# Face Recognition Attendance System

## Introduction

This document explains the functionality, features, and code implementation of the Face Recognition Attendance System. This system uses OpenCV and face\_recognition libraries to detect and recognize multiple faces simultaneously and mark attendance in an Excel file.

## Features

* ✔ Register new faces with unique names.
* ✔ Prevent duplicate face registrations.
* ✔ Mark attendance for multiple faces at the same time.
* ✔ Real-time attendance display.
* ✔ Delete registered persons.
* ✔ View attendance records of the last 24 hours.
* ✔ Show all registered persons in the system.

## Code Explanation

The system consists of several functions that perform face registration, face recognition, attendance marking, and user interactions via a Tkinter GUI.

### 1. Load Known Faces

This function loads all registered faces from the 'faces' directory and stores their encodings for comparison.

### 2. Register New Faces

Captures a face from the camera, checks if it is already registered, and saves it with a user-defined name.

### 3. Mark Attendance

Marks attendance in an Excel sheet by storing the name and timestamp.

### 4. Face Recognition

Detects and recognizes multiple faces in real-time and updates attendance accordingly.

### 5. GUI Implementation

Uses Tkinter to provide buttons for different functionalities like registration, attendance marking, and deleting persons.

## Complete Code

import cv2  
import face\_recognition  
import tkinter as tk  
from tkinter import simpledialog, messagebox, Toplevel  
import pandas as pd  
import os  
import threading  
from datetime import datetime  
from PIL import Image, ImageTk  
  
# Ensure directories exist  
if not os.path.exists('faces'):  
 os.makedirs('faces')  
if not os.path.exists('attendance'):  
 os.makedirs('attendance')  
  
attendance\_list = []  
cap = None # Global camera variable  
face\_encodings\_known = []  
face\_names\_known = []  
last\_attendance\_timestamp = None  
  
# Load known faces  
def load\_faces():  
 global face\_encodings\_known, face\_names\_known  
 face\_encodings\_known, face\_names\_known = [], []  
 for file in os.listdir('faces'):  
 if file.endswith('.jpg'):  
 image = face\_recognition.load\_image\_file(f'faces/{file}')  
 encodings = face\_recognition.face\_encodings(image)  
 if encodings:  
 face\_encodings\_known.append(encodings[0])  
 face\_names\_known.append(file.split('.')[0])  
  
# Check if the face is already registered  
def is\_duplicate\_face(face\_encoding):  
 matches = face\_recognition.compare\_faces(face\_encodings\_known, face\_encoding, tolerance=0.45)  
 return True in matches  
  
# Register new person  
def register\_person():  
 cap = cv2.VideoCapture(0)  
 if not cap.isOpened():  
 messagebox.showerror("Error", "Camera not accessible!")  
 return  
 while True:  
 ret, frame = cap.read()  
 cv2.imshow('Register - Press "s" to save', frame)  
 if cv2.waitKey(1) & 0xFF == ord('s'):  
 rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 encodings = face\_recognition.face\_encodings(rgb\_frame)  
 if not encodings:  
 messagebox.showerror("Error", "No face detected! Try again.")  
 break  
 if is\_duplicate\_face(encodings[0]):  
 messagebox.showerror("Error", "This face is already registered!")  
 break  
  
 name = simpledialog.askstring("Input", "Enter Name and Roll Number (e.g. John\_101):")  
 if name:  
 cv2.imwrite(f'faces/{name}.jpg', frame)  
 messagebox.showinfo("Success", f"{name} registered successfully!")  
 load\_faces()  
 break  
 cap.release()  
 cv2.destroyAllWindows()  
  
# Delete a person  
def delete\_person():  
 person = simpledialog.askstring("Delete", "Enter Name and Roll Number to delete:")  
 if person and os.path.exists(f'faces/{person}.jpg'):  
 os.remove(f'faces/{person}.jpg')  
 messagebox.showinfo("Deleted", f"{person} deleted successfully!")  
 load\_faces()  
 else:  
 messagebox.showerror("Error", "Person not found!")  
  
# Get today's attendance file  
def get\_attendance\_file\_path():  
 today = datetime.now().strftime("%Y-%m-%d")  
 return f'attendance/attendance\_{today}.xlsx'  
  
# Update attendance  
def mark\_attendance(name):  
 if name in attendance\_list:  
 return # Prevent duplicate attendance  
 attendance\_list.append(name)  
 timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
 file\_path = get\_attendance\_file\_path()  
 df = pd.DataFrame([[name, timestamp]], columns=['Name', 'Timestamp'])  
 if os.path.exists(file\_path):  
 df\_existing = pd.read\_excel(file\_path)  
 df = pd.concat([df\_existing, df], ignore\_index=True)  
 df.to\_excel(file\_path, index=False)  
 messagebox.showinfo("Attendance", f"{name}'s attendance recorded!")  
  
# Face recognition logic  
def recognize\_faces(frame):  
 rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 face\_locations = face\_recognition.face\_locations(rgb\_frame)  
 face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)  
  
 for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):  
 matches = face\_recognition.compare\_faces(face\_encodings\_known, face\_encoding, tolerance=0.45)  
 face\_distances = face\_recognition.face\_distance(face\_encodings\_known, face\_encoding)  
 best\_match = None if len(face\_distances) == 0 else face\_names\_known[face\_distances.argmin()]  
  
 if True in matches:  
 name = best\_match  
 color = (0, 255, 0)  
 mark\_attendance(name)  
 else:  
 name = "Unknown"  
 color = (0, 0, 255)  
  
 cv2.rectangle(frame, (left, top), (right, bottom), color, 2)  
 cv2.putText(frame, name, (left, top - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, color, 2)  
  
 return frame

**Code Breakdown**

**1. Importing Required Libraries**

python

CopyEdit

import cv2

import face\_recognition

import tkinter as tk

from tkinter import simpledialog, messagebox, Toplevel

import pandas as pd

import os

import threading

from datetime import datetime

from PIL import Image, ImageTk

* **cv2**: Handles webcam access and face detection.
* **face\_recognition**: Used for encoding and matching faces.
* **tkinter**: Provides a GUI for user interaction.
* **pandas**: Stores attendance in an Excel file.
* **os**: Manages file operations.
* **threading**: Ensures smooth execution.
* **datetime**: Handles timestamps.
* **PIL (Pillow)**: Displays images in the Tkinter GUI.

**2. Creating Required Directories**

python

CopyEdit

if not os.path.exists('faces'):

os.makedirs('faces')

if not os.path.exists('attendance'):

os.makedirs('attendance')

* Ensures that **"faces"** (for storing registered images) and **"attendance"** (for storing Excel files) folders exist.

**3. Global Variables**

python

CopyEdit

attendance\_list = []

cap = None

face\_encodings\_known = []

face\_names\_known = []

last\_attendance\_timestamp = None

* attendance\_list: Stores already marked attendance to prevent duplicates.
* cap: Stores the camera object.
* face\_encodings\_known: Stores encoded face data.
* face\_names\_known: Stores names of registered persons.
* last\_attendance\_timestamp: Keeps track of the last attendance time.

**4. Loading Known Faces**

python

CopyEdit

def load\_faces():

global face\_encodings\_known, face\_names\_known

face\_encodings\_known, face\_names\_known = [], []

for file in os.listdir('faces'):

if file.endswith('.jpg'):

image = face\_recognition.load\_image\_file(f'faces/{file}')

encodings = face\_recognition.face\_encodings(image)

if encodings:

face\_encodings\_known.append(encodings[0])

face\_names\_known.append(file.split('.')[0])

* Loads all registered faces from the faces folder.
* Extracts **face encodings** and stores them in face\_encodings\_known.
* Extracts **names** from file names and stores them in face\_names\_known.

**5. Checking for Duplicate Faces**

python

CopyEdit

def is\_duplicate\_face(face\_encoding):

matches = face\_recognition.compare\_faces(face\_encodings\_known, face\_encoding, tolerance=0.45)

return True in matches

* Compares a new face with existing ones.
* Returns True if a match is found, preventing duplicate registration.

**6. Registering a New Face**

python

CopyEdit

def register\_person():

cap = cv2.VideoCapture(0)

if not cap.isOpened():

messagebox.showerror("Error", "Camera not accessible!")

return

while True:

ret, frame = cap.read()

cv2.imshow('Register - Press "s" to save', frame)

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

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

encodings = face\_recognition.face\_encodings(rgb\_frame)

if not encodings:

messagebox.showerror("Error", "No face detected! Try again.")

break

if is\_duplicate\_face(encodings[0]):

messagebox.showerror("Error", "This face is already registered!")

break

name = simpledialog.askstring("Input", "Enter Name and Roll Number (e.g. John\_101):")

if name:

cv2.imwrite(f'faces/{name}.jpg', frame)

messagebox.showinfo("Success", f"{name} registered successfully!")

load\_faces()

break

cap.release()

cv2.destroyAllWindows()

* Captures a frame from the camera.
* Detects a face and checks if it's already registered.
* If it's a **new face**, asks for the **Name and Roll Number**.
* Saves the image and updates the database.

**7. Deleting a Person**

python

CopyEdit

def delete\_person():

person = simpledialog.askstring("Delete", "Enter Name and Roll Number to delete:")

if person and os.path.exists(f'faces/{person}.jpg'):

os.remove(f'faces/{person}.jpg')

messagebox.showinfo("Deleted", f"{person} deleted successfully!")

load\_faces()

else:

messagebox.showerror("Error", "Person not found!")

* Asks the user for a name.
* Deletes the corresponding **face image** from the system.

**8. Managing Attendance**

python

CopyEdit

def get\_attendance\_file\_path():

today = datetime.now().strftime("%Y-%m-%d")

return f'attendance/attendance\_{today}.xlsx'

* Generates a daily attendance file.

python

CopyEdit

def mark\_attendance(name):

if name in attendance\_list:

return

attendance\_list.append(name)

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

file\_path = get\_attendance\_file\_path()

df = pd.DataFrame([[name, timestamp]], columns=['Name', 'Timestamp'])

if os.path.exists(file\_path):

df\_existing = pd.read\_excel(file\_path)

df = pd.concat([df\_existing, df], ignore\_index=True)

df.to\_excel(file\_path, index=False)

messagebox.showinfo("Attendance", f"{name}'s attendance recorded!")

* Checks if the person is already marked.
* Saves attendance in an Excel sheet.

**9. Real-Time Face Recognition**

python

CopyEdit

def recognize\_faces(frame):

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

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

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

for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):

matches = face\_recognition.compare\_faces(face\_encodings\_known, face\_encoding, tolerance=0.45)

face\_distances = face\_recognition.face\_distance(face\_encodings\_known, face\_encoding)

best\_match = None if len(face\_distances) == 0 else face\_names\_known[face\_distances.argmin()]

if True in matches:

name = best\_match

color = (0, 255, 0)

mark\_attendance(name)

else:

name = "Unknown"

color = (0, 0, 255)

cv2.rectangle(frame, (left, top), (right, bottom), color, 2)

cv2.putText(frame, name, (left, top - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, color, 2)

return frame

* Detects **multiple faces** at once.
* Matches them with registered faces.
* Draws **green boxes** for known faces and **red boxes** for unknown faces.

**10. GUI and Camera Controls**

python

CopyEdit

def update\_frame():

global cap

ret, frame = cap.read()

if ret:

frame = recognize\_faces(frame)

frame = cv2.resize(frame, (640, 480))

img = ImageTk.PhotoImage(image=Image.fromarray(cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)))

canvas.create\_image(0, 0, anchor=tk.NW, image=img)

canvas.img = img

canvas.after(10, update\_frame)

* Continuously updates the **camera feed**.

**11. Running the System**

python

CopyEdit

load\_faces()

take\_attendance()

root.mainloop()

* **Loads registered faces**.
* **Starts real-time attendance**.
* **Opens the GUI**.

**Final Features**

✔ **Registers new faces with name & roll number**  
✔ **Prevents duplicate face registration**  
✔ **Marks attendance for multiple faces at once**  
✔ **Stores records in an Excel file**  
✔ **Allows viewing and deleting registered faces**  
✔ **Real-time face detection with Ui**