a file called **encodings.pkl** for future recognition.

🔹 **Step 3: Real-Time Face Recognition & Attendance Marking (main.py)**

- The main.py script opens the webcam and detects faces in real-time.
- It compares detected faces with the saved encodings.
- If a match is found, the system:
    - Displays the student's name on the webcam feed.
    - Marks the attendance **(Name, Date, Time)** in a CSV file.
- Duplicate entries are automatically ignored for the same session.

🔹 **Step 4: Automatic PDF Report Generation**

- After the attendance session is over, the system automatically generates a **PDF report** using the attendance CSV.
- The PDF report contains:
    - Student Name
    - Attendance Date
    - Attendance Time
- The PDF is saved as **Attendance_Report.pdf** in the project folder.

📌 **Folder Structure:**

Face_Attendance/

```
|
├── dataset/            # Folder containing face images
|    ├── Alam.jpg
|    ├── Rohit.jpg
|    └── ...
├── train.py            # Training file to encode faces
├── main.py             # Main attendance system
├── encodings.pkl       # Saved face encodings
├── attendance.csv      # Attendance record
└── Attendance_Report.pdf  # PDF Report
```

📌 **Advantages:**

- **Contactless:** Attendance is marked without any physical interaction.
- **Time-Saving:** Fast attendance marking compared to manual systems.
- **Automatic Reporting:** Attendance is saved in both CSV and PDF formats.
- **Accurate:** Reduces proxy attendance.

📌 **Limitations:**

- Requires **good lighting** for best face detection.
- If a student changes appearance significantly, retraining is needed.
- System depends on webcam quality.

📌 **Future Scope:**

- Can be integrated with cloud storage for centralized attendance tracking.
- Can be enhanced with GUI for better usability.
- Can add email alerts or SMS notification after attendance.

💻 **Full Code Explanation:**

```
// train.py
import face_recognition
import os
import pickle

path = 'dataset'  # Folder jahan images hain
images = []
classNames = []

print("Starting face encoding...")

for file in os.listdir(path):
    if file.lower().endswith(('.jpg', '.jpeg', '.png')):
        file_path = os.path.join(path, file)
        try:
            img = face_recognition.load_image_file(file_path)
            images.append(img)
            classNames.append(os.path.splitext(file)[0])  # Image ka naam hi student name
            print(f"Loaded {file}")
        except Exception as e:
            print(f"Error loading {file}: {e}")

if not images:
    print("No valid images found. Please check your dataset folder.")
else:
    print('Encoding faces...')
    encodings = [face_recognition.face_encodings(img)[0] for img in images]  # Unique face
signwithopen('encodings.pkl', 'wb') as f:
        pickle.dump((encodings, classNames), f)
    print('Face encodings saved successfully!')
```

✅ Explanation: (train.py)

`face_recognition`: Python library jo face detection aur face encoding karti hai.

-`os`: Files and folders handle karne ke liye.

- `pickle`: Python ka module jo objects ko file me store aur load karne me use hota hai. Yahaan hum face encodings ko save karne ke liye pickle use kar rahe hain.

Path : me hum apne images folder ka path de rahe hain.

os.listdir(path): Folder ke andar jitne bhi files hain, unko list me laata hai.

if file.lower().endswith(...): Sirf .jpg, .jpeg, ya .png files ko hi process karega.

os.path.join(path, file): Image ka full path bana raha hai.

face_recognition.load_image_file(file_path): Image ko load kar raha hai.

images.append(img): Loaded image ko list me daal raha hai.

os.path.splitext(file)[0]: File name ka extension hata kar sirf student ka naam list me store kar raha hai.

try-except: Agar image load me error aaye to process rukta nahi, bas error print karta hai.

images: Ek list banayi gayi hai jisme hum images load karenge.

classNames: Ek list jisme har image ka naam (student ka naam) store hoga.

os.listdir(path): Folder ke andar jitne bhi files hain, unko list me laata hai.

if file.lower().endswith(...): Sirf .jpg, .jpeg, ya .png files ko hi process karega.

os.path.join(path, file): Image ka full path bana raha hai.

face_recognition.load_image_file(file_path): Image ko load kar raha hai.

images.append(img): Loaded image ko list me daal raha hai.

os.path.splitext(file)[0]: File name ka extension hata kar sirf student ka naam list me store kar raha hai.

try-except: Agar image load me error aaye to process rukta nahi, bas error print karta hai.

Console me message print hota hai ki **encoding start** ho gayi hai.

face_recognition.face_encodings(img)[0]: Har image ka unique 128-dimension face encoding nikalta hai.

This is basically a unique vector that represents the face mathematically.

List comprehension: Sabhi images ke liye encoding ek hi step me generate kar raha hai.

`with open('encodings.pkl', 'wb') as f:`: Ek file open ki gayi hai jisme face encodings save karenge.

- `pickle.dump((encodings, classNames), f)`: Dono cheeze (encodings aur names) file me save kar di jaati hain.

- Console me message print hota hai ki training complete ho gayi hai.

✅ Explanation:(main.py)

```python
// main.py – Real-Time Attendance + PDF Export
import cv2
import numpy as np
import face_recognition
import pickle
from datetime import datetime
import pandas as pd
from fpdf import FPDF

# Encodings load karo
with open('encodings.pkl', 'rb') as f:
    known_encodings, known_names = pickle.load(f)

attendance = {}

cap = cv2.VideoCapture(0)
print("Starting attendance. Press 'q' to quit.")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)

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

    for face_encoding, face_location in zip(face_encodings, face_locations):
        matches = face_recognition.compare_faces(known_encodings, face_encoding)
        face_distances = face_recognition.face_distance(known_encodings, face_encoding)

        best_match_index = np.argmin(face_distances) if len(face_distances) > 0 else None

        name = "Unknown"
        if best_match_index is not None and matches[best_match_index]:
            name = known_names[best_match_index]

            if name not in attendance:
                now = datetime.now()
                date_str = now.strftime('%Y-%m-%d')
                time_str = now.strftime('%H:%M:%S')
                attendance[name] = (date_str, time_str)
                print(f"Marked attendance for {name} at {date_str} {time_str}")

        top, right, bottom, left = [v * 4 for v in face_location]
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, name, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

    cv2.imshow('Face Attendance', frame)

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

cap.release()
cv2.destroyAllWindows()

# Attendance CSV me save karo
csv_file = 'attendance.csv'
with open(csv_file, 'w') as f:
    f.write('Name,Date,Time\n')
    for name, (date, time) in attendance.items():
        f.write(f'{name},{date},{time}\n')

print(f"Attendance saved to {csv_file}")

# PDF Export
data = pd.read_csv(csv_file)
current_date = datetime.now().strftime("%Y-%m-%d")

pdf = FPDF()
pdf.add_page()
pdf.set_font('Arial', 'B', 16)
pdf.cell(0, 10, 'Attendance Report', ln=True, align='C')

pdf.set_font('Arial', '', 12)
pdf.cell(0, 10, f'Report Generated on: {datetime.now().strftime("%Y-%m-%d %H:%M:%S")}', ln=True,
align='C')
pdf.ln(10)

pdf.set_font('Arial', 'B', 12)
pdf.cell(80, 10, 'Name', 1)
pdf.cell(60, 10, 'Date', 1)
pdf.cell(50, 10, 'Time', 1)
pdf.ln()

pdf.set_font('Arial', '', 12)
for index, row in data.iterrows():
    pdf.cell(80, 10, str(row['Name']), 1)
    pdf.cell(60, 10, str(row['Date']), 1)
    pdf.cell(50, 10, str(row['Time']), 1)
    pdf.ln()

pdf.output('Attendance_Report.pdf')
print("PDF attendance report generated successfully!")
```

1. Library Imports:

📌 **Explanation:**

- cv2 → OpenCV is used to handle webcam video.
- numpy → Used for distance calculation in face recognition.
- face_recognition → For detecting and matching faces.
- pickle → To load the saved face encodings.
- datetime → To record current date and time for attendance.
- pandas → To manage and export attendance to CSV.
- fpdf → To create PDF attendance reports.

---

2. Load Encodings:

📌 **Explanation:**

- Loads the previously trained face encodings from the pickle file created by train.py.
- known_encodings → List of saved face vectors.
- known_names → Corresponding student names.

---

3. Initialize Webcam and Attendance Dictionary:

📌 **Explanation:**

- attendance → Dictionary to store unique attendance entries.
- cv2.VideoCapture(0) → Opens the system's default webcam.
- User instruction is printed.

---

4. Start Real-Time Face Detection Loop:

📌 **Explanation:**

- Webcam frame is captured continuously in a loop.
- If frame is not captured properly, the loop will break.

---

5. Resize for Faster Processing:

📌 **Explanation:**

- Image is resized to 1/4th size for **speed**.
- Face recognition library works in RGB format, so frame is converted.

---

6. Detect Faces in Frame:

📌 **Explanation:**

- face_locations → Finds the location of all faces in the frame.
- face_encodings → Extracts unique face vectors for detected faces.

---

7. Compare Detected Faces with Known Faces:

📌 **Explanation:**

- Compares detected faces with known faces.
- compare_faces → Checks which face matches.
- face_distance → Finds how close the detected face is to known faces.
- np.argmin(face_distances) → Finds the index of the **best matching face**.

---

8. Mark Attendance if Face is Recognized:

📌 **Explanation:**

- If face matches → Retrieve the name.
- Prevents **duplicate entries** using dictionary keys.
- If name is new → Current date and time are stored.

---

9. Display Face Name and Rectangle:

📌 **Explanation:**

- Draws a rectangle around the face.
- Shows the recognized name above the face.

---

10. Show Webcam Feed:

📌 **Explanation:**

- Real-time webcam window is displayed.
- Pressing q will stop the attendance session.

---

11. Save Attendance to CSV:

📌 **Explanation:**

- Attendance dictionary is written to attendance.csv.
- Each entry → Name, Date, Time.

---

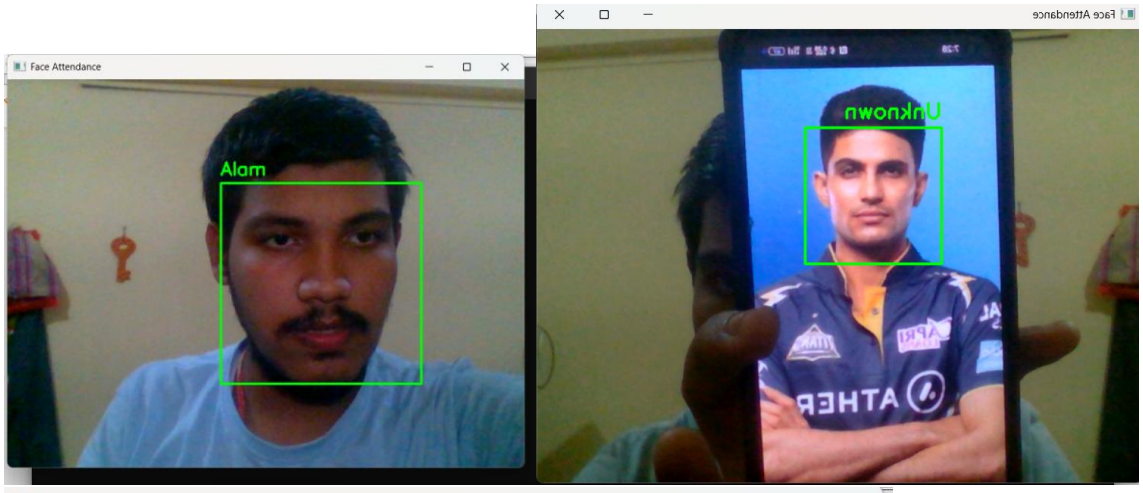
12. Generate PDF Report:

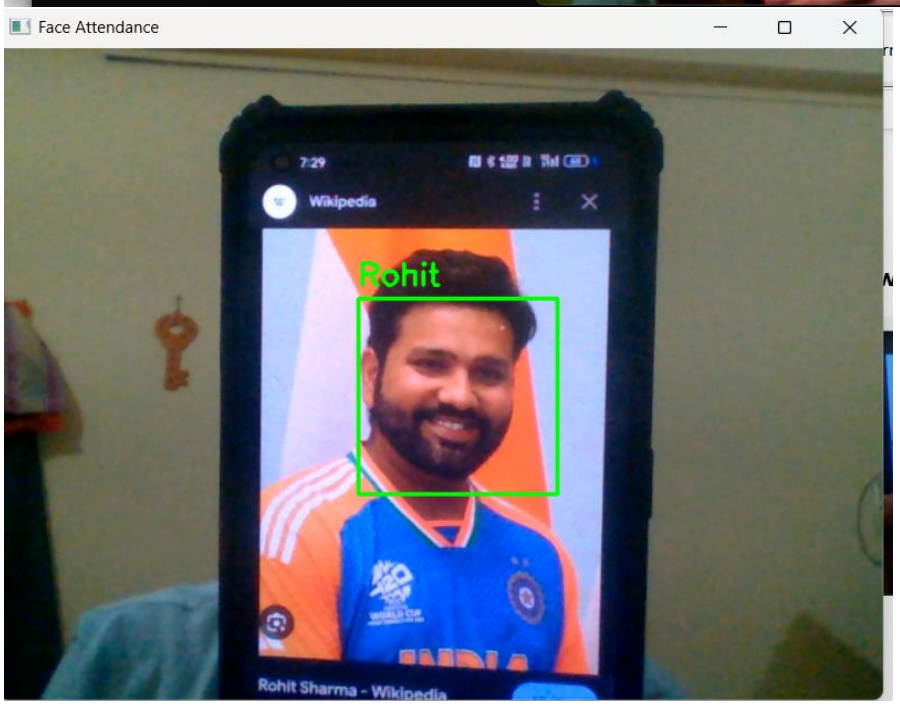
📌 **Explanation:**

- Reads the CSV file using pandas.
- Uses fpdf library to create a neat PDF table.
- Final PDF is saved as Attendance_Report.pdf.

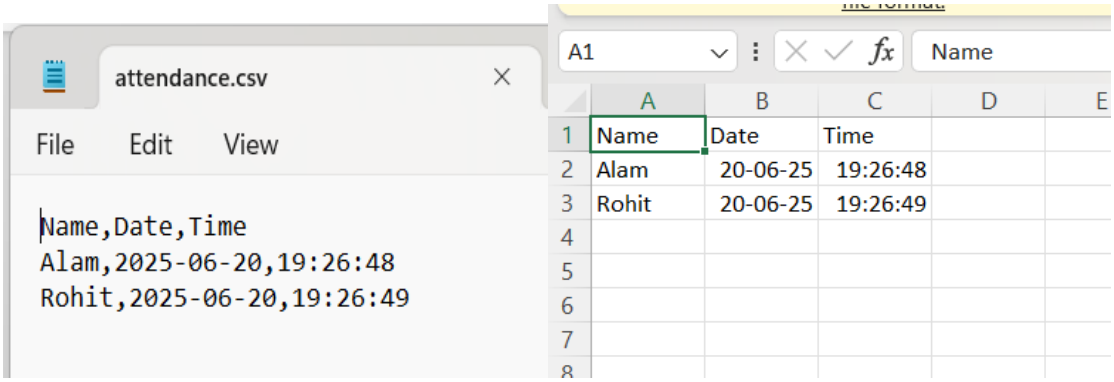
📸 **Output Images:**

1. **Real-Time Attendance Webcam Window.**

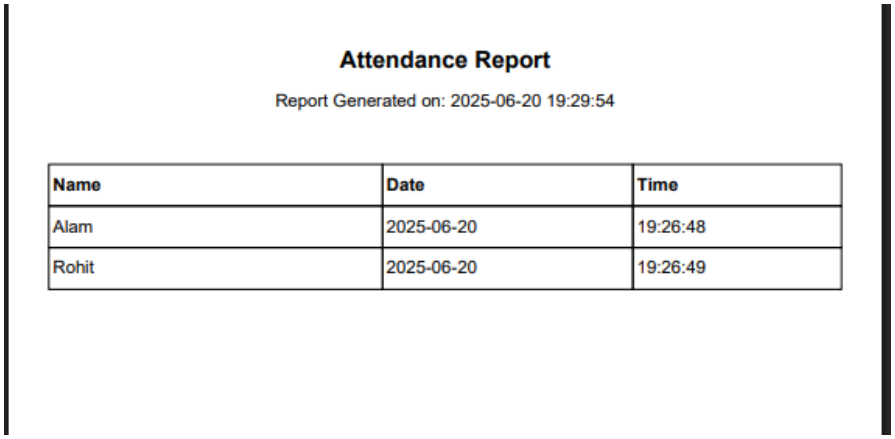




2. **CSV File Example (attendance.csv).**



3. **PDF Report Example (Attendance_Report.pdf).**



Attendance Report

Report Generated on: 2025-06-20 19:29:54

Name	Date	Time
Alam	2025-06-20	19:26:48
Rohit	2025-06-20	19:26:49

✅ **Final Flow:**

1. **train.py → Face images encode .**
2. **main.py → Webcam open karo → Faces detect karo → Attendance CSV banao → PDF auto generate .**

📌 **Conclusion:**

This project successfully demonstrates a **smart and automated face recognition-based attendance system** using Python, providing an efficient solution for modern classrooms, offices, and institutions.