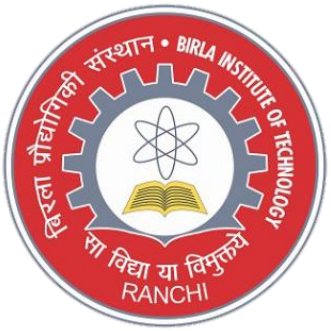


BIRLA INSTITUTE OF TECHNOLOGY
MESRA, RANCHI-835215
(Deemed University)
EXTENSION CENTER LALPUR, RANCHI.

Final Year Project

On Activity Tracker
Using Python

Batch : 1

Documentation And Report

BIRLA INSTITUTE OF TECHNOLOGY

MESRA, RANCHI-835215
(Deemed University)
EXTENSION CENTER LALPUR, RANCHI.

Certificate

This is to certify that the contents of this project entitled “Activity Tracker using Machine Learning in Python” is a bonafide work carried out by Prince,Tirthoraj , Shubham, Binay in fulfillment of the requirement for the award of Bachelor of Computer Applications under my guidance.

This is to further certify that the matter of the project has not been submitted by anyone else for the award of any degree.

Mrs. Aparna Shukla
Dept. of Computer science.
Birla Institute of Technology.

BIRLA INSTITUTE OF TECHNOLOGY

MESRA, RANCHI-835215
(Deemed University)
EXTENSION CENTER LALPUR, RANCHI.

Certificate of Approval

The foregoing project is hereby approved as a creditable work on "Activity Tracker using Machine Learning in Python", carried out and presented in the manner satisfactory to warrant its acceptance as prerequisite to the degree for which it has been entitled.

It is understood that by this approval the undersigned do not endorse any statement made or opinion expressed but approve work for the purpose for which it is submitted.

Committee for evaluation of the project

External Examiner	**Internal Examiner**
Date:	**Date:**

In Charge
BIRLA INSTITUTE OF TECHNOLOGY
Extension center Lalpur, Ranchi

Acknowledgement

We would like to express our special thanks of gratitude to my teacher Mrs. Aparna Shukla Ma'am who gave us the golden opportunity to do this wonderful project on the topic "Activity Tracker Using Machine Learning in Python," which also helped me in doing a lot of Research and hence we came to know about so many new things. We are really thankful to her.
Secondly, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Above all, we would like to thank Almighty God for showering his blessings on us which enlightened our path and helped us in successful completion of our project.

Index

Group Members

Name	Roll No.	Email Id	Phone No.
Binay Asim Purty	**BCA/40579/18**	binaypurty231@gmail.com	6204998628
Prince Kumar	**BCA/40579/18**	pk03215@gmail.com	9504008839
Tirthoraj Dasgupta	**BCA/40579/18**	tirthorajdasgupta@gmail.com	8434116014
Shubham Kr. Dutta	**BCA/40579/18**	shubhamdutta5694@gmail.com	6202561126

What our project is all about.

- Our Project is a Real time **Activity Tracker software**.
- The aim of the project is to keep a track of the activity of the user.
- It monitors whether the user who has logged In , is present in his position.
- It also has a **Countdown timer** for the user which works to show the left working hours.
- The software checks for the user availability infront of the monitor , continuously after a random interval of time.
- If **Presence** of the user is recognized, The software notes the time and the **countdown timer is carried forward**.
- If **Absence** of the user is recognized, The software **stops the countdown timer** until user is recognized.
- It maintains the record, whether the user is present for the allotted hours or not.
- We have used Python to build this project.

Essential Modules

- **OS module :** The OS module in Python provides functions for interacting with the operating system. In our project, we use this module to check whether path, directory, files, or images available or not.
- **Time module :** The Python time module provides many ways of representing time in code, such as objects, numbers, and strings.
- **Tkinter module :** Tkinter is the standard GUI library for Python There are various widgets like button, optionMenu, entry, checkbutton, etc. that are used to build the python GUI applications.
- **Datetime module :** Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals.
- **String module :** Python string module provides additional tools to manipulate strings. In our project, we use this module to convert integer and time value into string format.
- **OpneCV module :** OpenCV or cv2 module is a huge open-source library for computer vision, machine learning, and image processing. It can process images and videos to identify objects, faces, or even the handwriting of a human. We use this module to recognize employee face.
- **Face Recognition module :** This module is used recognize and manipulate faces. It automatically find all the faces in an image, locate the facial features of a person in an image and recognize faces in images and identify who they are.
- **Numpy module :** It stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.

  Using NumPy, mathematical and logical operations on arrays can be performed. We use this module in our to find maximum matching distance of faces in frame.

- **Shutil Module** : Shutil module in Python provides many functions of high-level operations on files and collections of files. We use this module in our project to copy and remove the image of employee from directory.
- **Smtplib Module** : smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an Simple Mail Transfer Protocol (SMTP). We use this, to send mail if employee is face any problem this software.
- **Random Module** : Python has a built-in random module that you can use to make random numbers, shuffle the list, etc. We use this module to generate a random integer between two integer.

Algorithm

HOG Algorithm

A HOG is a feature descriptor generally used for object detection. HOGs are widely known for their use in pedestrian detection. A HOG relies on the property of objects within an image to possess the distribution of intensity gradients or edge directions. Gradients are calculated within an image per block. A block is considered as a pixel grid in which gradients are constituted from the magnitude and direction of change in the intensities of the pixel within the block.

What is a Histogram of Oriented Gradients (HOG)?

A HOG is a feature descriptor generally used for object detection. HOGs are widely known for their use in pedestrian detection. A HOG relies on the property of objects within an image to possess the distribution of intensity gradients or edge directions. Gradients are calculated within an image per block. A block is considered as a pixel grid in which gradients are constituted from the magnitude and direction of change in the intensities of the pixel within the block.

The descriptors are gradient vectors generated per pixel of the image. The gradient for each pixel consists of magnitude and direction, calculated using the following formulae:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

Real World Example Application

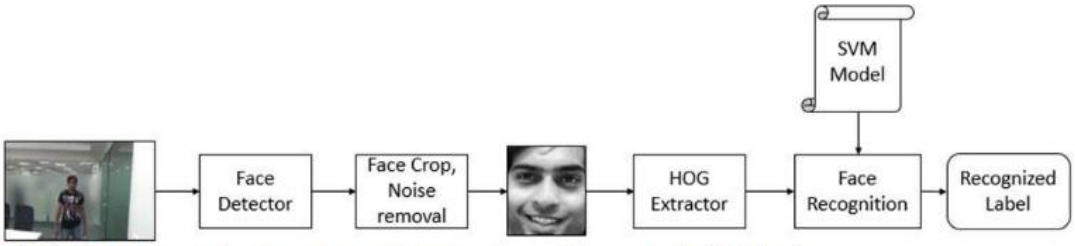
A project of **eInfochips :**

This real-world example application involved access management and analytics using face recognition in a video management system. In this project, a team at eInfochips (an Arrow Company) designed and implemented a video management system with face recognition to detect and recognize faces in the feeds of multiple IP cameras. This setup was deployed in a manufacturing industry, wherein cameras were installed at various locations in multiple buildings and were interconnected using a Local Area Network (LAN).

The team developed and installed a complete video management solution into this network, which empowered the security department with video streaming, alerts through video analytics, and access authentications. The face recognition service within the Video Management System was used to recognize the faces in real-time from camera feeds and generate system events to trigger the authentication process for employees and visitors to the premises. Additionally, these events consisted of details such as time of recognition, the name of the person, the location of the person on the map, etc. These events from the database were then analyzed using user interface clients within the video management system.

HOG Algorithm For Face Recognition

- The recognition of a face in a video sequence is split into three primary tasks: Face Detection, Face Prediction, and Face Tracking.
- The tasks performed in the Face Capture program are performed during face recognition as well. To recognize the face obtained, a vector of HOG features of the face is extracted.
- This vector is then used in the SVM model to determine a matching score for the input vector with each of the labels. The SVM returns the label with the maximum score, which represents the confidence to the closest match within the trained face data.
- The task of calculating matching scores is exceptionally heavy to compute. Hence, once detected and identified, the labeled face in an image needs to be tracked to reduce the computation in future frames until the face eventually disappears from the video. Of all the available trackers, the Camshift tracking algorithm is used since it produces the best results with faces.



Block diagram of the face recognition process (Source: eInfochips)

Software Used

1.Visual Studio Code

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including **Java, JavaScript, Go, Node.js, Python and C++.**

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

2.Python

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's developers strive to avoid premature optimization, and reject patches to non critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions, list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Benefits of Python

- Presence of Third-Party Modules
- Extensive Support Libraries
- Open Source and Community Development
- Learning Ease and Support Available
- User-friendly Data Structures
- Productivity and Speed
- Highly Extensible and Easily Readable Language

Hardware Used

- **Laptop/ PC**

- **External Web Cam / Web Cam**

Advantages

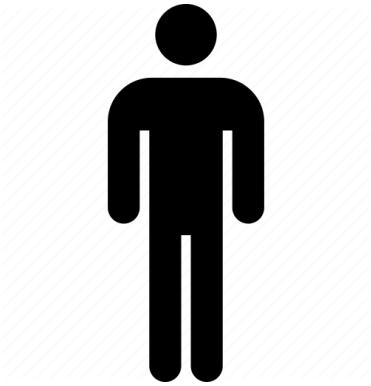
- The User cannot trick the organisation .

- The work efficiancy is increased. And Hence organization develops more.

- Potential of an individual is correctly determined.

Glitches

- It is restricted to single user functioning.



- During its run , As it uses the camera the camera may not open for some Other applications.

What is an Use-Case Diagram

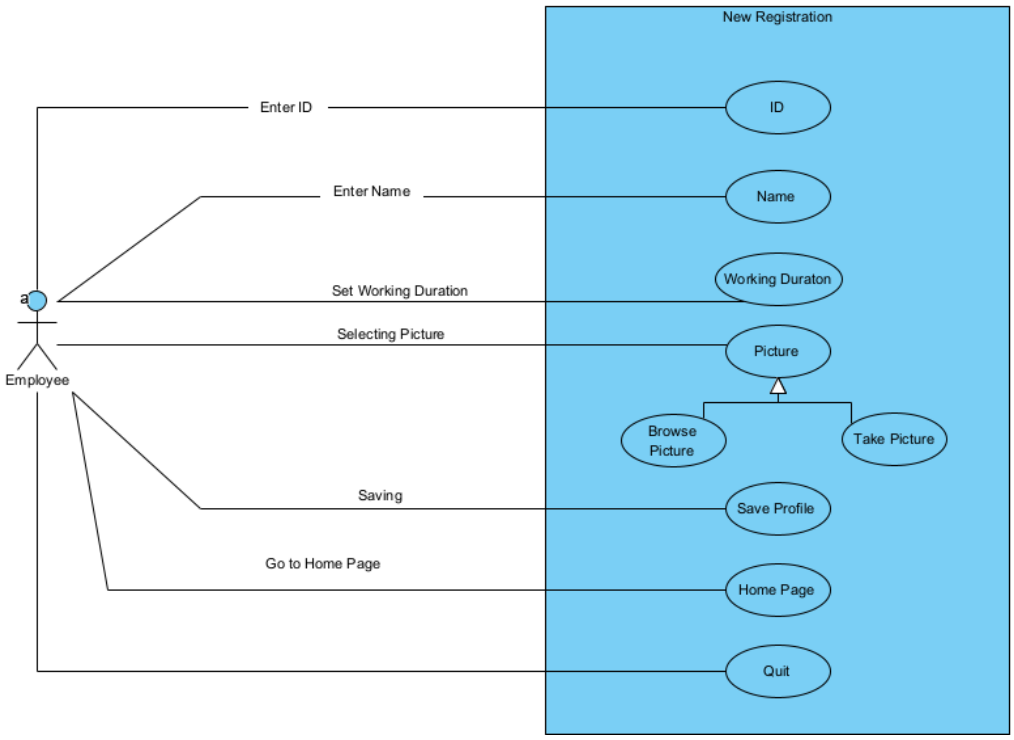
A **UML** use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.
A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes **some of the relationships** between use cases, actors, and systems.
- It does **not show the order** in which steps are performed to achieve the goals of each use case.

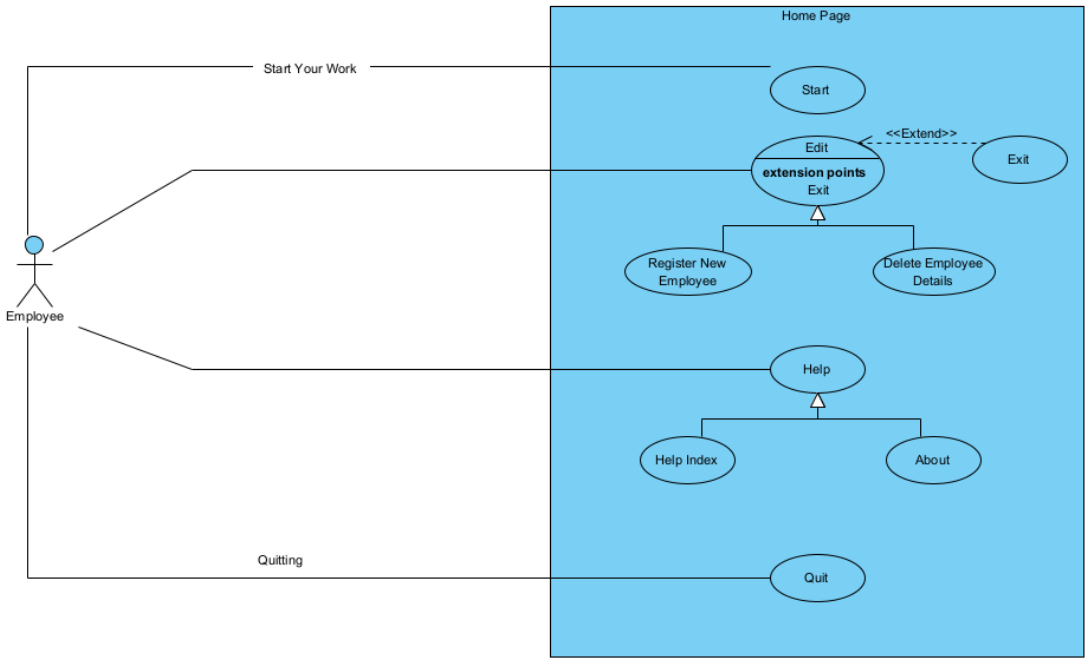
Use Case Diagram for our project

- Registration Window
- Home Page
- Help index
- About Page

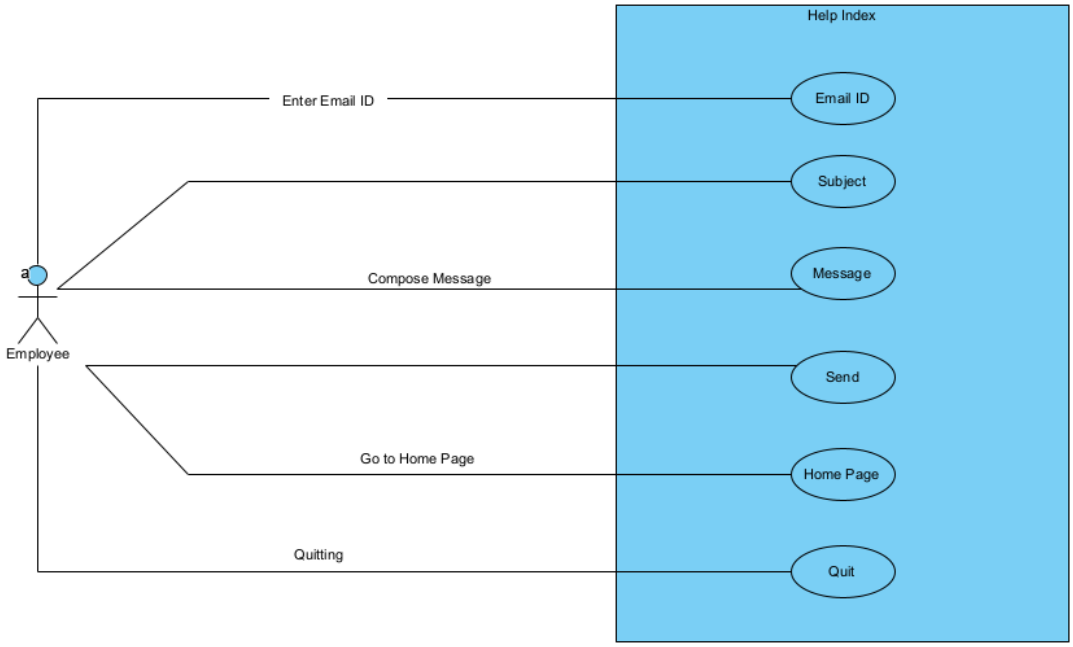
Registration Window Use-Case Diagram



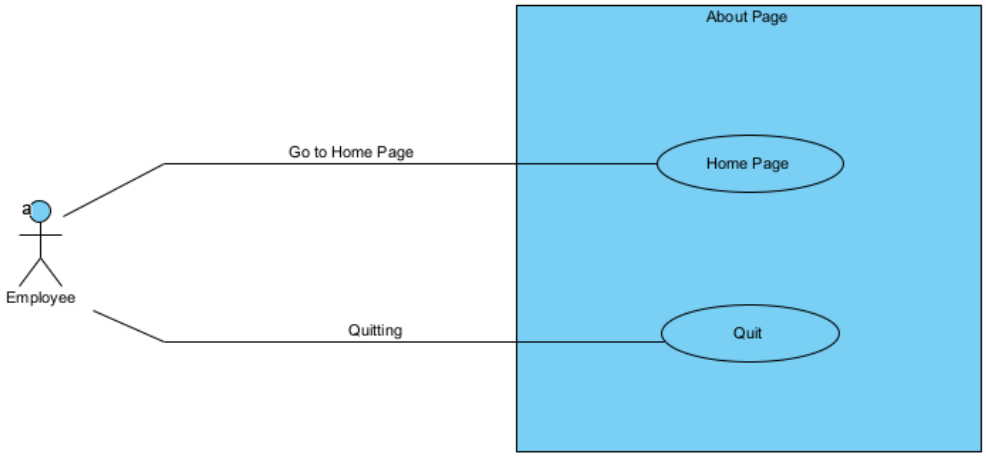
Home Page Use-Case Diagram



Help Index Use-Case Diagram



About Page Use-Case Diagram



What Is An ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

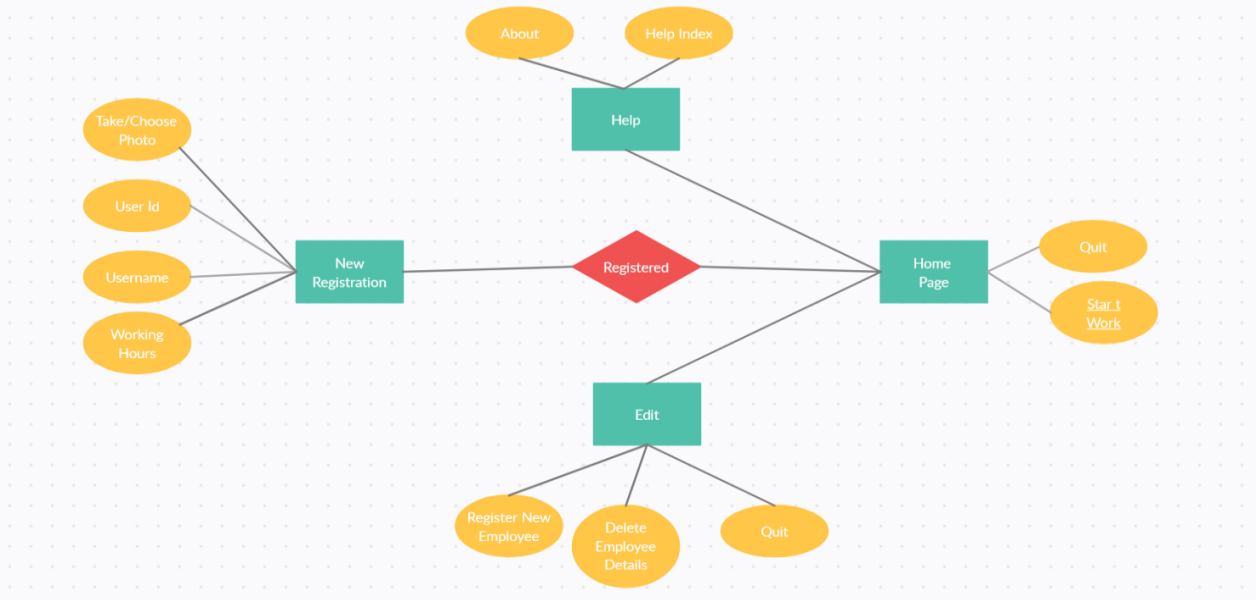
ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Why use ER Diagrams?

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users.

ER Diagram For Activity Tracker Software



What is a data flow diagram?

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. Often it is a preliminary step used to create an overview of the system that can later be elaborated.

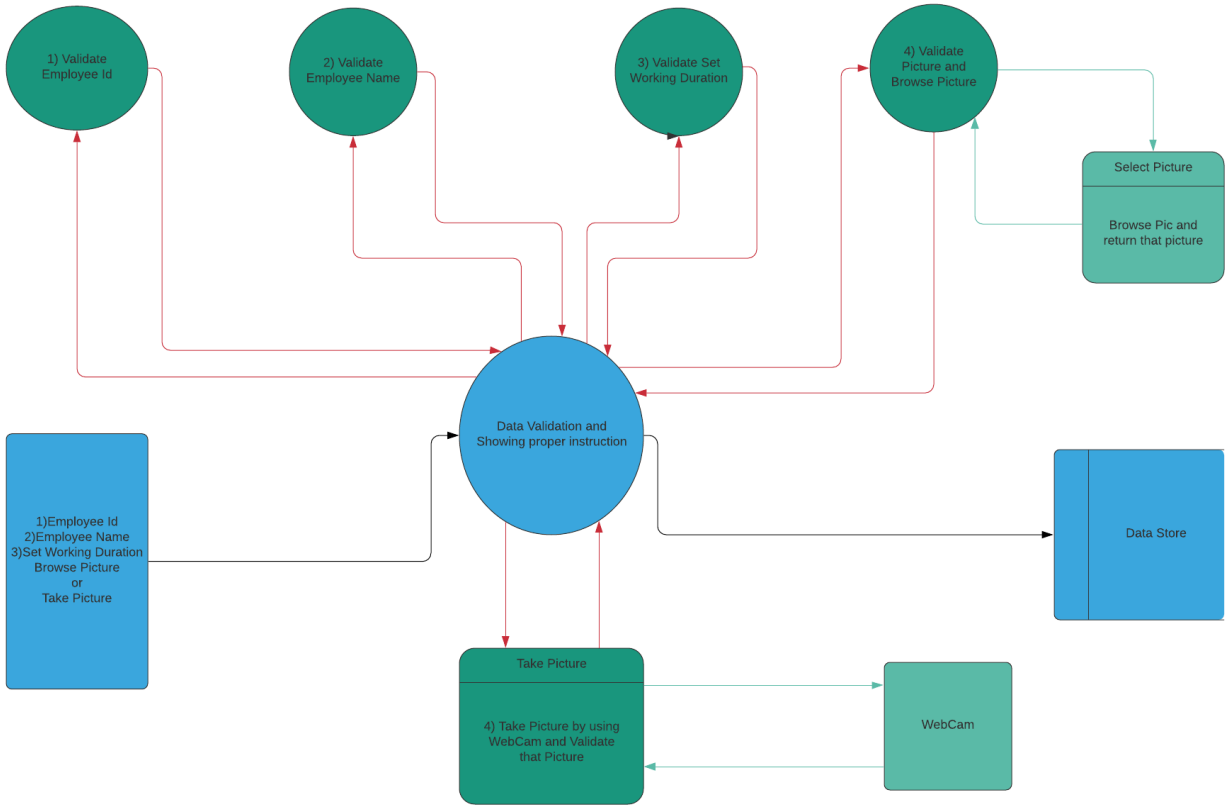
Data flow diagram for Our Project?

- Registration Page :The New User Registration page where the User inputs his requisites and also takes his picture for face-recognition.

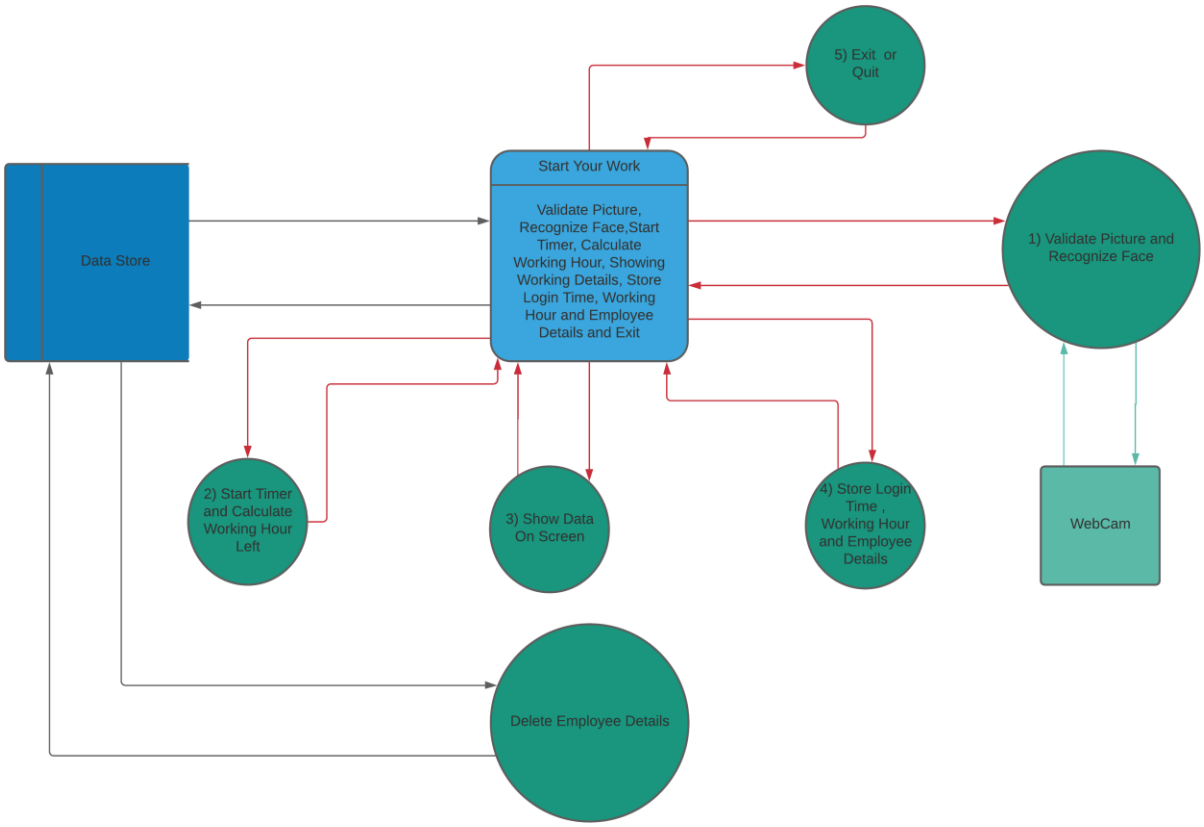
- Home Page : The page where from where the User can start his/her work and also his records are displayed over there. It's the page from where the User can be redirected to any page.

- Help Index : Help Index is a page where the user can connect to the software controller department for any problem occurred.

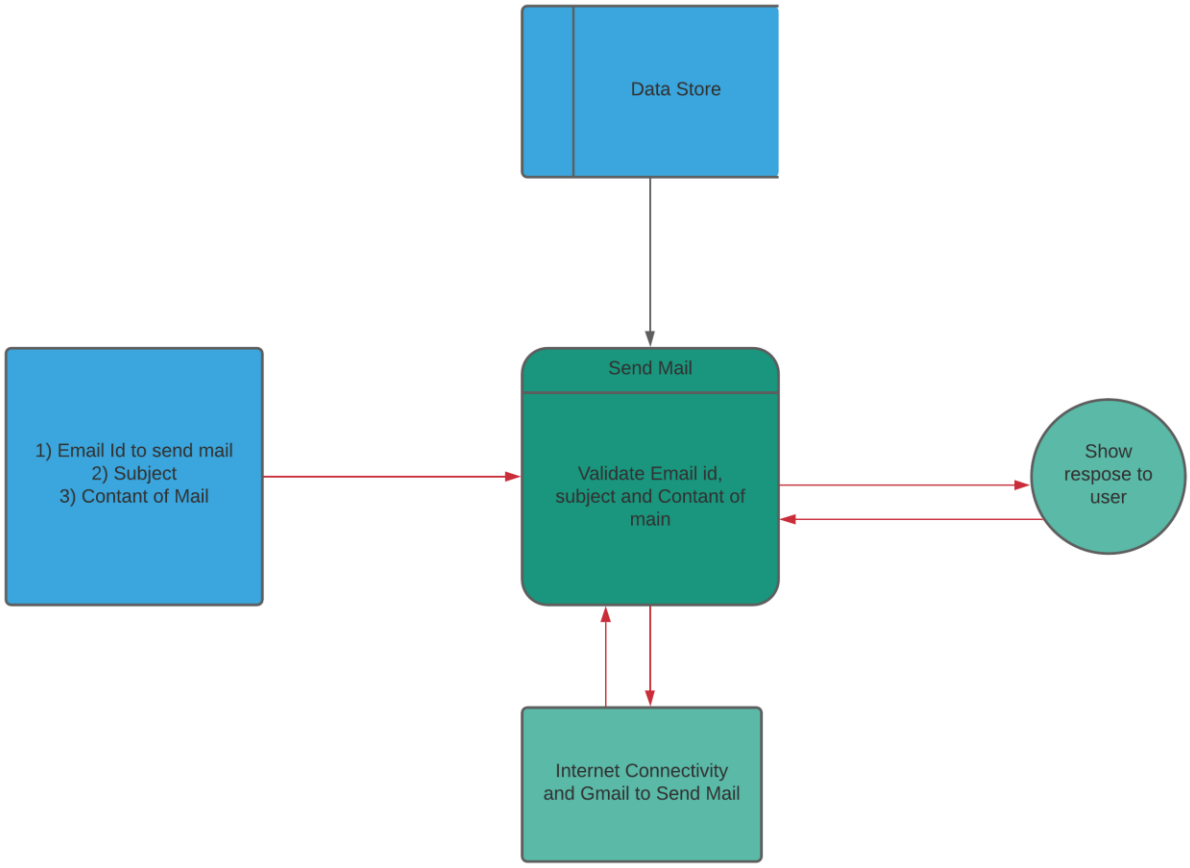
Registration Page Data Flow Diagram



Home Page Data Flow Diagram



Help Index Data Flow Diagram



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')

########################################################## Regi
stration 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 alr
eady 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 Yo
ur 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_B
GR2RGB)
                cv2.imshow(' Showing Clicked Picture ',self.
img_new)
                cv2.waitKey(2000)
                cap.release()
                cv2.destroyAllWindows()
                self.button1_label.config(text = "......Pict
ure Taken....",fg = 'green')
                if self.browse_pic_label:
                    self.browse_pic_label = False
                    self.var1.set("")
                    self.entry3_label.config(text = "* brows
ed 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 E
mployee ID!!!!!!")
        else:
            self.entry1_label.config(text = "")
        ################################### Validating NAME
###################
        NAME = self.entry2.get()
        if len(NAME) == 0 :
            self.entry2_label.config(text = "* Please Your N
ame!!!!!!")
        else:
            self.entry2_label.config(text = "")
        ################################### Validating DURAT
ION ##################
        DURATION = self.var.get()
        if DURATION == "HH:MM" :
            self.drop1_label.config(text = "* Please select
duration!!!!!!")
        else:
            self.drop1_label.config(text = "")
        ################################### Validating Brows
e 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 selec
t file or Take Picture ")
```

```
        else :
            ############################################## Sa
ving 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 alre
ady 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 = "* Eithe
r you can Browse pic or Take Picture.",fg = 'blue')
                    self.img_new = []
                    self.button1_label.config(text = "* Take
n 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 = "* y
our pic is register.....",fg = 'green')
                        messagebox.showinfo("Done","Successf
ully 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" + DU
RATION
                        f.writelines(detail)
                    path = 'images\\'
                    image_name = path + NAME + ".jpg"
                    self.pic_save = cv2.imwrite(filename = i
mage_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 Tr
acker")
        self.registration_window.iconbitmap("icon.ico")
```

```
        self.registration_window.configure(background = 'gra
y')

        frame1 = tk.Frame(self.registration_window, bg = 'gr
ay')
        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 =('verdan
a',12,'bold'))
        self.entry1.grid(pady=20,padx=20,row=0,column=1,stic
ky='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 = 'gr
ay', 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 =('verdan
a',12,'bold'))
```

```
        self.entry2.grid(pady=20,padx=20,row=1,column=1,stic
ky='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",b
g = '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","09:0
0","10:00","11:00","12:00")
        self.drop1.grid(pady=20,padx=20,row=2,column=1,stick
y='w')
        self.drop1_label = tk.Label(frame2,text = "",bg = 'g
ray', fg = "red",font = ('verdana',8,'bold'))
        self.drop1_label.grid(pady=20,padx=30,row=2,column=3
,sticky='w')

        ############################################## Browsi
ng Picture from file location ##############################

        self.var1 = tk.StringVar()
        self.var1.set("Select file location")
        self.entry3 = tk.Entry(frame2,bd = 4, font =('verdan
a',12),textvariable = self.var1)
        self.entry3.grid(pady=20,padx=20,row=3,column=1,stic
ky='w')

        def Browse_pic():
            self.registration_window.filedialog1  = filedial
og.askopenfilename(initialdir ="Desktop",title = "Select you
r photo",filetypes = (("jpg file","*.jpg"),("jpeg file","*.j
peg"),("png file","*.png")))
```

```
            self.var1.set(self.registration_window.filedialo
g1)

        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 = ('verd
ana',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,'b
old'),bg = '#262523',fg = 'white',height = 2, width = 35)
        button1.pack(anchor = 'center')
        self.button1_label = tk.Label(self.frame3,text = "Pr
ess '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,'b
old'),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, comm
and = 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 o
f 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)

    ########################################################## Fra
me1  ##################################
    frame1 = tk.Frame(about_window, bg = 'gray')
```

```
    frame1.place(relx = 0.11,rely = 0, relwidth = 0.80, relh
eight = 0.118)

    l1 = tk.Label(frame1,text = "Face Recognition Based Empl
oyee Activity Tracker",bg = 'gray', fg = "black",font = ('ve
rdana',20,'bold'))
    l1.pack(anchor = 'center',pady = 4 )

    ########################################################## Fra
me2  ###################################
    frame2 = tk.Frame(about_window,bg= 'gray')
    frame2.place(relx = 0.11,rely = 0.12, relwidth = 0.80, r
elheight = 0.75)

    l2 = tk.Label(frame2,text ='''This software aims to prov
ide a realtime continuous attendance system using the concep
t 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 afte
r certain interval of time making it an efficient way to
 \n
    ensure Employees availability through out the working ho
urs 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(an
chor = '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 :: tirthorajdasgupt
a@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, relh
eight = 0.15)

    l1 = tk.Label(frame1,text = "Face Recognition Based Empl
oyee Activity Tracker",bg = 'gray', fg = "black",font = ('ve
rdana',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')
    ########################################################## Bod
y 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,rel
height = 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 em
ail 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,rel
height = 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 ="gree
n",bg = "black",font = ('verdana',10))
    Subject_text_label.pack(anchor = "center")

    l3 = tk.Label(frame2,text = "Compose Message :",bg = "bl
ack", 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,rel
height = 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",fon
t = ('verdana',16,'bold')).grid(pady=50,padx=30,row=3,column
=0,sticky='w')
    tk.Label(frame2,text = "",bg = "black", fg = "white",fon
t = ('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 ente
r 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 fie
ld !!!!!!!!!! ")
        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 successful
ly....")
            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,'bo
ld'),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 o
f Help Window ##########################################
#############

########################################################## Home p
age 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, relh
eight = 0.118)
    frame2 = tk.Frame(window1,bg= 'gray')
    frame2.place(relx = 0.11,rely = 0.12, relwidth = 0.80, r
elheight = 0.118)
    frame3 = tk.Frame(window1, bg = 'gray')
    frame3.place(relx = 0.11,rely = 0.24, relwidth = 0.80, r
elheight = 0.643)
    frame4 = tk.Frame(window1, bg = 'gray')
    frame4.place(relx = 0.11,rely = 0.88, relwidth = 0.80, r
elheight = 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,myl
ist[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',fo
nt = ('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,du
ration,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_l
eft)
            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 Ba
sed Employee Activity Tracker",bg = 'gray', fg = "black",fon
t = ('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,calc
ulate_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_locatio
ns(imgs)
                encodefacesInFrame = face_recognition.face_e
ncodings(imgs,facesInFrame)
                for encodeface, faceloc_Cap in zip(encodefac
esInFrame,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 Duratio
n")
            self.reset_duration.iconbitmap("icon.ico")
            self.reset_duration.configure(background = 'blac
k')

            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,s
elf.var,*option)
            self.drop2.grid(pady=20,padx=20,row=0,column=1,s
ticky='e')
            self.drop2.configure(bg="gray",width = 10)

            self.reset_button = tk.Button(self.reset_duratio
n,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,col
umn=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,st
icky='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,stick
y='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 t
he duration!!!!!!")
                else:
                    with open("Details.txt",'w') as f:
                        data[2] = self.var.get()
                        f.writelines(data)
                        messagebox.showinfo("Done","Reset Du
artion Successfully.....")
                        self.reset_duration.destroy()

            else:
                self.l2.configure(text ="Please Register Fir
st!!!!!!")

    def reset_hour():
```

```
        obj = Reset_Working_hour()

    ################################## MenuBar ##############
#########################
    menubar = Menu(window1)
    filemenu = Menu(menubar, tearoff = 0)
    filemenu.add_command(label = 'Register New Employee', co
mmand = Register_page,font = ('verdana',8,'bold'))
    filemenu.add_command(label = 'Reset Working Duration', c
ommand = 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 = 'whi
te',height = 1, width = 10)
    button2.pack(anchor = 'w',side = 'bottom',pady = 10, pad
x = 50 )

    window1.iconbitmap("icon.ico")
```

```
############################################################ End of
 Home page window ##########################################
####################
home_page()
window1. mainloop()
```

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')
```

Conclusion

- Our project is a Machine Learning based Activity tracker using python programming.
- Our project is useful for various Induatrial organisations and also educational institutes.
- We Learned a lot about Python programming and its use in Machine Learning.
- We learned a lot about Python's various modules, libraries and their uses in making real-life applications.
- We learned about how HOG algorithm can be used for detection and recognition.
- We learned about various usability of Visual Studio Code.
- We Learned a lot about Machine Learning and its Applications.
- Human face recognition plays a very important role as part of modern surveillance and security applications. By applying the methods described in this article, accurate algorithms and quality face recognition results can be obtained. Moreover, with the help of HOG and SVM models, one can achieve high-performance levels in recognizing human faces and analyzing facial features, even in scenes containing complex backgrounds.

Bibliography

- Geeks For Geeks 
- YouTube 
- Rapid object detection using a boosted cascade of simple features.
- Kwok-Wai Wong, Kin-Man Lam, Wan-Chi Siu, An efficient algorithm for human face detection and facial feature extraction under different
- conditions, The Journal of the Pattern Recognition Society, (Aug 2000)
- Face Detection using Haar Cascades