Major Class,Functions and their uses :

- **Take_Picture :** In this function, first of all we check that whether any employee is registered or not. If yes , we will not allow any employee to register further, as our project is limited to single user. If not , We will allow the New Employee to capture his/her photo. If web cam couldn't open for any reason , then an exception for the same is raised.

  Employee can take picture by pressing 's' or 'S' and quit the camera window using 'q' or 'Q' keys.

```
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') and success:
            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
```

- **Save_Profile :** In this function , the Employee enters his details which includes ‘Employee ID’ , ‘Employee Name’ , ‘Working Duration’. He may also browse his image from local files. Employee can either take picture or browse picture. If Employee has already browsed his picture then , he won’t be able to take picture and vice versa.

  If Employee wants to clear the taken picture and he wants to browse his picture from the local files , He may click on browse button and browse his picture. Then the already taken picture is cleared and the browsed picture is saved.

  Finally the system gives a prompt , whether the user wants to save his profile or not. If clicked YES the profile gets saved. He may choose NO if he wants to edit his details further.

```
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
```

- **Start_face_reading :** In this function , we take three default argument like already login, Name and calculate working time. With the help of these three parameters we validate Employee is already registered or not. It also validates whether the user in front of the system is the registered employee or not using HOG algorithm. It compared registered image with the camera detected image. If there is a match then employee is logged in to start his work and the login details are shown on the Home Page window in the form of table.

  Anyhow , if camera isn't opened it will prompt a message 'Web cam unable to open' . All the rest of the functions are invoked from here.

```
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: int 2, 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... ")
```

- **Timer Class :** In this class a timer function is defined , which countdowns the working hours on start of work. It also invokes Start_face_reading function after any random interval of time. If the working time is over , all the information of his work will be stored in **Data.csv** and closes the application with the help of Exit function.

```
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()
```

- **Class Tracking_Working_time :** In this Class , We have defined init constructor to get details about the employee and his login time. This class has **tracking_working_hour** function which inserts details and left working hours into the table. This also has a get_data function which returns details of the employee like ID, Name, Login Time, duration , Working time left and all to class data_insert.

```
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)
```

- **Insert_Details :** This class have function called **Insert_details** which inserts data into the file called **Data.csv** which is received from get_data function of class **tracking_working_time.**

```
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)
```

- **Reset :** In this function , it checks for User data availability in **Details.txt.** If not found it tells the user to register first because non-availability of data means , No user is registered yet. If Data is found in **Details.txt** , it resets the working duration.

```
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!!!!!!")
```

- **Delete_Employee :** This function gives a prompt to the user whether he wants to delete his Employee details or not . If YES then check whether the input is of a valid employee or not . If VALID , the details of the user is deleted and promt a message to restart the application . If INVALID , then a prompt of ‘Employee not regisgtered’ is displayed.

```
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")
```

- **Send_Email :** This functions is used to send mail. This function takes details like Email Id to send , Subject and Body of the mail. If any of the three isn't given or provided in correct format , then it will show prompt accordingly. Else , it will read the Sender mail credentials from **email_details.txt** file and use SMTP protocols to send the mail. This function also initiates TLS encryption send the mail content in abstract form. If mail is sent correctly it will show 'Mail sent successfully'. While sending mail , there maybe connectivity issues , or incorrect credential issues which may rise exception.

```
def send_mail():
    id_to_send = email_id.get(1.0, "end")
    sub = Subject_text.get(1.0,"end")
    body = email_body.get(1.0, "end")
    if len(id_to_send) < 10 :
        email_id_label.configure(text = "Please enter correct Email id !!!!!!!!!!",fg = "red")
    else:
        email_id_label.configure(text = "",fg = "red")
    if len(sub) < 10 :
        Subject_text_label.configure(text = "Please fill subject field !!!!!!!!!! ",fg="red")
    else:
        Subject_text_label.configure(text = "")
    if len(body) < 10 :
        body_label.configure(text = "Please fill all field correctly !!!!!!!!!! ")
    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(emp_id,[id_to_send],message)
        body_label.configure(text = "Mail send successfully....",fg = 'green')
        ob.quit()
        email_id.delete(1.0,"end")
        Subject_text.delete(1.0,"end")
        email_body.delete(1.0,"end")
    except Exception as e:
        error = str(e)
        body_label.configure(text = error,fg = 'red')
```