```
# importing libraries

Import tkinter as tk

From tkinter import Message, Text

Import cv2

Import os

Import shutil

Import csv

Import numpy as np

From PIL import Image, ImageTk

Import pandas as pd

Import datetime

Import time

Import tkinter.ttk as ttk

Import tkinter.font as font
```

```
From pathlib import Path


Window = tk.Tk()

Window.title("Face_Recogniser")

Window.configure(background ='white')

Window.grid_rowconfigure(0, weight = 1)

Window.grid_columnconfigure(0, weight = 1)

Message = tk.Label(

   Window, text ="Face-Recognition-System",

   Bg ="green", fg = "white", width = 50,

   Height = 3, font = ('times', 30, 'bold'))



Message.place(x = 200, y = 20)

Lbl = tk.Label(window, text = "No.",
```

```
Width = 20, height = 2, fg =”green”,

Bg = “white”, font = (‘times’, 15, ‘ bold ‘) )

Lbl.place(x = 400, y = 200)


Txt = tk.Entry(window,

Width = 20, bg =”white”,

Fg =”green”, font = (‘times’, 15, ‘ bold ‘))

Txt.place(x = 700, y = 215)


Lbl2 = tk.Label(window, text =”Name”,

Width = 20, fg =”green”, bg =”white”,

Height = 2, font =(‘times’, 15, ‘ bold ‘))

Lbl2.place(x = 400, y = 300)


Txt2 = tk.Entry(window, width = 20,

Bg =”white”, fg =”green”,
```

```
Font = (‘times’, 15, ‘ bold ‘)  )

Txt2.place(x = 700, y = 315)

# The function below is used for checking
# whether the text below is number or not ?

Def is_number(s):
   Try:
      Float(s)
      Return True
   Except ValueError:
      Pass


   Try:
      Import unicodedata
      Unicodedata.numeric(s)
      Return True
```

```
    Except (TypeError, ValueError):

        Pass



    Return False
# Take Images is a function used for creating
# the sample of the images which is used for
# training the model. It takes 60 Images of
# every new user.

Def TakeImages():



    # Both ID and Name is used for recognising the Image

    Id =(txt.get())

    Name =(txt2.get())



    # Checking if the ID is numeric and name is Alphabetical

    If(is_number(Id) and name.isalpha()):

        # Opening the primary camera if you want to access
```

```
# the secondary camera you can mention the number

# as 1 inside the parenthesis

Cam = cv2.VideoCapture(0)

# Specifying the path to haarcascade file

harcascadePath = “data\haarcascade_frontalface_default.xml”

# Creating the classier based on the haarcascade file.

Detector = cv2.CascadeClassifier(harcascadePath)

# Initializing the sample number(No. Of images) as 0

sampleNum = 0

while(True):

    # Reading the video captures by camera frame by frame

    Ret, img = cam.read()

    # Converting the image into grayscale as most of

    # the the processing is done in gray scale format
```

```
Gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



# It converts the images in different sizes

# (decreases by 1.3 times) and 5 specifies the

# number of times scaling happens

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



# For creating a rectangle around the image

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

    # Specifying the coordinates of the image as well

    # as color and thickness of the rectangle.

    # incrementing sample number for each image

    Cv2.rectangle(img, (x, y), (

        X + w, y + h), (255, 0, 0), 2)

    sampleNum = sampleNum + 1
```

```
    # saving the captured face in the dataset folder

    # TrainingImage as the image needs to be trained

    # are saved in this folder

    Cv2.imwrite(

      "TrainingImage\ "+name +"."+Id +'.'+ str(

        sampleNum) + ".jpg", gray[y:y + h, x:x + w])

    # display the frame that has been captured

    # and drawn rectangle around it.

    Cv2.imshow('frame', img)

# wait for 100 milliseconds

If cv2.waitKey(100) & 0xFF == ord('q'):

    Break

# break if the sample number is more than 60

Elif sampleNum>60:
```

```
        Break

# releasing the resources

Cam.release()

# closing all the windows

Cv2.destroyAllWindows()

# Displaying message for the user

Res = “Images Saved for ID : “ + Id +” Name : “+ name

# Creating the entry for the user in a csv file

Row = [Id, name]

With open(‘UserDetails\UserDetails.csv’, ‘a+’) as csvFile:

   Writer = csv.writer(csvFile)

   # Entry of the row in csv file

   Writer.writerow(row)

csvFile.close()

message.configure(text = res)
```

```
    else:

        if(is_number(Id)):

            res = “Enter Alphabetical Name”

            message.configure(text = res)

        if(name.isalpha()):

            res = “Enter Numeric Id”

            message.configure(text = res)


# Training the images saved in training image folder

Def TrainImages():

    # Local Binary Pattern Histogram is an Face Recognizer

    # algorithm inside OpenCV module used for training the image dataset

    Recognizer = cv2.face.LBPHFaceRecognizer_create()

    # Specifying the path for HaarCascade file

    harcascadePath = “data\haarcascade_frontalface_default.xml”
```

```
    # creating detector for faces

    Detector = cv2.CascadeClassifier(harcascadePath)

    # Saving the detected faces in variables

    Faces, Id = getImagesAndLabels("TrainingImage")

    # Saving the trained faces and their respective ID's

    # in a model named as "trainner.yml".

    Recognizer.train(faces, np.array(Id))

    Recognizer.save("TrainingImageLabel\Trainner.yml")

    # Displaying the message

    Res = "Image Trained"

    Message.configure(text= res)


Def getImagesAndLabels(path):

    # get the path of all the files in the folder

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

```
faces =[]

# creating empty ID list

Ids =[]

# now looping through all the image paths and loading the

# Ids and the images saved in the folder

For imagePath in imagePaths:

    # loading the image and converting it to gray scale

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

    # Now we are converting the PIL image into numpy array

    imageNp = np.array(pilImage, 'uint8')

    # getting the Id from the image

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

    # extract the face from the training image sample

    Faces.append(imageNp)
```

```
        Ids.append(Id)

    Return faces, Ids
# For testing phase

Def TrackImages():

    Recognizer = cv2.face.LBPHFaceRecognizer_create()

    # Reading the trained model

    Recognizer.read("TrainingImageLabel\Trainner.yml")

    harcascadePath = "data\haarcascade_frontalface_default.xml"

    faceCascade = cv2.CascadeClassifier(harcascadePath)

    # getting the name from "userdetails.csv"

    Df = pd.read_csv("UserDetails\UserDetails.csv")

    Cam = cv2.VideoCapture(0)

    Font = cv2.FONT_HERSHEY_SIMPLEX

    While True:

        Ret, im = cam.read()
```

```
Gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

Faces = faceCascade.detectMultiScale(gray, 1.2, 5)

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

  Cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)

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

  If(conf < 50):

    Aa = df.loc[df['Id'] == Id]['Name'].values

    Tt = str(Id)+"-"+aa

  Else:

    Id ='Unknown'

    Tt = str(Id)

  If(conf > 75):

    noOfFile = len(os.listdir("ImagesUnknown"))+1

    cv2.imwrite("ImagesUnknown\Image"+

    str(noOfFile) + ".jpg", im[y:y + h, x:x + w])
```

```
            cv2.putText(im, str(tt), (x, y + h),

            font, 1, (255, 255, 255), 2)

        cv2.imshow('im', im)

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

            break

    cam.release()

    cv2.destroyAllWindows()
```

```
takeImg = tk.Button(window, text ="Sample",

command = TakeImages, fg ="white", bg ="green",

width = 20, height = 3, activebackground = "Red",

font =('times', 15, ' bold '))

takeImg.place(x = 200, y = 500)
```

```
trainImg = tk.Button(window, text ="Training",

command = TrainImages, fg ="white", bg ="green",

width = 20, height = 3, activebackground = "Red",

font =('times', 15, ' bold '))

trainImg.place(x = 500, y = 500)

trackImg = tk.Button(window, text ="Testing",

command = TrackImages, fg ="white", bg ="green",

width = 20, height = 3, activebackground = "Red",

font =('times', 15, ' bold '))

trackImg.place(x = 800, y = 500)

quitWindow = tk.Button(window, text ="Quit",

command = window.destroy, fg ="white", bg ="green",

width = 20, height = 3, activebackground = "Red",

font =('times', 15, ' bold '))

quitWindow.place(x = 1100, y = 500)
```

```
window.mainloop()
```