**CODE EXPLANATION**

* **Algorithm Used:** Haar cascade Classifier / Algorithm
* **Libraries/Modules used:**
  + - * 1. Tk – tools
        2. Datetime
        3. Pytest – shutil
        4. Python-csv
        5. Numpy
        6. Pillow
        7. Pandas

**1.Machine Learning Approach**:

* Utilizes a cascade function trained with positive and negative images.
* Positive images contain the object of interest (e.g., faces), while negative images do not.

**2. Training Process**:

* Requires many positive images (images with faces) and negative images (images without faces).
* Extracts features such as eyes, nose, mouth, and other facial characteristics from positive images.
* Learns to identify features common in positive images but rare in negative images.

**3. Haar Cascade Classifier**:

* The trained classifier is saved in an XML file (e.g., "haarcascade\_frontalface\_default.xml").
* This XML file contains information about the trained classifiers and features.

**4**. **Face Detection**:

* The XML file is loaded to apply the Haar cascade classifier.
* The algorithm scans input images to detect regions matching the learned features.

**5. Post-Detection**: Once a potential face is detected, further analysis can be done for facial recognition, emotion detection, etc.

**OVERVIEW OF LIBARARIES AND MODULES:**

**Tk**:

* Provides tools for creating graphical user interfaces (GUIs) in Python.

**Datetime**:

* Supplies classes for manipulating dates and times.

**Pytest**:

* A testing framework used for writing simple and scalable test cases in Python.

**Shutil**:

* Offers functions for high-level file operations like copying and removing files.

**Python-csv**:

* Facilitates reading from and writing to CSV (Comma Separated Values) files.

**Numpy**:

* Provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on them.

**Pillow**:

* A library for opening, manipulating, and saving various image file formats.

**CODE:**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mess

from tkinter import PhotoImage

from PIL import Image, ImageTk

import tkinter.simpledialog as tsd

import cv2, os

import csv

import numpy as np

from PIL import Image

import pandas as pd

import datetime

import time

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

**# Function to ensure that a directory exists**

def assure\_path\_exists(path):

dir = os.path.dirname(path)

if not os.path.exists(dir):

os.makedirs(dir)

**# Function to update the clock label with the current time every second**

def tick():

current\_time = time.strftime('%I:%M:%S %p')

clock.config(text=current\_time)

clock.after(1000, tick)

**# Function to display contact information**

def contact():

mess.\_show(title='Contact us', message="Please contact us on : 'shubhamkumar8180323@gmail.com' ")

**# Function to check if the Haar Cascade XML file exists**

def check\_haarcascadefile():

exists = os.path.isfile("haarcascade\_frontalface\_default.xml")

if not exists:

mess.\_show(title='Some file missing', message='Please contact us for help')

window.destroy()

**# Function to save or change the application password**

def save\_pass():

assure\_path\_exists("TrainingImageLabel/")

exists1 = os.path.isfile("TrainingImageLabel\psd.txt")

if exists1:

tf = open("TrainingImageLabel\psd.txt", "r")

key = tf.read()

else:

master.destroy()

new\_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show='\*')

if new\_pas is None:

mess.\_show(title='No Password Entered', message='Password not set!! Please try again')

else:

tf = open("TrainingImageLabel\psd.txt", "w")

tf.write(new\_pas)

mess.\_show(title='Password Registered', message='New password was registered successfully!!')

return

op = (old.get())

newp = (new.get())

nnewp = (nnew.get())

if (op == key):

if (newp == nnewp):

txf = open("TrainingImageLabel\psd.txt", "w")

txf.write(newp)

else:

mess.\_show(title='Error', message='Confirm new password again!!!')

return

else:

mess.\_show(title='Wrong Password', message='Please enter correct old password.')

return

mess.\_show(title='Password Changed', message='Password changed successfully!!')

master.destroy()

**# Function to create a dialog for changing the password**

def change\_pass():

global master

master = tk.Tk()

master.geometry("400x160")

master.resizable(False, False)

master.title("Change Password")

master.configure(background="white")

# Create labels and entry fields for old and new passwords

lbl4 = tk.Label(master, text=' Enter Old Password', bg='white', font=('comic', 12, ' bold '))

lbl4.place(x=10, y=10)

global old

old = tk.Entry(master, width=25, fg="black", relief='solid', font=('comic', 12, ' bold '), show='\*')

old.place(x=180, y=10)

lbl5 = tk.Label(master, text=' Enter New Password', bg='white', font=('comic', 12, ' bold '))

lbl5.place(x=10, y=45)

global new

new = tk.Entry(master, width=25, fg="black", relief='solid', font=('comic', 12, ' bold '), show='\*')

new.place(x=180, y=45)

lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('comic', 12, ' bold '))

lbl6.place(x=10, y=80)

global nnew

nnew = tk.Entry(master, width=25, fg="black", relief='solid', font=('comic', 12, ' bold '), show='\*')

nnew.place(x=180, y=80)

# Create buttons for saving or canceling the password change

cancel = tk.Button(master, text="Cancel", command=master.destroy, fg="black", bg="red", height=1, width=25, activebackground="white", font=('comic', 10, ' bold '))

cancel.place(x=200, y=120)

save1 = tk.Button(master, text="Save", command=save\_pass, fg="black", bg="#00fcca", height=1, width=25, activebackground="white", font=('comic', 10, ' bold '))

save1.place(x=10, y=120)

master.mainloop()

**# Function to prompt for the password if it doesn't exist and start training images**

def psw():

assure\_path\_exists("TrainingImageLabel/")

exists1 = os.path.isfile("TrainingImageLabel\psd.txt")

if exists1:

tf = open("TrainingImageLabel\psd.txt", "r")

key = tf.read()

else:

new\_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show='\*')

if new\_pas is None:

mess.\_show(title='No Password Entered', message='Password not set!! Please try again')

else:

tf = open("TrainingImageLabel\psd.txt", "w")

tf.write(new\_pas)

mess.\_show(title='Password Registered', message='New password was registered successfully!!')

return

password = tsd.askstring('Password', 'Enter Password', show='\*')

if (password == key):

TrainImages()

elif (password is None):

pass

else:

mess.\_show(title='Wrong Password', message='You have entered wrong password')

**# Function to clear the input fields and reset the message label**

def clear():

txt.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

**# Function to clear the second input field and reset the message label**

def clear2():

txt2.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

**# Function to capture images of the student and save them**

def TakeImages():

check\_haarcascadefile()

columns = ['SERIAL NO.', '', 'ID', '', 'NAME']

assure\_path\_exists("StudentDetails/")

assure\_path\_exists("TrainingImage/")

serial = 0

exists = os.path.isfile("StudentDetails\StudentDetails.csv")

if exists:

with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:

reader1 = csv.reader(csvFile1)

for l in reader1:

serial = serial + 1

serial = (serial // 2)

else:

with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile1:

writer = csv.writer(csvFile1)

writer.writerow(columns)

serial = 1

Id = (txt.get())

name = (txt2.get())

if (name.isalpha() or (' ' in name)):

cam = cv2.VideoCapture(0)

harcascadePath = "haarcascade\_frontalface\_default.xml"

detector = cv2.CascadeClassifier(harcascadePath)

sampleNum = 0

while (True):

ret, img = cam.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = detector.detectMultiScale(gray, 1.3, 5)

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

sampleNum = sampleNum + 1

cv2.imwrite("TrainingImage/ " + name + "." + Id + '.' + str(sampleNum) + ".jpg", gray[y:y + h, x:x + w])

cv2.imshow('Taking Images', img)

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

break

elif (sampleNum > 60):

break

cam.release()

cv2.destroyAllWindows()

res = "Images Saved for ID : " + Id + " Name : " + name

row = [serial, '', Id, '', name]

with open("StudentDetails\StudentDetails.csv", 'a+') as csvFile:

writer = csv.writer(csvFile)

writer.writerow(row)

message1.configure(text=res)

else:

if (name == ''):

message1.configure(text="Please enter a valid name")

else:

message1.configure(text="Name should be alphabetic")

**# Function to train the face recognition model with the captured images**

def TrainImages():

check\_haarcascadefile()

assure\_path\_exists("TrainingImageLabel/")

faces, Ids = getImagesAndLabels('TrainingImage')

recognizer.train(faces, np.array(Ids))

recognizer.save('TrainingImageLabel\trainer.yml')

res = "Profile Saved Successfully"

message1.configure(text=res)

**# Function to extract faces and IDs from images in the specified path**

def getImagesAndLabels(path):

imagePaths = [os.path.join(path, f) for f in os.listdir(path)]

faces = []

Ids = []

for imagePath in imagePaths:

img = Image.open(imagePath).convert('L')

imgNp = np.array(img, 'uint8')

Id = int(os.path.split(imagePath)[-1].split('.')[1])

faces.append(imgNp)

Ids.append(Id)

return faces, Ids

**# Function to recognize faces from webcam feed and update attendance records**

def TrackImages():

check\_haarcascadefile()

assure\_path\_exists("Attendance/")

cam = cv2.VideoCapture(0)

recognizer.read('TrainingImageLabel\trainer.yml')

harcascadePath = "haarcascade\_frontalface\_default.xml"

detector = cv2.CascadeClassifier(harcascadePath)

font = cv2.FONT\_HERSHEY\_SIMPLEX

ids = pd.read\_csv("StudentDetails\StudentDetails.csv")

ids = ids.iloc[::2]

ids.columns = ['S.NO', 'ID', 'NAME']

while True:

ret, img = cam.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = detector.detectMultiScale(gray, 1.3, 5)

for (x, y, w, h) in faces:

id, confidence = recognizer.predict(gray[y:y + h, x:x + w])

if (confidence < 100):

name = ids.loc[ids['ID'] == id]['NAME'].values

name = name[0]

else:

name = 'Unknown'

cv2.putText(img, str(name), (x, y + h), font, 1, (255, 0, 0), 2)

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

markAttendance(id, name)

cv2.imshow('Face', img)

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

break

cam.release()

cv2.destroyAllWindows()

**# Function to mark the attendance of recognized faces in the CSV file**

def markAttendance(id, name):

today = datetime.date.today()

dt = time.strftime('%H:%M:%S')

exist = os.path.isfile("Attendance\Attendance.csv")

if not exist:

with open("Attendance\Attendance.csv", 'a+') as csvFile:

writer = csv.writer(csvFile)

writer.writerow(['ID', 'NAME', 'DATE', 'TIME'])

with open("Attendance\Attendance.csv", 'a+') as csvFile:

writer = csv.writer(csvFile)

writer.writerow([id, name, today, dt])

**# Function to send an email with the attendance CSV file attached**

def send\_email():

from\_email = 'your\_email@example.com'

password = 'your\_password'

to\_email = to\_email\_var.get()

domain = domain\_var.get()

subject = 'Attendance Report'

body = 'Please find attached the attendance report.'

filename = 'Attendance/Attendance.csv'

if not os.path.isfile(filename):

mess.\_show(title='Error', message='Attendance file not found.')

return

msg = MIMEMultipart()

msg['From'] = from\_email

msg['To'] = to\_email

msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

attachment = open(filename, 'rb')

part = MIMEBase('application', 'octet-stream')

part.set\_payload(attachment.read())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', f'attachment; filename={filename}')

msg.attach(part)

try:

server = smtplib.SMTP('smtp.' + domain, 587)

server.starttls()

server.login(from\_email, password)

text = msg.as\_string()

server.sendmail(from\_email, to\_email, text)

mess.\_show(title='Email Sent', message='Attendance report sent successfully!')

except Exception as e:

mess.\_show(title='Error', message=str(e))

finally:

server.quit()

**# Create the main window**

window = tk.Tk()

window.title("Face Recognition Based Attendance System")

window.geometry('700x600')

window.configure(bg='white')

**# Display the current time**

clock = tk.Label(window, font=('calibri', 40, 'bold'), background='purple', foreground='white')

clock.pack()

tick()

**# Add buttons and other UI elements**

**# For example:**

btn1 = tk.Button(window, text="Take Images", command=TakeImages)

btn1.pack()

btn2 = tk.Button(window, text="Train Images", command=TrainImages)

btn2.pack()

btn3 = tk.Button(window, text="Track Images", command=TrackImages)

btn3.pack()

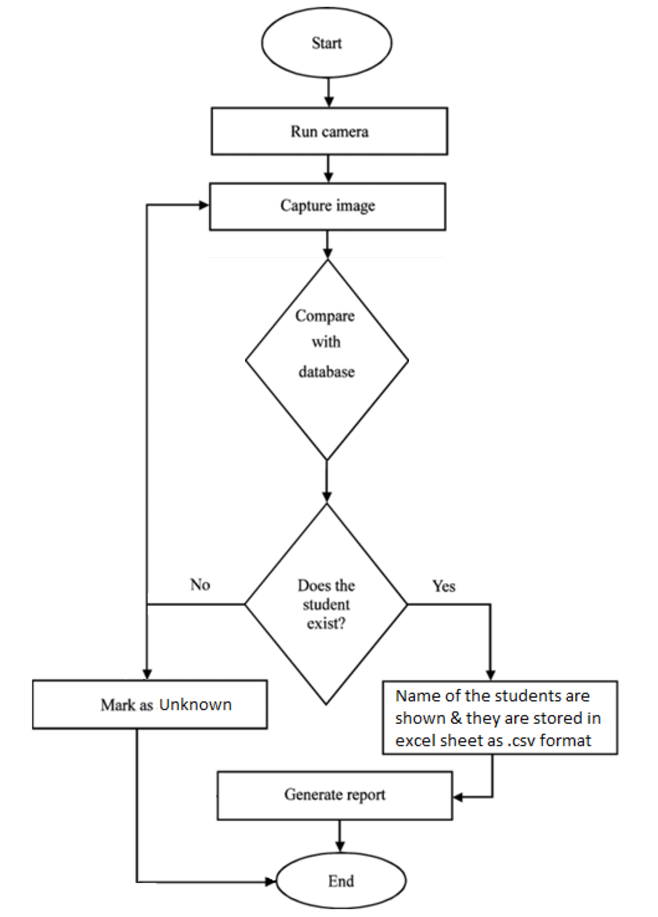
btn4 = tk.Button(window, text="Send Email", command=send\_email)

btn4.pack()

**# Run the application**

window.mainloop()

**FLOW CHART OF THE CODE:**



**Step 1:** Click on the python attendance application to run the program.

**Step 2:** Register user by providing the name and ID.

**Step 3:** Take images of the users by scanning it from the application.

**Step 4:** Click on the “Take attendance” in order to scan the images for the attendance

**Step 5:** Close the application once the attendance has been recorded.

**Step 6:**The attendance can also be viewed from the Excel sheet.

