The Code For Our Activity Tracker Project

import os

import time

import tkinter as tk

from tkinter import messagebox

from datetime import datetime

from tkinter import Menu, ttk ,filedialog

import string

from cv2 import cv2

import face\_recognition

import numpy as np

import shutil

import smtplib as smtp

import random

window1 = tk.Tk()

window1.geometry("1280x720")

window1.minsize(1080,720)

window1.title("Employee Activity Tracker")

window1.configure(background = 'gray')

####################################################### Registration Window #########################################################

class Registration:

    def \_\_init\_\_(self):

        self.img\_new =[]

        self.browse\_pic\_label = False

    def Take\_Picture(self):

        path = 'images'

        mylist = os.listdir(path)

        if len(mylist)>0:                 ##################### Checking path folder have already an image or not ###################

            self.button1\_label.config(text = "\* your are already register!!!!")

            return 0

        self.entry3\_label.config(text = "")

        cap = cv2.VideoCapture(0)

        if not cap.isOpened():

            raise Exception(" Could not open camera ")

        while True:

            success, img = cap.read()

            cv2.imshow('WebCam',img)

            key = cv2.waitKey(1)

             ################################### Validating NAME ####################

            NAME = self.entry2.get()

            if len(NAME) == 0 :

                self.entry2\_label.config(text = "\* Please Your Name!!!!!!")

                cap.release()

                cv2.destroyAllWindows()

                break

            else:

                self.entry2\_label.config(text = "")

            if key == ord('s') or key == ord('S') :

                self.img\_new = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

                cv2.imshow(' Showing Clicked Picture ',self.img\_new)

                cv2.waitKey(2000)

                cap.release()

                cv2.destroyAllWindows()

                self.button1\_label.config(text = "......Picture Taken....",fg = 'green')

                if self.browse\_pic\_label:

                    self.browse\_pic\_label = False

                    self.var1.set("")

                    self.entry3\_label.config(text = "\* browsed Picture is clear!!!!",fg = 'blue')

                break

            elif key == ord('q') or key == ord('Q'):

                cap.release()

                cv2.destroyAllWindows()

                break

    def Save\_Profile(self):

        ################################### Validating ID ####################

        ID = self.entry1.get()

        if len(ID) == 0 :

            self.entry1\_label.config(text = "\* Please Your Employee ID!!!!!!")

        else:

            self.entry1\_label.config(text = "")

        ################################### Validating NAME ####################

        NAME = self.entry2.get()

        if len(NAME) == 0 :

            self.entry2\_label.config(text = "\* Please Your Name!!!!!!")

        else:

            self.entry2\_label.config(text = "")

        ################################### Validating DURATION ####################

        DURATION = self.var.get()

        if DURATION == "HH:MM" :

            self.drop1\_label.config(text = "\* Please select duration!!!!!!")

        else:

            self.drop1\_label.config(text = "")

        ################################### Validating Browse Picture and Take picture ####################

        file\_loc = self.entry3.get()

        if (not os.path.isfile(file\_loc)) and (len(self.img\_new) == 0):

            self.entry3\_label.config(text = "\* Please select file or Take Picture")

            self.button1\_label.config(text = "\* Please select file or Take Picture ")

        else :

            ############################################# Saving Picture ################################################

            self.pic\_save = False

            if os.path.isfile(file\_loc):                            #################### Checking file\_loc exist or not #########################

                path = 'images'

                mylist = os.listdir(path)

                self.button1\_label.config(text = "")

                if len(mylist)>0:                                   ##################### Checking path folder have already an image or not #####

                    self.entry3\_label.config(text = "\* your pic is already register!!!!")

                elif len(self.img\_new) > 0:

                    self.entry3\_label.config(text = "\* Either you can Browse pic or Take Picture.",fg = 'blue')

                    self.img\_new = []

                    self.button1\_label.config(text = "\* Taken picture clear. if you want to register you taken picture, Please Click on <<Take Picture>>",fg = 'blue')

                    self.browse\_pic\_label = True

                else:

                    if messagebox.askquestion("Comfirm","Are You Sure?") == 'yes':

                        with open('Details.txt','w') as f:

                            detail =ID + "\n" + NAME + "\n" + DURATION

                            f.writelines(detail)

                        shutil.copy(file\_loc, path)                     ##################### Copying photo to file\_loc to path #####################

                        self.entry3\_label.config(text = "\* your pic is register.....",fg = 'green')

                        messagebox.showinfo("Done","Successfully Register \n Now please go to Home page..")

                    else:

                        return 0

            else:

                self.entry3\_label.config(text = "")

                if messagebox.askquestion("Comfirm","Are You Sure?") == 'yes':

                    with open('Details.txt','w') as f:

                        detail =ID + "\n" + NAME + "\n" + DURATION

                        f.writelines(detail)

                    path = 'images\\'

                    image\_name = path + NAME + ".jpg"

                    self.pic\_save = cv2.imwrite(filename = image\_name, img = self.img\_new)

                    messagebox.showinfo("Done","Successfully Register \n Now please go to Home page..")

                else:

                    return 0

    def register\_new\_employee(self,window1 = window1):

        window1.destroy()

        def Go\_to\_home\_page():

            window1 = tk.Tk()

            window1.geometry("1280x720")

            window1.minsize(1080,720)

            window1.title("Employee Activity Tracker")

            window1.configure(background = 'gray')

            home\_page(window1)

            self.registration\_window.destroy()

        self.registration\_window = tk.Tk()

        self.registration\_window.geometry("1280x720")

        self.registration\_window.minsize(1280,720)

        self.registration\_window.title("Employee Activity Tracker")

        self.registration\_window.iconbitmap("icon.ico")

        self.registration\_window.configure(background = 'gray')

        frame1 = tk.Frame(self.registration\_window, bg = 'gray')

        frame1.place(relx = 0.11,rely = 0, relwidth = 0.80, relheight = 0.118)

        l1 = tk.Label(frame1,text = "Face Recognition Based Employee Activity Tracker",bg = 'gray', fg = "black",font = ('verdana',20,'bold'))

        l1.pack(anchor = 'center',pady = 4 )

        l2 = tk.Label(frame1,text = "New Registration",bg = 'gray', fg = "green",font = ('verdana',16,'bold'))

        l2.pack(anchor = 'center',pady = 4 ,side = 'bottom')

        #################################################### Frame2  #################################

        frame2 = tk.Frame(self.registration\_window,bg= 'gray')

        frame2.place(relx = 0.2,rely = 0.12, relwidth = 0.75, relheight = 0.5)

        l3 = tk.Label(frame2,text = "Employee ID",bg = 'gray', fg = "white",font = ('verdana',16,'bold'))

        l3.grid(pady=20,padx=30,row=0,column=0,sticky='w')

        self.entry1 = tk.Entry(frame2,bd = 4, font =('verdana',12,'bold'))

        self.entry1.grid(pady=20,padx=20,row=0,column=1,sticky='w')

        self.entry1\_label = tk.Label(frame2,text = "",bg = 'gray', fg = "red",font = ('verdana',8,'bold'))

        self.entry1\_label.grid(pady=20,padx=30,row=0,column=3,sticky='w')

        l4 = tk.Label(frame2,text = "Employee Name",bg = 'gray', fg = "white",font = ('verdana',16,'bold'))

        l4.grid(pady=20,padx=30,row=1,column=0,sticky='w')

        self.entry2 = tk.Entry(frame2,bd = 4, font =('verdana',12,'bold'))

        self.entry2.grid(pady=20,padx=20,row=1,column=1,sticky='w')

        self.entry2\_label = tk.Label(frame2,text = "",bg = 'gray', fg = "red",font = ('verdana',8,'bold'))

        self.entry2\_label.grid(pady=20,padx=30,row=1,column=3,sticky='w')

        l4 = tk.Label(frame2,text = "Set Working Duration",bg = 'gray', fg = "white",font = ('verdana',16,'bold'))

        l4.grid(pady=20,padx=30,row=2,column=0,sticky='w')

        self.var = tk.StringVar()

        self.var.set("HH:MM")

        self.drop1 = tk.OptionMenu(frame2,self.var,"01:00","02:00","03:00","04:00","05:00","06:00","07:00","08:00","09:00","10:00","11:00","12:00")

        self.drop1.grid(pady=20,padx=20,row=2,column=1,sticky='w')

        self.drop1\_label = tk.Label(frame2,text = "",bg = 'gray', fg = "red",font = ('verdana',8,'bold'))

        self.drop1\_label.grid(pady=20,padx=30,row=2,column=3,sticky='w')

        ############################################# Browsing Picture from file location ##############################

        self.var1 = tk.StringVar()

        self.var1.set("Select file location")

        self.entry3 = tk.Entry(frame2,bd = 4, font =('verdana',12),textvariable = self.var1)

        self.entry3.grid(pady=20,padx=20,row=3,column=1,sticky='w')

        def Browse\_pic():

            self.registration\_window.filedialog1  = filedialog.askopenfilename(initialdir ="Desktop",title = "Select your photo",filetypes = (("jpg file","\*.jpg"),("jpeg file","\*.jpeg"),("png file","\*.png")))

            self.var1.set(self.registration\_window.filedialog1)

        button2 = tk.Button(frame2,text = 'Browse Picture', bd=2,bg = 'gray', fg = "black" ,command = Browse\_pic,font = ('verdana',12,'bold'))

        button2.grid(pady=20,padx=20,row=3,column=0,sticky='w')

        self.entry3\_label = tk.Label(frame2,text = "",bg = 'gray', fg = "red",font = ('verdana',8,'bold'))

        self.entry3\_label.grid(pady=20,padx=30,row=3,column=3,sticky='w')

        #################################################### Frame3  #################################

        self.frame3 = tk.Frame(self.registration\_window, bg = 'gray')

        self.frame3.place(relx = 0.15,rely = 0.5, relwidth = 0.8, relheight = 0.34)

        l5 = tk.Label(self.frame3, text = 'OR',font = ('verdana',12,'bold'),bg = '#262523',fg = 'white')

        l5.pack(anchor = 'center',pady =5 )

        button1 = tk.Button(self.frame3,text = 'Take Picture', bd=4, command = self.Take\_Picture,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 2, width = 35)

        button1.pack(anchor = 'center')

        self.button1\_label = tk.Label(self.frame3,text = "Press 's' to take picture and 'q' to close WebCam",bg = 'gray', fg = "blue",font = ('verdana',8,'bold'))

        self.button1\_label.pack(anchor = 'center',pady =10)

        button3 = tk.Button(self.frame3,text = 'Save Profile', bd=4, command = self.Save\_Profile,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 2, width = 35)

        button3.pack(anchor = 'center',pady =5)

        self.button3\_label = tk.Label(self.frame3,text = "",bg = 'gray', fg = "red",font = ('verdana',8,'bold'))

        self.button3\_label.pack(anchor = 'center')

        frame4 = tk.Frame(self.registration\_window,bg= 'gray')

        frame4.place(relx = 0.11,rely = 0.9, relwidth = 0.80, relheight = 0.14)

        button1 = tk.Button(frame4,text = 'Quit', bd=4, command = self.registration\_window.destroy,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 10)

        button1.grid(pady=20,padx=20,row=0,column=1,sticky='e')

        button2 = tk.Button(frame4,text = 'Go to home page', bd=4, command = Go\_to\_home\_page,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 15)

        button2.grid(pady=20,padx=20,row=0,column=0,sticky='w')

        self.registration\_window.mainloop()

###################################################### End of Registration Window ###################################################

###################################################### About Window #################################################################

def About(window1, about\_window):

    window1.destroy()

    def Go\_to\_home\_page():

        window1 = tk.Tk()

        window1.geometry("1280x720")

        window1.minsize(1080,720)

        window1.title("Employee Activity Tracker")

        window1.configure(background = 'gray')

        about\_window.destroy()

        home\_page(window1)

    #################################################### Frame1  #################################

    frame1 = tk.Frame(about\_window, bg = 'gray')

    frame1.place(relx = 0.11,rely = 0, relwidth = 0.80, relheight = 0.118)

    l1 = tk.Label(frame1,text = "Face Recognition Based Employee Activity Tracker",bg = 'gray', fg = "black",font = ('verdana',20,'bold'))

    l1.pack(anchor = 'center',pady = 4 )

    #################################################### Frame2  #################################

    frame2 = tk.Frame(about\_window,bg= 'gray')

    frame2.place(relx = 0.11,rely = 0.12, relwidth = 0.80, relheight = 0.75)

    l2 = tk.Label(frame2,text ='''This software aims to provide a realtime continuous attendance system using the concept of machine Learning.\n

    The software is designed and built by Prince , Tirthoraj, Shubham and Binay of BCA final year session 2018-2021. \n\n

    The software notes the user's presence continuously after certain interval of time making it an efficient way to     \n

    ensure Employees availability through out the working hours and hence ensures organisation's productivity.          \n\n

    ''',bg = 'gray', fg = "black",font = ('verdana',12))

    l2.pack(anchor = 'center',pady = 15 ,side = 'top')

    tk.Label(frame2,text = '''Contact us :-''',bg = 'gray', fg = "black",font = ('verdana',12)).pack(anchor = 'w',pady = 15,padx = 50)

    tk.Label(frame2,text = '''•Prince :: pk03215@gmail.com :: 9504008839''',bg = 'gray', fg = "black",font = ('verdana',12)).pack(anchor = 'w',pady = 15,padx = 150)

    tk.Label(frame2,text = '''•Tirthoraj :: tirthorajdasgupta@gmail.com :: 8434116014''',bg = 'gray', fg = "black",font = ('verdana',12)).pack(anchor = 'w',pady = 15,padx = 150)

    tk.Label(frame2,text = '''•Shubham :: shubhamdutta5694@gmail.com:: 6202561126''',bg = 'gray', fg = "black",font = ('verdana',12)).pack(anchor = 'w',pady = 15,padx = 150)

    tk.Label(frame2,text = '''•Binay :: binaypurty22@gmail.com:: 6204998628''',bg = 'gray', fg = "black",font = ('verdana',12)).pack(anchor = 'w',pady = 15,padx = 150)

    frame3 = tk.Frame(about\_window,bg= 'gray')

    frame3.place(relx = 0.11,rely = 0.85, relwidth = 0.80, relheight = 0.13)

    button1 = tk.Button(frame3,text = 'Quit', bd=4, command = about\_window.destroy,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 10)

    button1.grid(pady=20,padx=20,row=0,column=1,sticky='e')

    button2 = tk.Button(frame3,text = 'Go to home page', bd=4, command = Go\_to\_home\_page,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 15)

    button2.grid(pady=20,padx=20,row=0,column=0,sticky='w')

    about\_window.mainloop()

###################################################### End of About Window ##########################################################

###################################################### Help Index Window ############################################################

def help\_index(window1, help\_index\_window):

    window1.destroy()

    def Go\_to\_home\_page():

        window1 = tk.Tk()

        window1.geometry("1280x720")

        window1.minsize(1080,720)

        window1.title("Employee Activity Tracker")

        window1.configure(background = 'gray')

        help\_index\_window.destroy()

        home\_page(window1)

    #################################################### Heading Frame  #################################

    frame1 = tk.Frame(help\_index\_window, bg = 'gray')

    frame1.place(relx = 0.11,rely = 0, relwidth = 0.80, relheight = 0.15)

    l1 = tk.Label(frame1,text = "Face Recognition Based Employee Activity Tracker",bg = 'gray', fg = "black",font = ('verdana',20,'bold'))

    l1.pack(anchor = 'center',pady = 4 )

    l2 = tk.Label(frame1,text =''' If you need any help feel free to contact us. Let us know about problem that you are facing.''',bg = 'gray', fg = "white",font = ('verdana',12))

    l2.pack(anchor = 'center',pady = 15 ,side = 'top')

    #################################################### Body Frame  ###########################################################################

    frame2 = tk.Frame(help\_index\_window,bg= 'black')

    frame2.place(relx = 0.11,rely = 0.151, relwidth = 0.80, relheight = 0.74)

    l1 = tk.Label(frame2,text = "Send To :",bg = "black", fg = "white",font = ('verdana',16,'bold'),)

    l1.grid(pady=10,padx=30,row=0,column=0,sticky='w')

    frame3 = tk.Frame(help\_index\_window,bg="black")

    frame3.place(relx = 0.37,rely = 0.17,relwidth = 0.45,relheight = 0.08)

    email\_id =  tk.Text(frame3,width = 100,height = 1,bd = 4, bg ="black",fg = "white",font = ('verdana',14,"bold"))

    email\_id.pack()

    email\_id\_label = tk.Label(frame3,text = "You can send email to more than one by seperating comma(,)",fg ="green",bg = "black",font = ('verdana',10))

    email\_id\_label.pack(anchor = "center")

    l2 = tk.Label(frame2,text = "Subject :",bg = "black", fg = "white",font = ('verdana',16,'bold'))

    l2.grid(pady=40,padx=30,row=1,column=0,sticky='w')

    frame4 = tk.Frame(help\_index\_window,bg="black")

    frame4.place(relx = 0.37,rely = 0.27,relwidth = 0.45,relheight = 0.08)

    Subject\_text = tk.Text(frame4,width = 100,height = 1,bd = 4, bg ="black",fg = "white",font = ('verdana',14,"bold"))

    Subject\_text.pack()

    Subject\_text\_label = tk.Label(frame4,text = "",fg ="green",bg = "black",font = ('verdana',10))

    Subject\_text\_label.pack(anchor = "center")

    l3 = tk.Label(frame2,text = "Compose Message :",bg = "black", fg = "white",font = ('verdana',16,'bold'))

    l3.grid(pady=30,padx=30,row=2,column=0,sticky='w')

    frame5 = tk.Frame(help\_index\_window,bg="black")

    frame5.place(relx = 0.37,rely = 0.38,relwidth = 0.45,relheight = 0.32)

    email\_body = tk.Text(frame5,width = 100,height = 25,bd = 6, bg ="black",fg = "white",font = ('verdana',10))

    email\_body.pack()

    tk.Label(frame2,text = "",bg = "black", fg = "white",font = ('verdana',16,'bold')).grid(pady=50,padx=30,row=3,column=0,sticky='w')

    tk.Label(frame2,text = "",bg = "black", fg = "white",font = ('verdana',16,'bold')).grid(pady=20,padx=30,row=4,column=0,sticky='w')

    def send\_mail():

        id\_to\_send = email\_id.get(1.0, "end")

        email\_id\_to\_send\_mail = id\_to\_send.split(",")

        sub = Subject\_text.get(1.0,"end")

        body = email\_body.get(1.0, "end")

        for id in email\_id\_to\_send\_mail:

            if len(id) < 10 :

                email\_id\_label.configure(text = "Please enter correct Email id !!!!!!!!!!",fg = "red")

                break

            else:

                email\_id\_label.configure(text = "",fg = "red")

        if len(sub) < 2 :

            Subject\_text\_label.configure(text = "Please fill subject field !!!!!!!!!!! ",fg="red")

        else:

            Subject\_text\_label.configure(text = "")

        if len(body) < 2 :

            body\_label.configure(text = "Please fill all field !!!!!!!!!!! ")

        else:

            body\_label.configure(text = "")

        try:

           with open("email\_details.txt","r") as f:

            id\_and\_pwd\_of\_emp= f.readlines()

            emp\_id = id\_and\_pwd\_of\_emp[0].replace("\n","")

            emp\_pwd = id\_and\_pwd\_of\_emp[1]

            ob = smtp.SMTP("smtp.gmail.com",587)

            ob.starttls()#this initiat tls encryption

            ob.login(emp\_id,emp\_pwd)

            message = "Subject :{}\n\n{}".format(sub,body)

            ob.sendmail(id,[email\_id\_to\_send\_mail],message)

            messagebox.showinfo("Done","Mail send successfully....")

            ob.quit()

            email\_id.delete(1.0,"end")

            Subject\_text.delete(1.0,"end")

            email\_body.delete(1.0,"end")

        except Exception as e:

            messagebox.showinfo("Error in connection",e)

    send\_button = tk.Button(frame2,text = "SEND",bg = "black",command = send\_mail, fg = "white",font = ('verdana',16,'bold'),width= 8)

    send\_button.grid(pady=70,padx=20,row=5,column=0,sticky='w')

    body\_label = tk.Label(frame2,text = "",bg = "black", fg = "red",font = ('verdana',10))

    body\_label.grid(pady=30,padx=30,row=5,column=1,sticky='e')

    ######################################### Footer Frame ##################################################################################

    frame3 = tk.Frame(help\_index\_window,bg= 'gray')

    frame3.place(relx = 0.11,rely = 0.895, relwidth = 0.80, relheight = 0.09)

    button1 = tk.Button(frame3,text = 'Quit', bd=4, command = help\_index\_window.destroy,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 10)

    button1.grid(pady=20,padx=20,row=0,column=1,sticky='w')

    button2 = tk.Button(frame3,text = 'Go to home page', bd=4, command = Go\_to\_home\_page,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 15)

    button2.grid(pady=20,padx=20,row=0,column=0,sticky='w')

    help\_index\_window.mainloop()

###################################################### End of Help Window ###########################################################

##################################################### Home page window ##############################################################

def home\_page(window1 = window1):

    ################################ Framing Of Window ##############################

    frame1 = tk.Frame(window1, bg = 'gray')

    frame1.place(relx = 0.11,rely = 0, relwidth = 0.80, relheight = 0.118)

    frame2 = tk.Frame(window1,bg= 'gray')

    frame2.place(relx = 0.11,rely = 0.12, relwidth = 0.80, relheight = 0.118)

    frame3 = tk.Frame(window1, bg = 'gray')

    frame3.place(relx = 0.11,rely = 0.24, relwidth = 0.80, relheight = 0.643)

    frame4 = tk.Frame(window1, bg = 'gray')

    frame4.place(relx = 0.11,rely = 0.88, relwidth = 0.80, relheight = 0.1)

    ################################ Loading Register Image and Performing operation for start\_face\_reading function ################

    def Register\_page():

        Registration\_page = Registration()

        Registration\_page.register\_new\_employee(window1)

    path = 'images'

    images = []

    mylist = os.listdir(path)

    def find\_Encoding(images):

        encodedList = []

        for img in images:

            img = cv2.resize(img,(0,0),None,0.25,0.25)

            img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

            encode = face\_recognition.face\_encodings(img)[0]

            encodedList.append(encode)

        return encodedList

    if (len(mylist)>0) and os.path.isdir(path):

        registered\_img = cv2.imread("{}\\{}".format(path,mylist[0]))

        images.append(registered\_img)

        encode\_of\_registered\_img = find\_Encoding(images)

        with open('Details.txt','r') as f :

            detail = f.readlines()

            id = detail[0].replace('\n','')

            name = detail[1].replace('\n','')

            duration = detail[2]+ ":00"

    else:

        Register\_page()

    ############################################### Showing details of Employee ##################################################

    tree = ttk.Treeview(frame3,height = 25)

    style = ttk.Style(tree)

    style.configure(".",font = ('Helvetica',10))

    style.theme\_use("clam")

    style.configure("Treeview.Heading",foreground = 'red',font = ('Helvetica',10,"bold"))

    style.configure("Treeview",rowheight=25,fieldbackground="silver")

    style.map('Treeview',background=[('selected',"red")])

    tree.tag\_configure('evenrow',background = "lightblue")

    tree.tag\_configure('oddrow',background = "white")

    tree['show'] = 'headings'

    tree["columns"] = ("one","two","three","four","five")

    tree.column("one",width = 125,anchor = 'center')

    tree.column("two",width = 250,anchor = 'center')

    tree.column("three",width = 150,anchor = 'center')

    tree.column("four",width = 100,anchor='center')

    tree.column("five",width = 150,anchor='center')

    tree.heading("one", text='ID')

    tree.heading("two", text='Name')

    tree.heading("three",text='Working Hours')

    tree.heading("four",text='Login time')

    tree.heading("five",text='Working Hours Left')

    tree.pack()

    login\_time = (datetime.now()).strftime('%H:%M:%S')

    def insert\_details\_to\_home\_page():

        tree.insert('','end',text = "0", value = (id,name,duration,login\_time,duration),tags = ('evenrow',))

    class set\_working\_hour\_left():

        def \_\_init\_\_(self):

            self.work\_hour\_left=""

        def set\_working\_hour(self,whl):

            self.work\_hour\_left = whl

        def get\_whl(self):

            return self.work\_hour\_left

    obj = set\_working\_hour\_left()

    class tracking\_working\_time():

        def \_\_init\_\_(self):

            self.date = (datetime.now()).strftime("20%y-%m-%d")

            self.id = id

            self.name = name

            self.duration = duration

            self.working\_hour\_left = obj.get\_whl()

            self.i = 1

        def get\_data(self):

            lst = [self.date,self.id,self.name,self.duration,login\_time,self.working\_hour\_left]

            return lst

        def tracking\_working\_hour(self):

            self.working\_hour\_left = obj.get\_whl()

            self.working\_hour\_left = str(self.working\_hour\_left)

            if self.i % 2 ==0 :

                tree.insert('','end', value = (self.id,self.name,self.duration,login\_time,self.working\_hour\_left),tags = ('evenrow',))

            else:

                tree.insert('','end', value = (self.id,self.name,self.duration,login\_time,self.working\_hour\_left),tags = ('oddrow',))

            self.i +=1

    class Data\_insert():

        def insert\_details(self):

            lst = calculate\_working\_time.get\_data()

            if lst[5] == "":

                return 0

            with open('Data.csv','a') as f1:

                data = ("\n{},{},{},{},{},{}").format(lst[0],lst[1],lst[2],lst[3],lst[4],lst[5])

                f1.writelines(data)

    ################################ Heading Of Window ##############################

    l1 = tk.Label(master= frame1,text = "Face Recognition Based Employee Activity Tracker",bg = 'gray', fg = "black",font = ('verdana',20,'bold'))

    l1.pack(anchor = 'center',pady = 4 )

    l2 = tk.Label(master=frame1,text="",font = ('verdana',16,'bold'),bg = "gray", fg = "black")

    l2.pack(anchor = 'e',side = 'right',pady = 10)

    ########################### Operation on loaded register image completed #################################

    calculate\_working\_time = tracking\_working\_time()

    def start\_face\_reading(already\_login=True,name=name,calculate\_working\_time=calculate\_working\_time):

        count = 0

        cap = cv2.VideoCapture(0)

        while True:

            success, img = cap.read()

            if success:

                imgs = cv2.resize(img,(0,0),None,0.25,0.25)

                imgs = cv2.cvtColor(imgs,cv2.COLOR\_BGR2RGB)

                facesInFrame = face\_recognition.face\_locations(imgs)

                encodefacesInFrame = face\_recognition.face\_encodings(imgs,facesInFrame)

                for encodeface, faceloc\_Cap in zip(encodefacesInFrame,facesInFrame):

                    matches = face\_recognition.compare\_faces(encode\_of\_registered\_img,encodeface)

                    facedis = face\_recognition.face\_distance(encode\_of\_registered\_img,encodeface)

                    matchIndex = np.argmin(facedis)

                    if matches[matchIndex]:

                        y1, x2, y2, x1 = faceloc\_Cap

                        y1, x2, y2, x1 = y1\*4,x2\*4,y2\*4,x1\*4

                        cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)

                        #cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),2,cv2.FILLED)

                        cv2.putText(img,name,(x1,y2+30),cv2.FONT\_HERSHEY\_COMPLEX,1,(255,255,255),2)

                        count +=1

                        if count > 3 and already\_login:

                            cap.release()

                            cv2.destroyAllWindows()

                            insert\_details\_to\_home\_page()

                            timer1 = timer()

                            timer1.countdown()

                            count = 0

                            return 0

                        elif count > 3:

                            cap.release()

                            cv2.destroyAllWindows()

                            calculate\_working\_time.tracking\_working\_hour()

                            count = 0

                            return 0

                cv2.imshow("Detecting Face ",img)

                cv2.waitKey(1)

            else:

                 messagebox.showinfo("Camera Opening Error","Can't able open webCam... ")

    def exit():

        obj = Data\_insert()

        obj.insert\_details()

        window1.destroy()

    ################################## TIMER #######################################

    class timer():

        def \_\_init\_\_(self,duration = duration):

            self.t = duration.split(":")[0]

            self.t = int(self.t)\*60\*60

            self.count = 0

            self.rand\_time = random.randint(10,15)

        def countdown(self):

            mins, sec = divmod(self.t,60)

            hour, mins = divmod(mins,60)

            timer = '{:02d}:{:02d}:{:02d}'.format(hour,mins,sec)

            obj.set\_working\_hour(timer)

            l2.config(text = timer)

            l2.after(1000,self.countdown)

            self.t -= 1

            if self.count >= self.rand\_time:

                start\_face\_reading(False)

                self.count = 0

            self.count += 1

            if self.t <= 0 :

                exit()

    def go\_to\_help\_index():

        help\_index\_window = tk.Tk()

        help\_index\_window.geometry("1280x780")

        help\_index\_window.minsize(1080,720)

        help\_index\_window.title("Employee Activity Tracker")

        help\_index\_window.iconbitmap("icon.ico")

        help\_index\_window.configure(background = 'gray')

        help\_index(window1 , help\_index\_window)

    def go\_to\_about():

        about\_window = tk.Tk()

        about\_window.geometry("1280x720")

        about\_window.minsize(1080,720)

        about\_window.title("Employee Activity Tracker")

        about\_window.iconbitmap("icon.ico")

        about\_window.configure(background = 'gray')

        About(window1 , about\_window)

    def Delete\_employee():

        msg = messagebox.askquestion("Comfirm","Are you sure !!!!\nYou want to Delete Employee details")

        if msg == "yes":

            path = "images"

            image = os.listdir(path)

            for img in image:

                path = path + "\\" + img

                if os.path.isfile(path):

                    os.remove(path)

                else:

                    messagebox.showinfo("Invalid","Emyployee not registered")

            if os.path.isfile("Details.txt"):

                with open("Details.txt","w") as f:

                    f.write("")

                messagebox.showinfo("Closing ","Restart the application")

                window1.destroy()

            else:

                messagebox.showinfo("Invalid","Emyployee not registered")

    class Reset\_Working\_hour():

        def \_\_init\_\_(self):

            self.reset\_duration = tk.Tk()

            self.reset\_duration.geometry("360x180")

            self.reset\_duration.resizable('false','false')

            self.reset\_duration.title("Reset Working Duration")

            self.reset\_duration.iconbitmap("icon.ico")

            self.reset\_duration.configure(background = 'black')

            l1 = tk.Label(self.reset\_duration,text = "Reset Duration : ",bg = "black", fg = "white",font = ('Helvetica',12,'bold'))

            l1.grid(pady=20,padx=20,row=0,column=0,sticky='w')

            option = ["01:00","02:00","03:00","04:00","05:00","06:00","07:00","08:00","09:00","10:00","11:00","12:00"]

            self.var = tk.StringVar()

            self.var.set("HH:MM")

            self.drop2 = tk.OptionMenu(self.reset\_duration,self.var,\*option)

            self.drop2.grid(pady=20,padx=20,row=0,column=1,sticky='e')

            self.drop2.configure(bg="gray",width = 10)

            self.reset\_button = tk.Button(self.reset\_duration,text = "Reset",command = self.reset  ,font = ('Helvetica',12,'bold'),bg = "gray",fg="white",bd = 4,width = 10)

            self.reset\_button.grid(pady=20,padx=20,row=1,column=0,sticky='w')

            self.Quit = tk.Button(self.reset\_duration,text = "Quit", command = self.exit,font = ('Helvetica',12,'bold'),bg = "gray",fg="white",bd = 4,width = 10)

            self.Quit.grid(pady=20,padx=20,row=1,column=1,sticky='e')

            self.l2 = tk.Label(self.reset\_duration,text = "",bg = "black", fg = "red",font = ('Helvetica',12,'bold'))

            self.l2.grid(pady=0,padx=20,row=2,column=0,sticky='w',columnspan = 2)

            self.reset\_duration.mainloop()

        def exit(self):

            self.reset\_duration.destroy()

        def reset(self):

            if os.path.isfile("Details.txt"):

                with open("Details.txt",'r') as f:

                    data = f.readlines()

                if (len(data) < 1):

                    self.l2.configure(text ="Please Register First!!!!!!")

                elif (self.var.get() == "HH:MM"):

                    self.l2.configure(text ="Please select the duration!!!!!!")

                else:

                    with open("Details.txt",'w') as f:

                        data[2] = self.var.get()

                        f.writelines(data)

                        messagebox.showinfo("Done","Reset Duartion Successfully.....")

                        self.reset\_duration.destroy()

            else:

                self.l2.configure(text ="Please Register First!!!!!!")

    def reset\_hour():

        obj = Reset\_Working\_hour()

    ################################# MenuBar ########################################

    menubar = Menu(window1)

    filemenu = Menu(menubar, tearoff = 0)

    filemenu.add\_command(label = 'Register New Employee', command = Register\_page,font = ('verdana',8,'bold'))

    filemenu.add\_command(label = 'Reset Working Duration', command = reset\_hour,font = ('verdana',8,'bold'))

    filemenu.add\_command(label = 'Delete Employee Details', command = Delete\_employee,font = ('verdana',8,'bold'))

    filemenu.add\_command(label = 'Exit', command = exit,font = ('verdana',8,'bold'))

    menubar.add\_cascade(label = 'Edit', menu = filemenu)

    helpmenu = Menu(menubar, tearoff = 0)

    helpmenu.add\_command(label = 'Help index', command = go\_to\_help\_index, font = ('verdana',8,'bold'))

    helpmenu.add\_command(label = 'About...', command = go\_to\_about, font = ('verdana',8,'bold'))

    menubar.add\_cascade(label = 'Help',menu = helpmenu)

    window1.config(menu = menubar )

    ################################## Buttons of home page ############################

    button1 = tk.Button(frame2,text = 'Start Your Work', bd=4, command = start\_face\_reading ,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 2, width = 35)

    button1.pack(anchor = 'center',pady = 20)

    button2 = tk.Button(frame4,text = 'Quit', bd=4, command = exit,font = ('verdana',12,'bold'),bg = '#262523',fg = 'white',height = 1, width = 10)

    button2.pack(anchor = 'w',side = 'bottom',pady = 10, padx = 50 )

    window1.iconbitmap("icon.ico")

##################################################### End of Home page window ##############################################################

home\_page()

window1. mainloop()