Project Report

On

Real-Time Intrusion Detection Security System

SUBMITTED BY

Samarpan Biswas **ID: sb6165**

Ishita Chowdhury **ID: ivc211**

Rachana Swamy **ID: rms816**

Year (2019)

Date: 5th May 2019

Certificate

TO WHOMEVER IT MAY CONCERN

This is to certify that the following students have completed their final class project "Real-Time Intrusion Detection Security System" in the partial fulfillment of the awards of the degree of M.S. in Computer Engineering submitted in the Machine Learning Course of NYU Tandon School of Engineering.

The team compromises of following students:

1. Samarpan Biswas
2. Ishita Chowdhury
3. Rachana Swamy

(Professor Sundeep Rangan)
Project Mentor

ACKNOWLEDGEMENT

At this stage of our semester when we have finished our project for Machine Learning course, we cannot help but think that this has been a collective effort, rather than individual contribution. From the very first day that we started taking this course till the very last lecture, including every single extra help meeting, we always felt the help, motivation and support of so many people.

Our completion of this final project could not have been accomplished without the excessive supervision and support of Professor Rangan. We consider ourselves extremely lucky that we have found a professor like him to guide and motivate us in our project as well as in our lectures. We would like to thank Professor Rangan, for guiding us efficiently and providing help and suggestion selflessly throughout the whole project. We received complete cooperation, resources and information from their expertise which not helped us to finish the project but also to discover and research many other topics. We are forever indebted to them for giving their precious time to us and examine any problems that we had.

INDEX

Contents Page Number

Chapter 1: INTRODUCTION

1.1 General Introduction:

Our everyday lives are explicitly and implicitly dependent on some form of face detection and recognition software. We encounter them more than we think. Be it our phone with its face recognition module, our office entry with a face detection camera, or any kind of security measure that employs and depends on face recognition in real time. This module has been used explicitly in various projects but still has wide ranges of potential applications that it can be used for. For this project we decided to research this face recognition domain and dynamically adapt and conform it to be used in Intrusion Detection Security system. In this project, we have successfully reached our goal of real time face recognition and intruder detection using a facial recognition module embedded in a GUI interface along with other security modules. Later, this can be deployed in houses as a "Security System Module", that uses camera and security panel to ensure safety of a house by monitoring every corner of the house.

1.2 Related Work:

Decent amount of research has been done on this newly proposed idea of face recognition in small servers and in mobile services. A lot of projects and works have dived into this idea to be successfully implemented for usage in real time. However, to prove any kind of face detection and recognition software is robust and dynamically adaptive, it must show accurate results over multiple trials and experiments

Guo et al. has worked on to create a framework to successfully recognize one million celebrities from their images, by using all the pictures of their faces found on the internet as a database and training method. This method can be used to improve face detection algorithm by improving its accuracy and potentially deploying it to be used in various real-time applications such as live video analysis and work with large training data sets [1].

Li and Zhu, 2016 have worked on Face recognition technology that has been implemented in mobile phone servers. This paper's work puts mobile phone camera to be used as primary capturing device. It takes pictures using the phones camera and then sends them to a back server to be properly trained using a neural network. The result of this work shows that face recognition can be accurately done on a cellphone by simplifying and reducing the performance requirement of the cellphone otherwise [2].

However, our research and project focus on home security systems and how face recognition module can be used to ensure safety of one's house. The face recognition module used in our

project uses deep learning library called “dlib” which is built for implementation in python. The implementation of a user friendly GUI interface not only makes the overall system easy to use but also modify security components, such as passwords, lock key manually. Usual approaches require hundreds of pictures of a user to train the detection module, however, our project only requires one picture of the user to be trained with and it is enough to find accurate face recognition in very short amount of time and without errors.

1.3 Our Project Introduction:

Dynamic Target Recognition is a big and emerging field in Real-Time image and video processing. Its uses range from security protocols to gaming devices. For the final project of this class, we collectively have decided to work on Webcam based intrusion detection and face recognition by using python programming language. The main idea of the project is to have a GUI interface that embeds security modules, real- time face recognition module and instant notification on intrusion.

The user of this system provides picture of each family member or person it trusts to enter in the house in .jpg format to the backend server. This for our project will simply mean our backend database to have pictures of the family members and trusted person saved with their respective names. Names are used as labels to train the respective faces using facial recognition training module. Once the system is installed, it will keep detecting for the intruders. On enabling the full lock, only the trusted persons or the family members whose photos were stored and trained in the backend sever who for the facial recognition module are “known” person will have access to enter the house or use the security panel. On detection of intruders or „unknown“ person, the user will be notified, and their photos will be stored on the backend server. The security panel installed acts as a GUI Module.

The GUI module along with other security module (like System Lock, Full Lock and Password Change) embedded in it, uses “tkinter” library from python while facial recognition and facial training module uses “dlib” and “face_recognition” libraries along with other numerous library available in the python programming language. The photos of the intruder or unknown persons are appended with real-time timestamps and then stored on the back-end server. They can be accessed later by logging into the back-end server.

Chapter 2: IMPLEMENTATION

We have built a Real time Intrusion Detection System that incorporates two modules, namely an interactive GUI and an Intrusion Detection module.

2.1 GUI

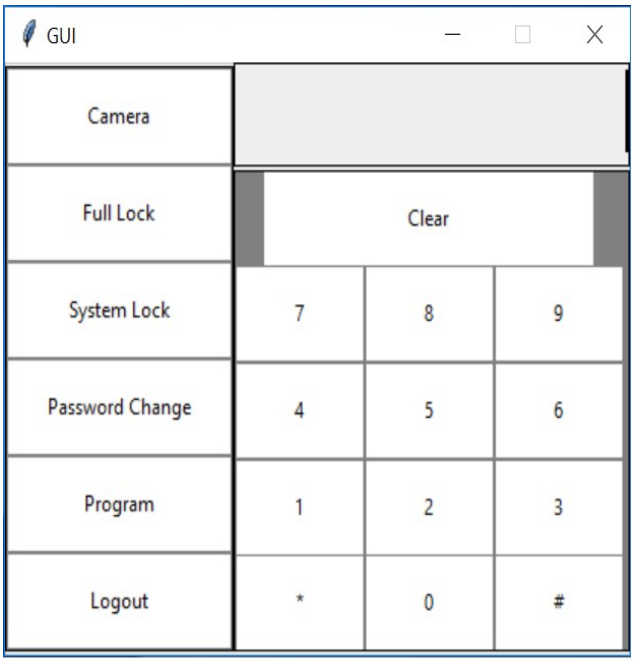

Figure 2.1: GUI

We have two security modules embedded in the GUI that can enabled/disabled using passcodes.

1. **System Lock:**

   What is the function?
   Let's first think of a situation when the device is installed in a family's house and it has access to multiple cameras that have been installed at inner and surrounding corners of the house.

   If the user enables the System Lock, it locks the features of changing passwords and adding new faces to the facial training module embedded in the GUI. Only the Camera (i.e. facial recognition module) button and the Full lockdown feature are accessible. In this case no outsider or intruder can manually access and change the passwords for the

two security modules and use the "Program" button to add a new face to the back-end server for facial training.

Why is the Camera button still accessible?

The user might want to use the camera option to check activities on every camera attached to the system. In case there is a stranger or an intruder presence while everyone is inside the house, the user can instantly see that. If intruder detected, then pictures along with timestamp of the intruder will be taken and stored in the database of the back-end server.

Why is the full lockdown button still accessible?

The user might want to put the system on full lockdown (if the family decides to go out of the house for some time) provided the user enters the correct password and it triggers the facial recognition module automatically.

Another case:

This can also be helpful for the user to prevent tempering from the children playing around.

2. **Full Lock**

What is the function?

Now let's think of a situation when the family is going for a vacation and no one will be in the house for a significant amount of time. The user obviously will want to activate a Full Lockdown which will automatically trigger facial recognition module to be alarmed and notify the user for intruder activity.

How does it help?

When the full lockdown in activated, activities of intruders or burglars can be reported. Since the facial recognition module is on full lockdown mode, it will be on the lookout for any such possibilities. If an unknown face is detected, pictures will be clicked and stored in the database and the user is notified. These activities can also be checked by logging to the back-end server.

Added feature:

Due to poor lighting and various other conditions, the facial recognition module can do mis-prediction maybe due to the slight resemblance of the person being recognized with someone else, maybe with one of the known persons. Our algorithm checks for such possibilities for a pre-set number of times and if it still finds the confidence value of the prediction to be high continues with the final decision. As an additional security measure, after the face unlock, the user is asked for the password to completely disable the full

lockdown mode. If wrong password entered, the access to GUI/Security-Panel is disabled and recognition module reruns the process.

Other Buttons in the GUI interface include Password Change, Program and logout buttons. Each one is explained in

detail below.

1. **Password Change**
   It provides options to change password of the Full Lock and as well as the System Lock. You still must enter the current password which has been already set, in order to set or modify a new password.

2. **Program**
   It opens a module that will allow the user, to add a new face in the back-end server for the facial training model to train this face as a „known" face. The entered name acts as a label for the captured face.

3. **Logout**
   Completely shuts down the system.

2.2 Face Recognition Module

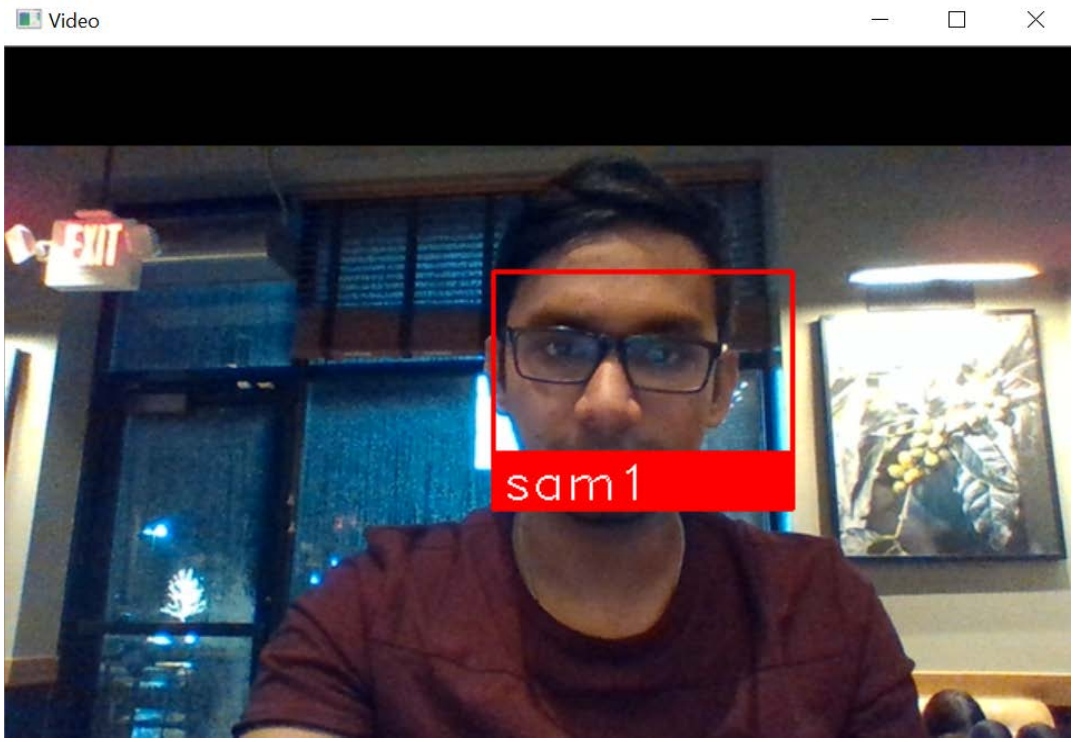

Figure 1.2: Face Recognition

How did we implement Face recognition?

Every Machine Learning algorithm takes a dataset as input and then learns from that input data and then it goes on to identifying patterns in the data

There are multiple things we can look at as a pattern:

- Height/width of the face
- Color of the face in RGB format
- Width of other parts of the face like lips, nose, etc.
- Hair patter and eye brow gap

As we can clearly see, there is a pattern here – different faces have different dimensions and similar faces have similar dimensions.

Machine Learning algorithms only understand numbers. This numerical representation of a “face” is termed as a feature vector.

Hence, our image is now a vector that could be represented as, say (23.1, 15.8, 255, 224, 189, 5.2, 4.4)

The situation will look like something depicted below:

Height of face (cm)	Width of face (cm)	Average color of face (RGB)	Width of lips (cm)	Height of nose (cm)
23.1	15.8	(255, 224, 189)	5.2	4.4

Now, once we have encoded each image into a feature vector, the problem can be proceeded with. It can be seen clearly that when we have 2 faces that represent the same person, the feature vectors derived will be quite similar. The “distance” between the 2 feature vectors will be quite small.

Machine Learning can help us with 2 things:

- **Deriving the feature vector:** A Machine Learning algorithm can intelligently label out many of such features as mentioned above. For example, a complex feature could be: ratio of height of lips, nose and width of forehead. Now it will be quite difficult for a human to list down all such features in details as there can be many.
- **Matching algorithms:** Once the feature vectors have been obtained, a Machine Learning algorithm needs to match a new image with the set of feature vectors that have already been stored in the database.

We have used a Python library that encapsulates all of what we have done above by creating feature vectors out of faces and knowing how to differentiate across faces. This Python library is called as face_recognition and deep within, it employs dlib – a modern C++ toolkit that contains several machine learning algorithms.

The Code implements the following process:

- Looping over each image.
- Encoding the image into a feature vector.
- Comparing the loaded image with the image to be recognized.
- If it is a match, we show it in real time and allow the user to perform a set of functions.

The detailed code has been listed down in the Appendix section.

Chapter 3: RESULTS

3.1 Result

Case 1: When System Lock and Full Lock both are disabled

(1) Password for System Lock and Full Lock can be changed

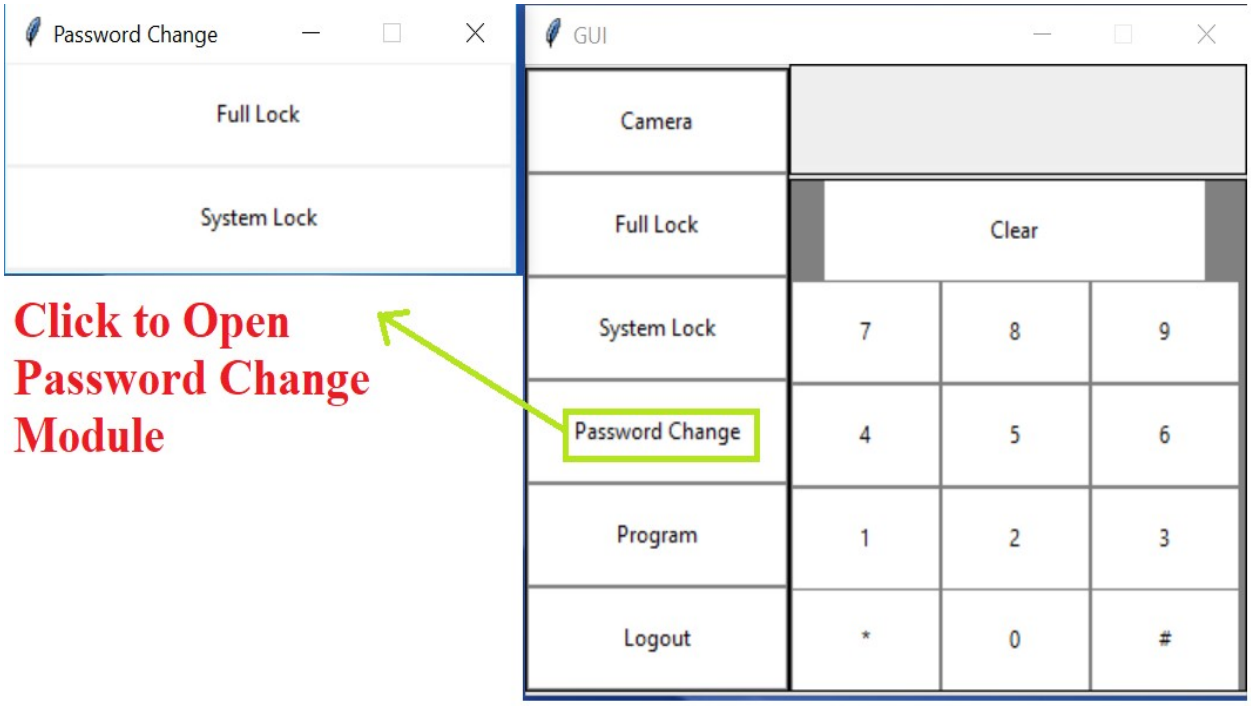

Figure 3.1.1.1: Password Change Module

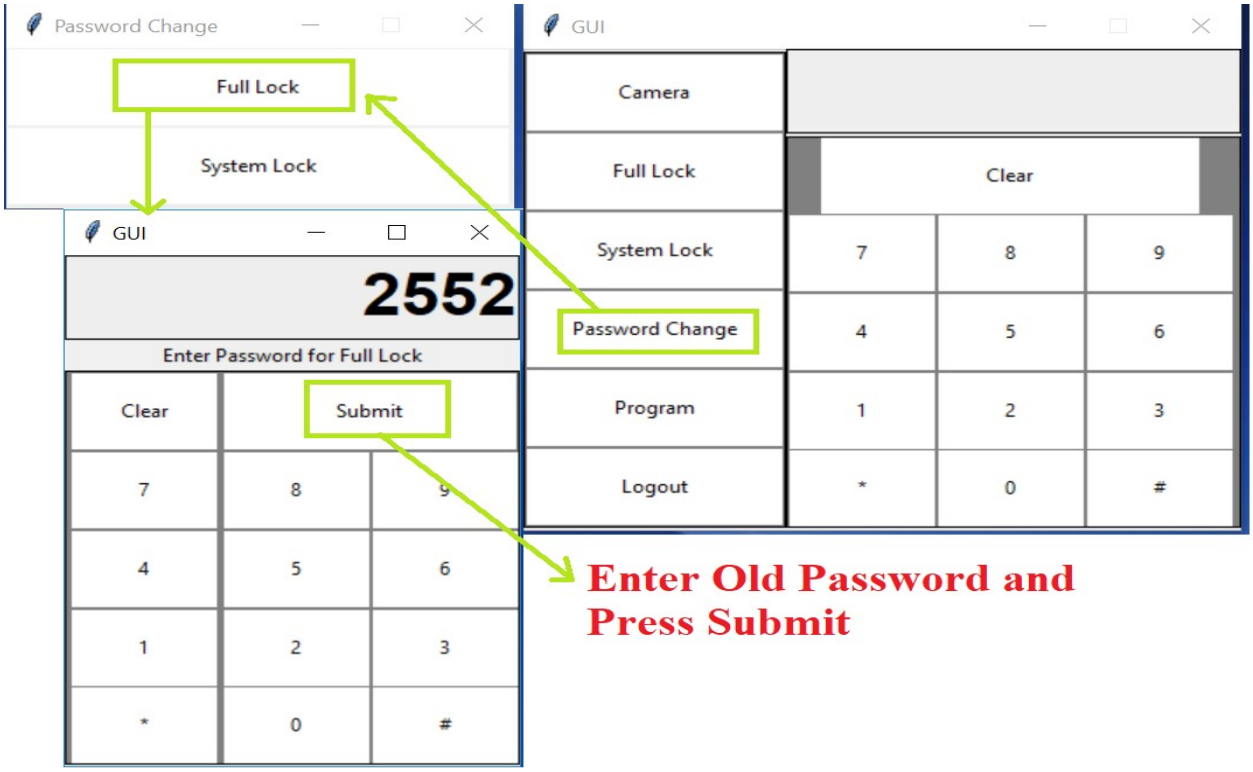

Figure 3.1.1.2: Step1: Enter Old Password to Change Password for Full Lock

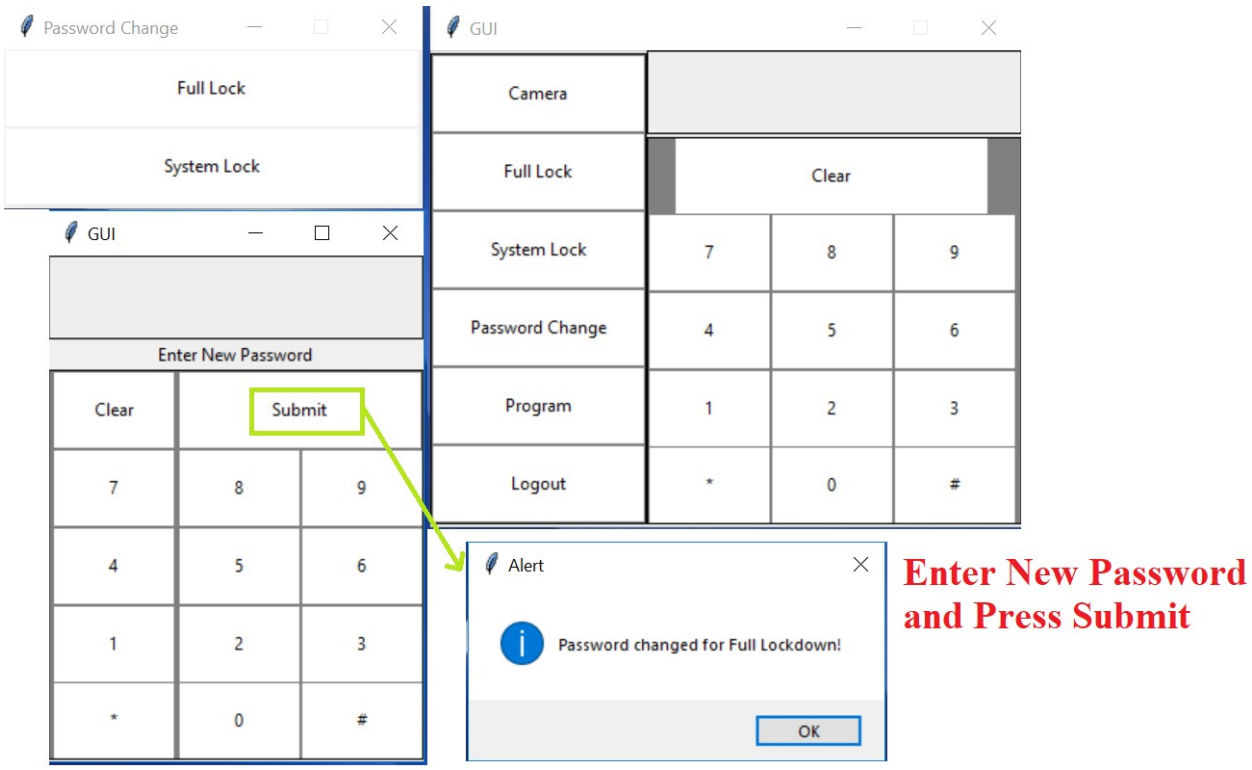

Figure 3.1.1.3: Step2: Enter New Password for Full Lock and Password is Changed

(2) Adding New Faces to Facial Training Module

Before:

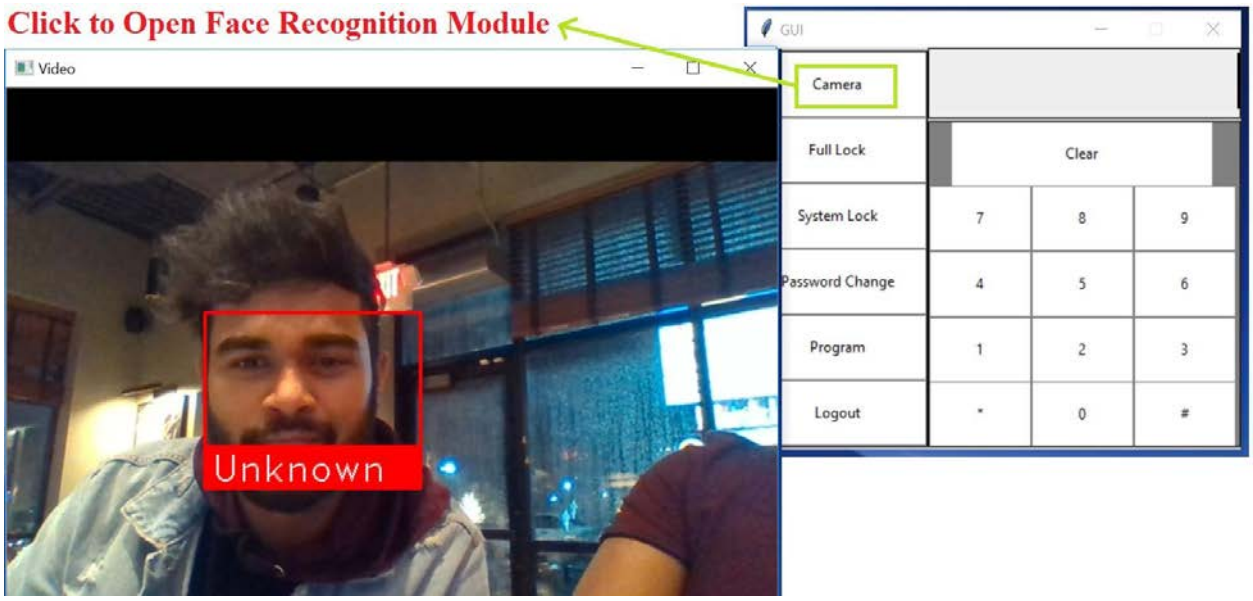

Figure 3.1.1.4: Facial Recognition Before Training the face

After:

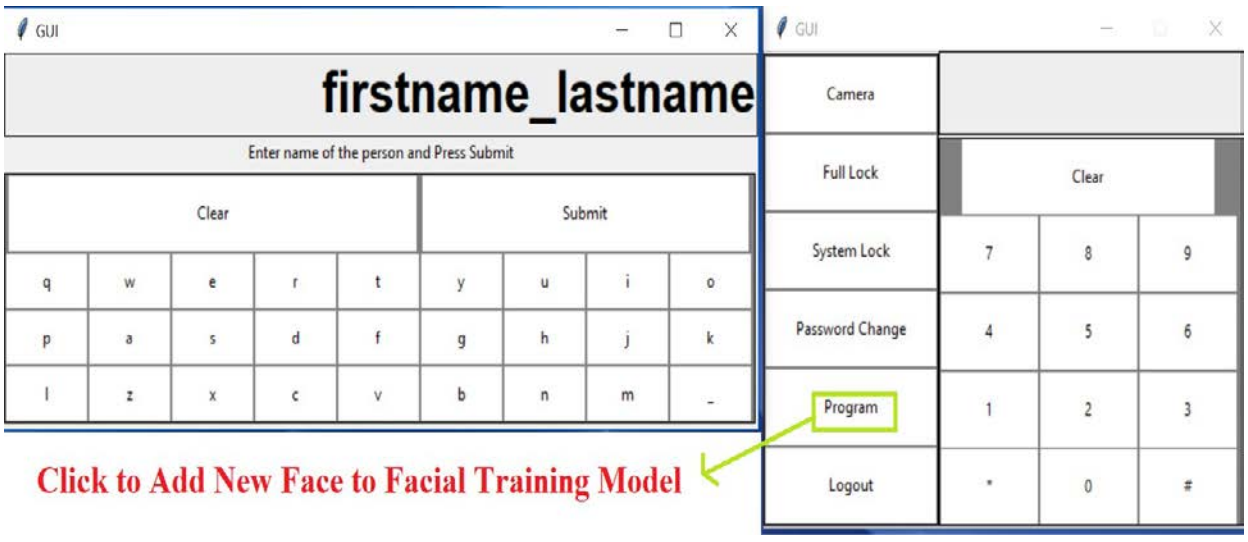

Figure 3.1.1.5: Step 1: Open the Module

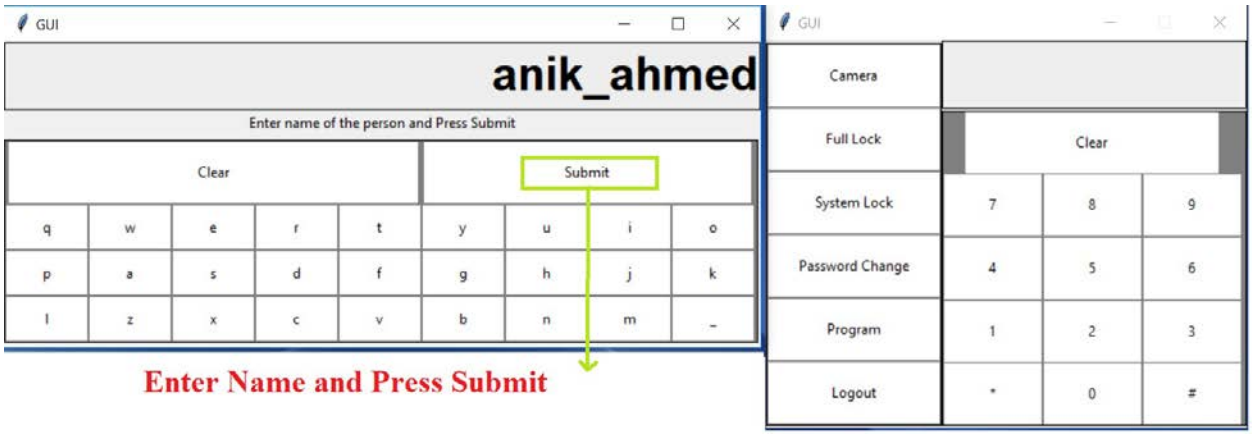

Figure 2.1.1.6: Step 2: Enter Name and Press Submit.

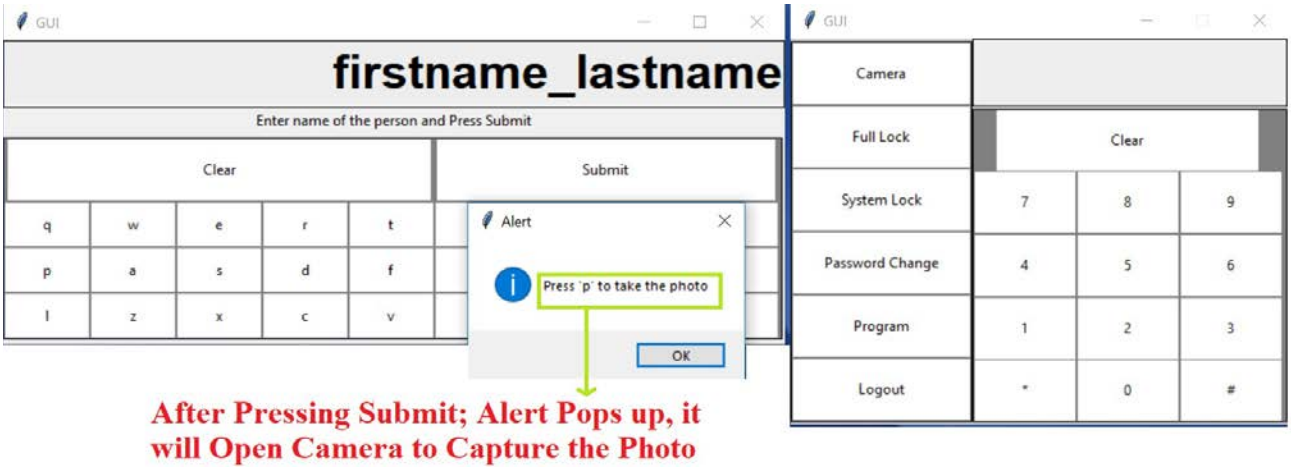

Figure 3.1.1.7: Step 3: Press Ok to Open Camera.

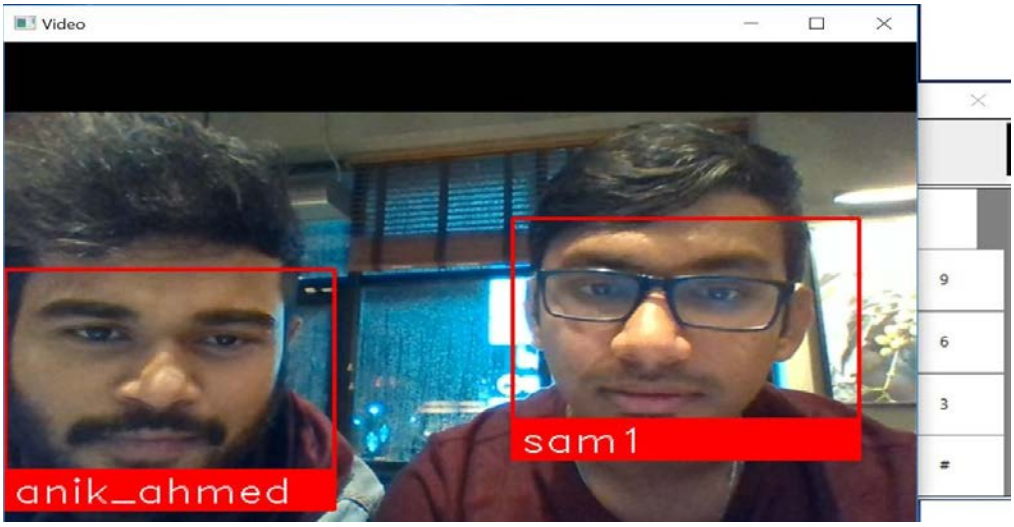

Figure 3.1.1.8: Step 4: Open Camera and Face gets Detected.

Case 2: When System Lock is Enabled and Full Lock Disabled

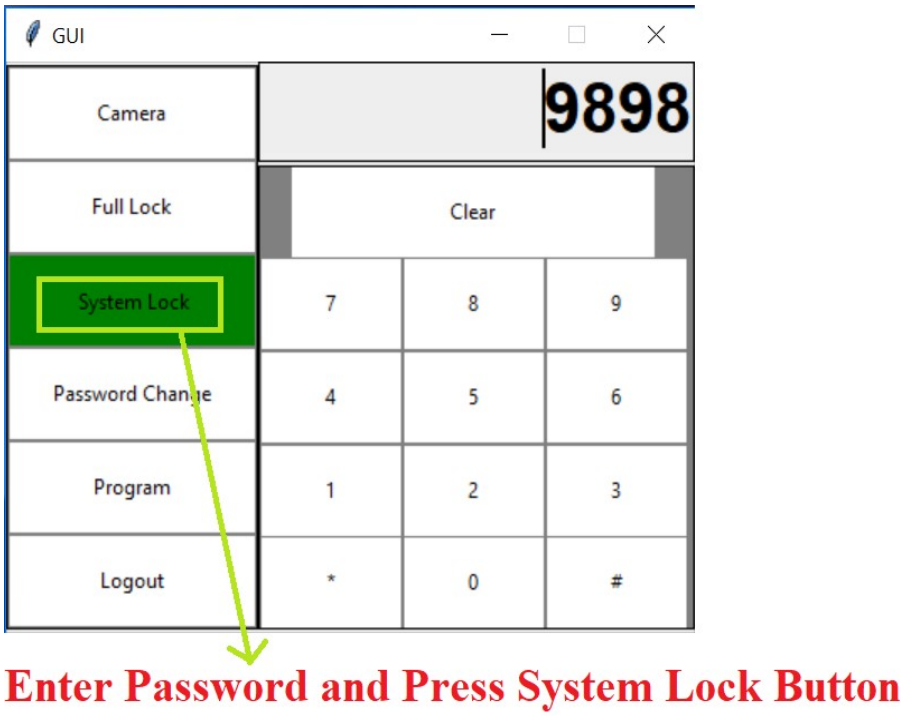

Figure 3.1.2.1: Enable System Lock

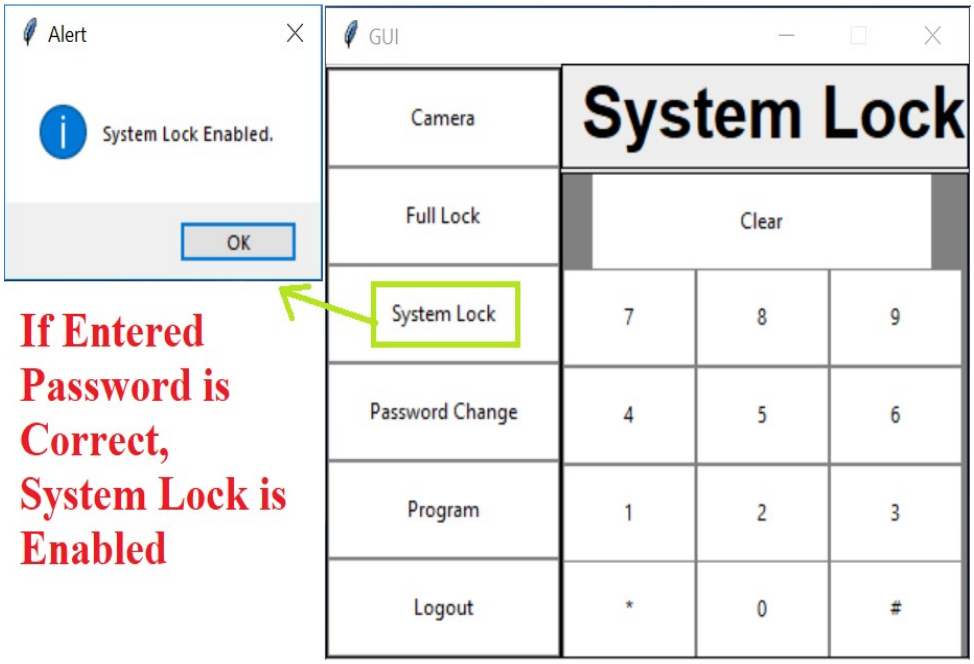

Figure 3.1.2.2: System Lock Enabled

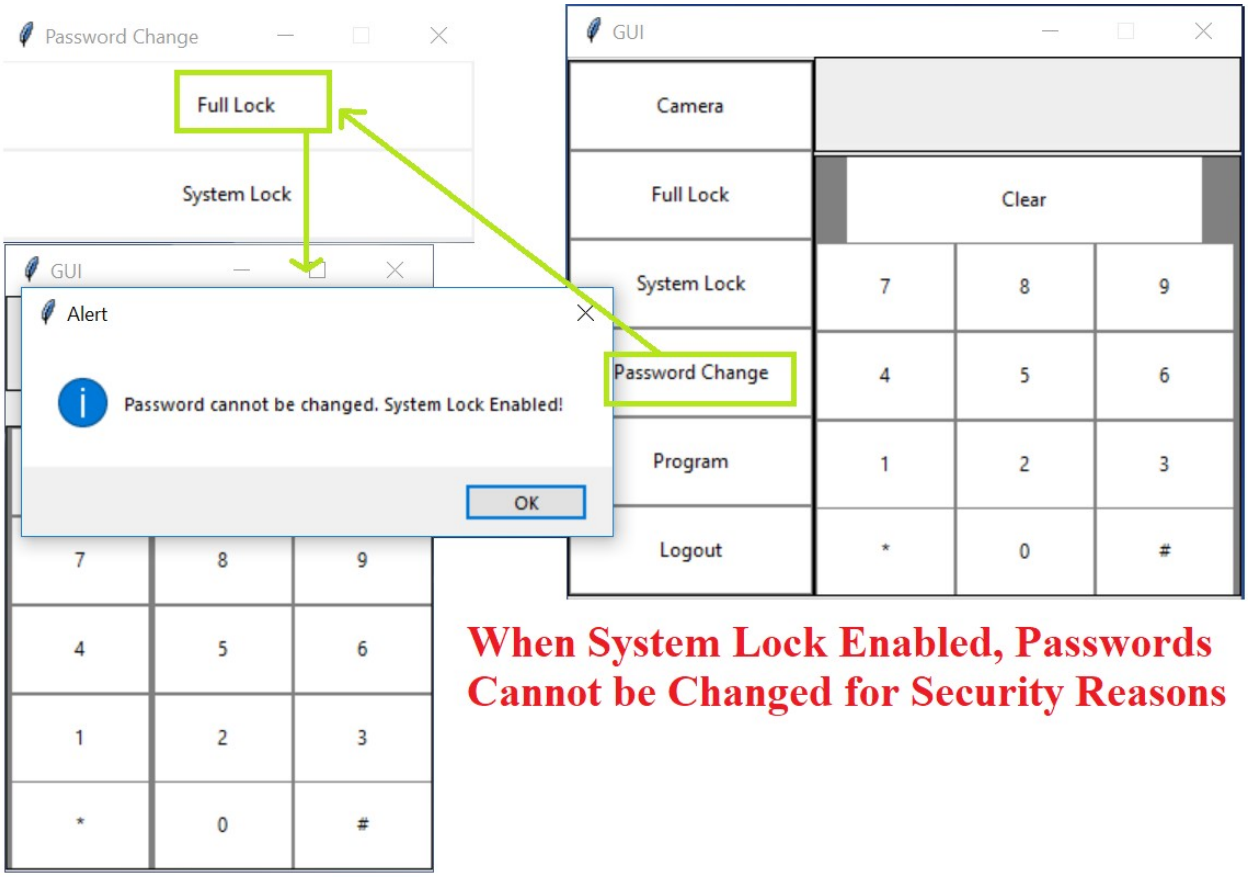

Figure 3.1.2.3: Password Change Disabled

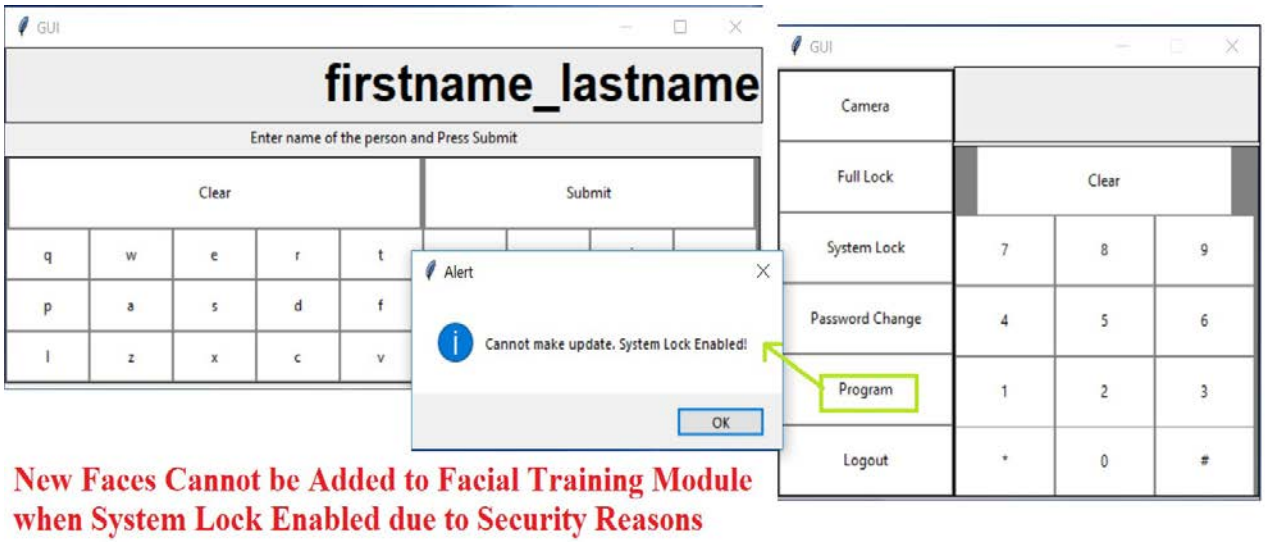

Figure 3.1.2.4: Facial Training Module Disabled

Case 3: System Lock and Full Lock both Enabled

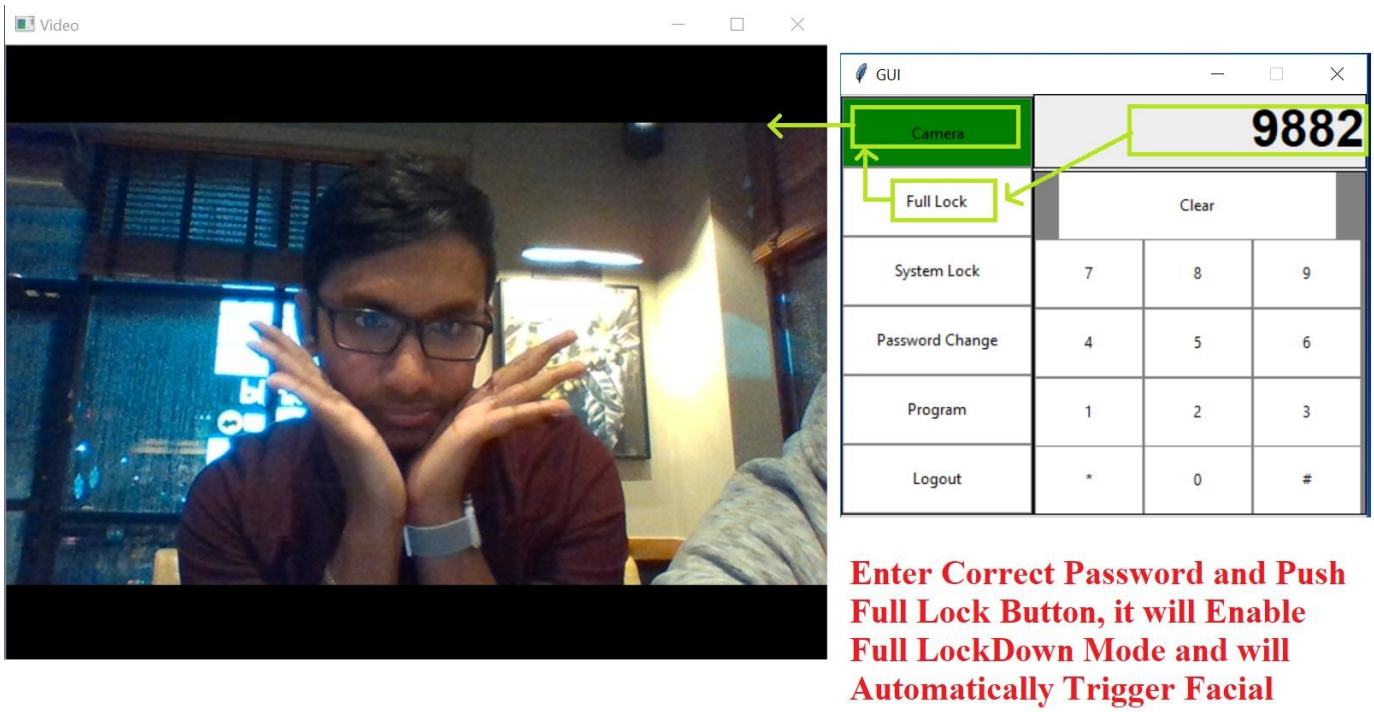

Figure 3.1.3.1: Enter Password to Enable Full Lockdown Mode

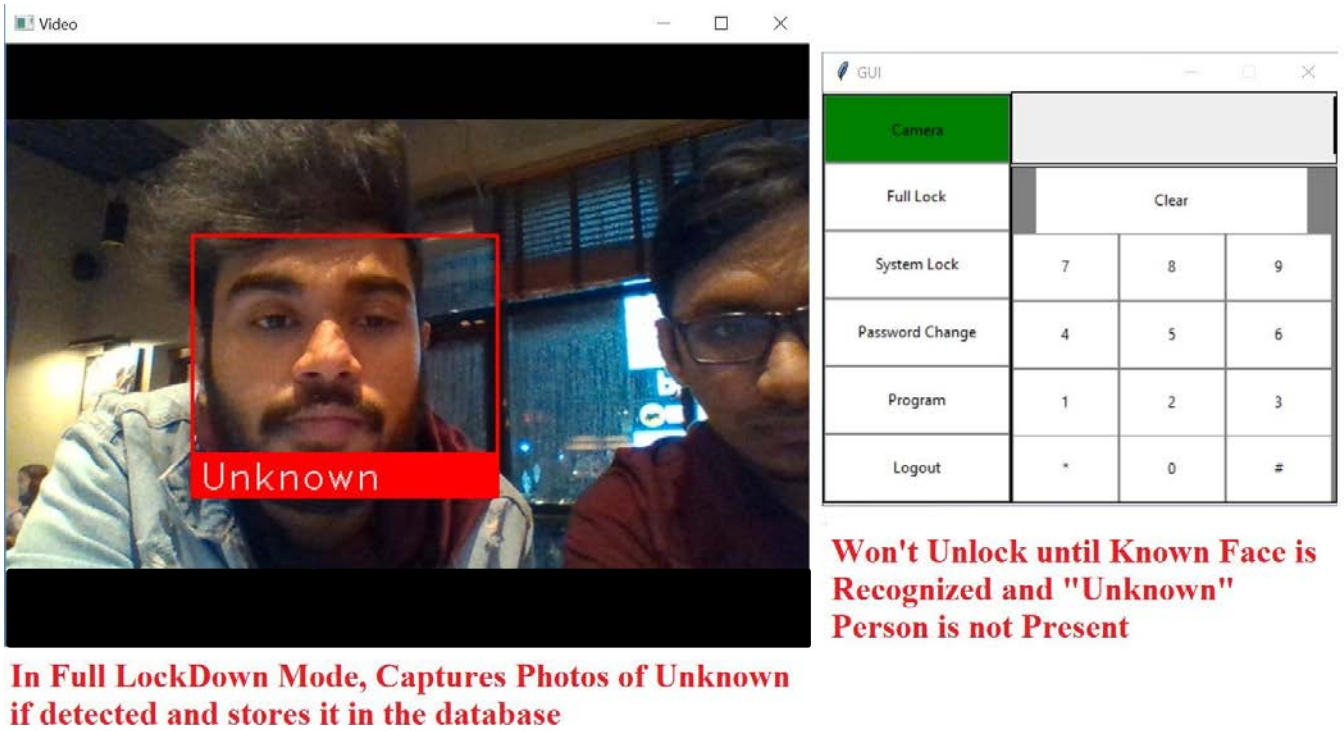

Figure 3.1.3.2: Intrusion Detected, Capture the Photo. Full Lockdown in Process

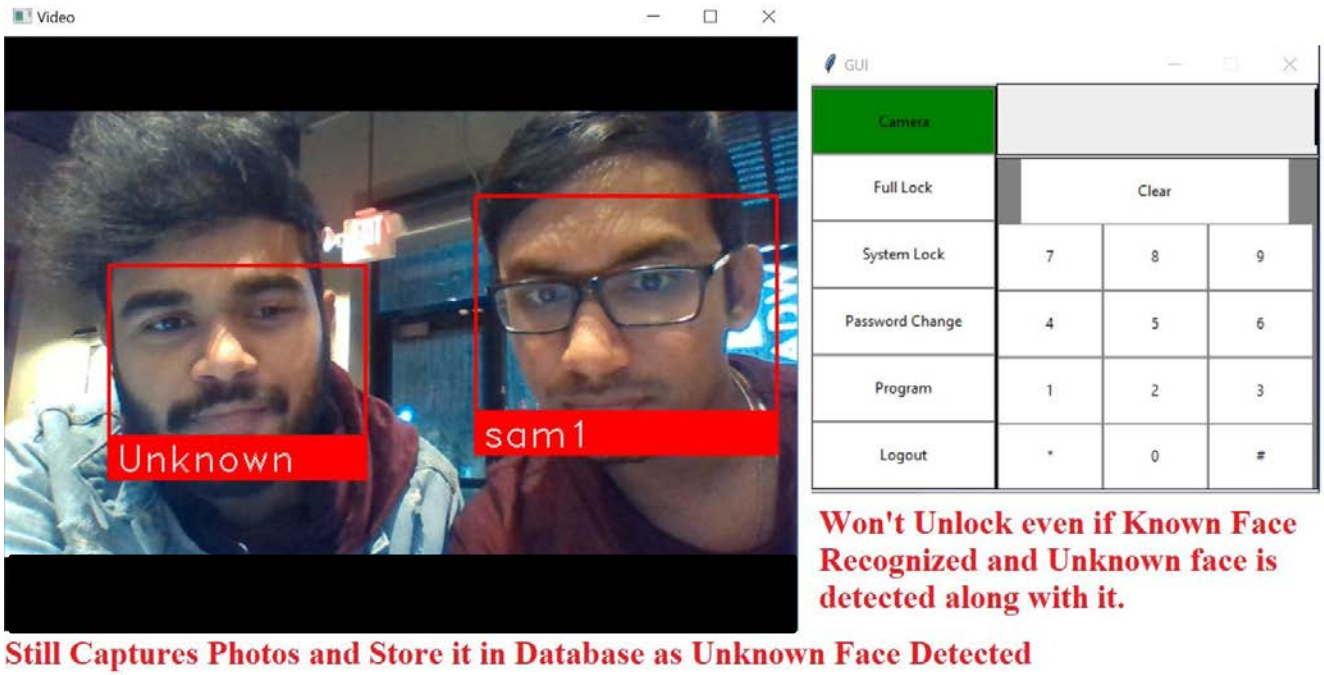

Figure 3.1.3.3: Presence of Unknown person, Capture Photo. Full LockDown still in Process.

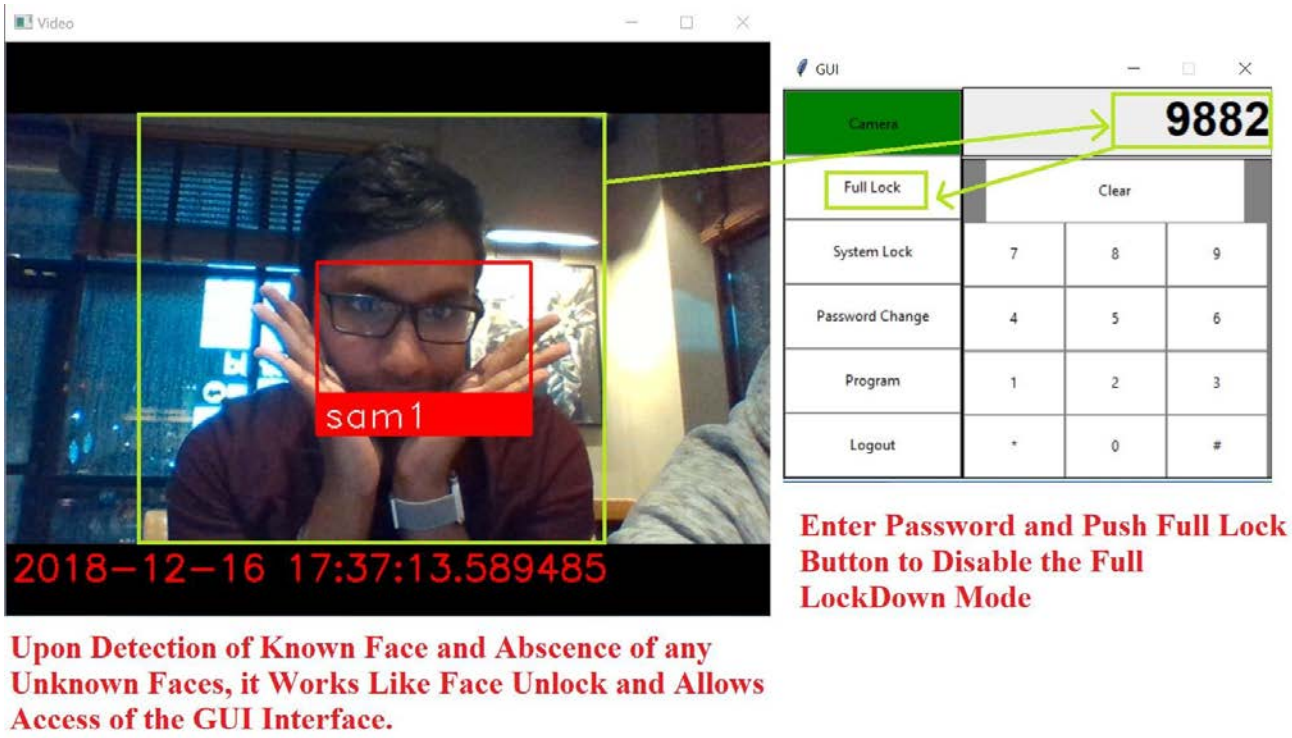

Figure 3.1.3.4: Upon Detection of Known Face, Unlock the GUI Access. Enter Password to Disable

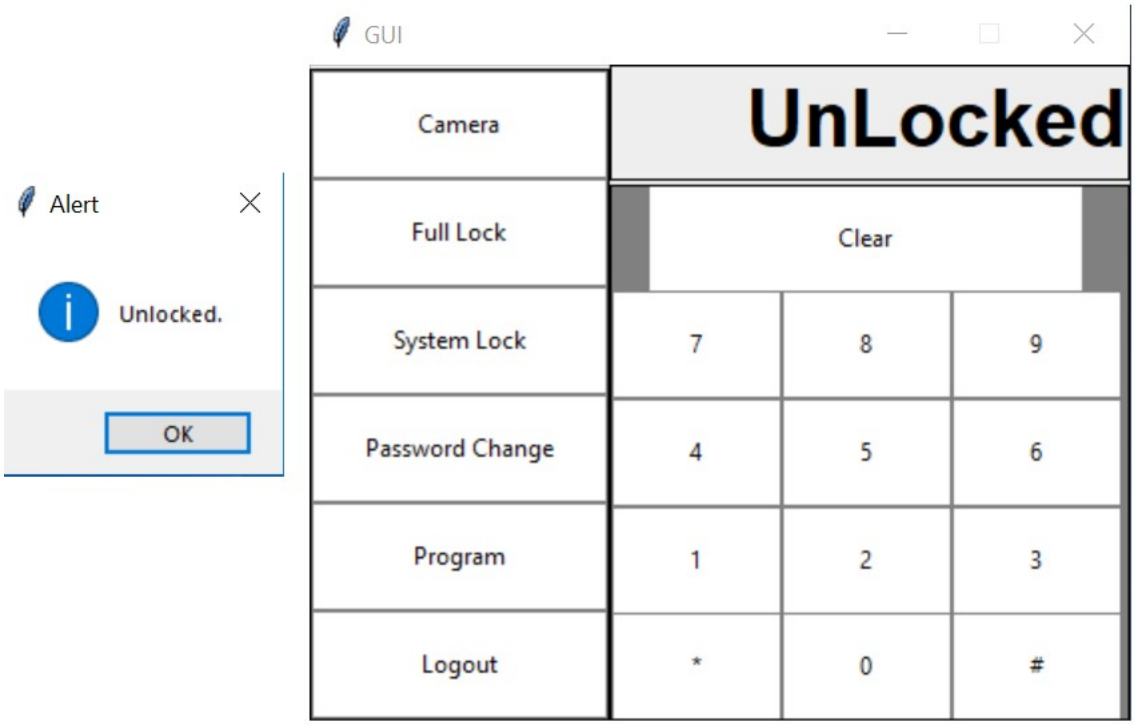

Figure 3.1.3.5: Full Lockdown Disabled.

3.2 Advantages/Disadvantages:

The advantages and disadvantages are given below to highlight the main functions of this project.

Advantages:

- The face recognition is getting used in a security control framework which also uses a GUI.
- The GUI employed makes it easier for the user to explore all the options and use it accordingly
- The algorithm makes high confident prediction of the detected face and allows training module to train the model using just one picture saved of that each person.
- The face recognition module can be used to make sure all the people entering a house have been checked and given clearance based on the user's preference.
- Helps in real-time intrusion detection and instant notification for unwanted activities in the house.

Disadvantage:

- The facial recognition module needs proper lighting in order to be able to make highly confident prediction.

3.3 Future Works

We will make sure to include a feature which will email the pictures taken to the main user's mobile or post it on a mobile application. As an additional security measure, we will trigger an alarm which can be only turned off if the user activates unlocking mechanism from its end.

The current face recognition due to poor surrounding lights or low lights makes mis-prediction. We would like to do more research in future on how to capture the frame in poor lighting conditions and apply masks to make confident prediction. This will go on to make our system more robustly interactive and better suited for more usability and more marketable.

3.4 Conclusion

After a lot of research and multiple experiments with different faces we were able to successfully run our real-time intrusion detection security system. We have successfully embedded security modules, facial training and face-recognition modules to the GUI. The compilation works without glitches or errors. The pictures of the intruders are stored on the back-end server with the timestamp in .jpg format.

References

[1]. Guo, Y., Zhang, L., Hu, Y., He, X. and Gao, J. (2016). *MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition*. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1607.08221.pdf [Accessed 10 Dec. 2018].

[2] Li, H. and Zhu, X. (2016). *Face recognition technology research and implementation based on mobile phone system - IEEE Conference Publication*. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/7603310 [Accessed 10 Dec. 2018].

[3] https://github.com/ageitgey/face_recognition Adam Geitgey for the Face recognition module which we then implemented changes in.

Appendix

Code 1- GUI.py

```
import sys
import os
import face_recognition
import cv2
import numpy as np
import pandas as pd
import time
from datetime import datetime
from save_photo import go_snap
from Detection import intrusion_fn
if sys.version_info[0] < 3:
    from Tkinter import *
    import tkMessageBox
else:
    from tkinter import *
    from tkinter import messagebox



global PassLock
global PassChild
global expression
global expression_1
global expression_2
global expression_3
global dirname
global numfiles
global x_signal
```

```
global y_signal
global z_signal
global key_signal
global lckdwn
global parlckdwn
numfiles = 0
x_signal = 0
y_signal = 0
z_signal = 0
key_signal = 0
lckdwn = 0
parlckdwn  = 0
labels = []
file = open("Passwords.txt","r+")
p_signal = file.readlines()
for line in p_signal:
        words = line.split(" ")
PassLock = words[0]
PassChild = words[1]
print(PassLock)
print(PassChild)
file.close()

# creating basic window
window = Tk()
window.geometry("410x330") # size of the window width:- 500, height:- 375
window.resizable(0, 0) # this prevents from resizing the window
window.title("GUI")

def btn_click(item):
```

```
    global expression
    expression = expression + str(item)
    input_text.set(expression)
def btn_clear():
    global expression
    expression = ""
    input_text.set("")

def detect(detect1):
        intrusion_fn(detect1)

def btn_click_1(item_1):
    global expression_1
    expression_1 = expression_1 + str(item_1)
    input_text_1.set(expression_1)
def btn_clear_1():
    global expression_1
    expression_1 = ""
    input_text_1.set("")

def btn_click_2(item_2):
    global expression_2
    expression_2 = expression_2 + str(item_2)
    input_text_3.set(expression_2)
def btn_clear_2():
    global expression_2
    expression_2 = ""
    input_text_3.set("")

def btn_click_3(item_3):
```

```
    global expression_3
    expression_3 = expression_3 + str(item_3)
    input_text_5.set(expression_3)
def btn_clear_3():
    global expression_3
    expression_3 = ""
    input_text_5.set("firstname_lastname")

def btn_Lock(try1):
        global parlckdwn
        global lckdwn
        global expression
        global PassLock
        #if parlckdwn == 0:
        if lckdwn == 0: #to lock
                if try1 == PassLock: input_text.set("Lock down")
                        messagebox.showinfo("Alert","Complete Lock down initiated.")
                        input_text.set("")
                        expression = ""
                        lckdwn = 1
                        detect(lckdwn)
                else:
                        input_text.set("Error")
                        expression = ""
        elif lckdwn == 1: #to unlock
                if try1 == PassLock:
                        input_text.set("UnLocked")
                        messagebox.showinfo("Alert","Unlocked.")
                        input_text.set("")
```

```
                expression = ""
                lckdwn = 0
            else:
                input_text.set("Error")
                expression = ""
                messagebox.showinfo("Alert","Wrong Password Entered")
                detect(lckdwn)
#       elif parlckdwn == 1:
#           input_text.set("Error")
#           messagebox.showinfo("Alert","First Disable Parental Lock")
#           input_text.set("")
#           expression = ""

def btn_SystemLock(try2):
    global lckdwn
    global parlckdwn
    global expression
    global PassChild
    if lckdwn == 0: #to lock
        if parlckdwn == 0:
            if try2 == PassChild: input_text.set("System Lock")
                messagebox.showinfo("Alert","System Lock Enabled.")
                input_text.set("")
                expression = ""
                parlckdwn = 1
            else:
                input_text.set("Error")
                expression=""
        elif parlckdwn == 1:
```

```
                    if try2 == PassChild: input_text.set("UnLocked")
                        messagebox.showinfo("Alert","System Lock Disabled.")
                        input_text.set("")
                        expression = ""
                        parlckdwn = 0
                    else:
                        input_text.set("Error")
                        expression=""
        elif lckdwn == 1:

            input_text.set("Error")
            messagebox.showinfo("Alert","Full Lockdown. Can't access System Control")
            input_text.set("")
            expression = ""


def submit_keyboard():
        global expression_3
        global key_signal
        global lckdwn
        global parlckdwn global dirname
        input_text_5.set("firstname_lastname")
        dirname = expression_3
        expression_3=""
        if lckdwn == 0 and parlckdwn == 0:
            if dirname != "":
                messagebox.showinfo("Alert", "Press 'p' to take the photo")
                go_snap(dirname)
            else:
```

```
                messagebox.showinfo("Alert", "No input given")
        else:
                if lckdwn == 1:
                        messagebox.showinfo("Alert", "Cannot make update. System on Full
Lockdown!")
                elif parlckdwn == 1:
                        messagebox.showinfo("Alert", "Cannot make update. System Lock
Enabled!")


def Change_it_1():
        global x_signal
        global z_signal
        global lckdwn
        global parlckdwn
        global expression_1
        global PassLock
        global PassChild
        if lckdwn == 0 and parlckdwn == 0:
                if x_signal == 0:
                        if expression_1 == PassLock:
                                input_text_1.set("")
                                input_text_2.set("Enter New Password")
                                expression_1=""
                                x_signal = 1
                        else:
                                input_text_1.set("Error")
                                expression_1=""
                elif x_signal == 1:
                        z_signal = str(expression_1) + str(" ") + str(PassChild)
                        print(z_signal)
```

PassLock = expression_1

```
                print(PassLock)
                file = open("Passwords.txt", "w")
                file.seek(0,0)
                file.write(z_signal)
                file.close()
                messagebox.showinfo("Alert","Password changed for Full Lockdown!")
                x_signal = 0
                input_text_2.set("Enter Password for Full Lock")
                input_text_1.set("")
                expression_1=""
        else:
            if lckdwn == 1:
                input_text_1.set("Error")
                messagebox.showinfo("Alert","Password cannot be changed. System on
Full Lockdown!")
                input_text_1.set("")
                expression_1=""
            elif parlckdwn == 1:
                input_text_1.set("Error")
                messagebox.showinfo("Alert","Password cannot be changed. System Lock
Enabled!")
                input_text_1.set("")
                expression_1=""


def Change_it_2():
        global y_signal
        global z_signal
        global lckdwn
        global parlckdwn
        global expression_2
```

```
global PassChild
global PassLock
if lckdwn == 0 and parlckdwn == 0:
        if y_signal == 0:
                if expression_2 == PassChild:
                        input_text_3.set("")
                        input_text_4.set("Enter New Password")
                        expression_2=""
                        y_signal = 1
                else:
                        input_text_3.set("Error")
                        expression_2=""
        elif y_signal == 1:
                z_signal = str(PassLock) + str(" ") + str(expression_2)
                print(z_signal)
                PassChild = expression_2
                print(PassChild)
                file = open("Passwords.txt", "w")
                file.seek(0, 0)
                file.write(z_signal)
                file.close()
                messagebox.showinfo("Alert","Password changed for system Lock!")
                y_signal = 0
                input_text_4.set("Enter Password for Full Lock")
                input_text_3.set("")
                expression_2=""
else:
        if lckdwn == 1:
                input_text_3.set("Error")
```

```
                messagebox.showinfo("Alert","Password cannot be changed. System on
Full Lockdown!")
                input_text_3.set("")
                expression_2=""
            elif parlckdwn == 1:
                input_text_3.set("Error")
                messagebox.showinfo("Alert","Password cannot be changed. System Lock
Enabled!")
                input_text_1.set("")
                expression_2=""



def Pass_window():
    newwin1 = Toplevel(window)
    newwin1.geometry("290x110")
    newwin1.resizable(0,0)
    newwin1.title("Password Change")
    Op1 = Button(newwin1, text = "Full Lock", fg = "black", width = 40, height = 3, bd = 0, bg =
"#fff",activebackground="green",cursor = "hand2", command = lambda:
Pass_select_Op1()).grid(row = 0, column = 0, columnspan = 1, padx = 1, pady = 1)
    Op2 = Button(newwin1, text = "System Lock", fg = "black", width = 40, height = 3, bd = 0, bg
= "#fff",activebackground="green",cursor = "hand2", command = lambda:
Pass_select_Op2()).grid(row = 1, column = 0, columnspan = 1, padx = 1, pady = 1)


def Pass_select_Op1():
        global expression_1
        global x_signal
        x_signal = 0
        expression_1=""
        input_text_1.set("")
        input_text_2.set("Enter Password for Full Lock")
        newwin2 = Toplevel(window)
```

```
        newwin2.geometry("260x350")

    #newwin2.resizable(1,1)

        input_frame_1 = Frame(newwin2, width = 30, height = 20, bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

        input_frame_1.pack(side = TOP,ipady=2)


        input_field_1 = Entry(input_frame_1, font = ('arial', 30, 'bold'), textvariable = input_text_1, width = 30, bg = "#eee", bd = 0, justify = RIGHT)

        input_field_1.grid(row = 0, column = 0,columnspan=3)

        input_field_1.pack(ipady=2) # 'ipady' is internal padding to increase the height of input field


        Label1 = Label(newwin2,textvariable=input_text_2)

        Label1.pack()


        btns_frame_1 = Frame(newwin2, width = 50, height = 330, bg = "grey",bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

        btns_frame_1.pack(side=LEFT,ipadx=0)

        clear_1 = Button(btns_frame_1, text = "Clear", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_clear_1()).grid(row = 0, column = 0, columnspan = 1, padx = 3, pady = 1)

        submit_bt = Button(btns_frame_1, text = "Submit", fg = "black", width = 23, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: Change_it_1()).grid(row = 0, column = 1, columnspan = 2, padx = 0, pady = 1)


        seven_1 = Button(btns_frame_1, text = "7", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(7)).grid(row = 1, column = 0, padx = 1, pady = 1)

        eight_1 = Button(btns_frame_1, text = "8", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(8)).grid(row = 1, column = 1, padx = 1, pady = 1)

        nine_1 = Button(btns_frame_1, text = "9", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(9)).grid(row = 1, column = 2, padx = 1, pady = 1)
```

```
        four_1 = Button(btns_frame_1, text = "4", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(4)).grid(row = 2, column = 0, padx = 1, pady = 1)

        five_1 = Button(btns_frame_1, text = "5", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(5)).grid(row = 2, column = 1, padx = 1, pady = 1)

        six_1 = Button(btns_frame_1, text = "6", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(6)).grid(row = 2, column = 2, padx = 1, pady = 1)


        one_1 = Button(btns_frame_1, text = "1", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(1)).grid(row = 3, column = 0, padx = 1, pady = 1)

        two_1 = Button(btns_frame_1, text = "2", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(2)).grid(row = 3, column = 1, padx = 1, pady = 1)

        three_1 = Button(btns_frame_1, text = "3", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(3)).grid(row = 3, column = 2, padx = 1, pady = 1)


        astk_1 = Button(btns_frame_1, text = "*", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1("*")).grid(row = 4, column = 0,padx = 1, pady = 0)

        zero_1 = Button(btns_frame_1, text = "0", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1(0)).grid(row = 4, column = 1,padx = 1, pady = 0)

        hstg_1 = Button(btns_frame_1, text = "#", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_1("#")).grid(row = 4, column = 2, padx = 1, pady = 0)




def Pass_select_Op2():

        global expression_2

        global y_signal

        expression_2=""

        input_text_3.set("")

        y_signal = 0
```

```
        input_text_4.set("Enter Password for System Lock")

        newwin3 = Toplevel(window)

        newwin3.geometry("260x350")

    #newwin3.resizable(1,1)

        input_frame_2 = Frame(newwin3, width = 30, height = 20, bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

        input_frame_2.pack(side = TOP,ipady=2)


        input_field_2 = Entry(input_frame_2, font = ('arial', 30, 'bold'), textvariable = input_text_3, width = 30, bg = "#eee", bd = 0, justify = RIGHT)

        input_field_2.grid(row = 0, column = 0,columnspan=3)

        input_field_2.pack(ipady=2) # 'ipady' is internal padding to increase the height of input field


        Label2 = Label(newwin3,textvariable=input_text_4)

        Label2.pack()


        btns_frame_2 = Frame(newwin3, width = 50, height = 330, bg = "grey",bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

        btns_frame_2.pack(side=LEFT,ipadx=0)


        clear_2 = Button(btns_frame_2, text = "Clear", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_clear_2()).grid(row = 0, column = 0, columnspan = 1, padx = 3, pady = 1)

        submit_bt_1 = Button(btns_frame_2, text = "Submit", fg = "black", width = 23, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: Change_it_2()).grid(row = 0, column = 1, columnspan = 2, padx = 0, pady = 1)


        seven_2 = Button(btns_frame_2, text = "7", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(7)).grid(row = 1, column = 0, padx = 1, pady = 1)

        eight_2 = Button(btns_frame_2, text = "8", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(8)).grid(row = 1, column = 1, padx = 1, pady = 1)
```

```
        nine_2 = Button(btns_frame_2, text = "9", fg = "black", width = 11, height = 3, bd = 0,
bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(9)).grid(row = 1, column = 2,
padx = 1, pady = 1)


        four_2 = Button(btns_frame_2, text = "4", fg = "black", width = 11, height = 3, bd = 0,
bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(4)).grid(row = 2, column = 0,
padx = 1, pady = 1)

        five_2 = Button(btns_frame_2, text = "5", fg = "black", width = 11, height = 3, bd = 0, bg
= "#fff", cursor = "hand2", command = lambda: btn_click_2(5)).grid(row = 2, column = 1, padx
= 1, pady = 1)

        six_2 = Button(btns_frame_2, text = "6", fg = "black", width = 11, height = 3, bd = 0, bg
= "#fff", cursor = "hand2", command = lambda: btn_click_2(6)).grid(row = 2, column = 2, padx
= 1, pady = 1)


        one_2 = Button(btns_frame_2, text = "1", fg = "black", width = 11, height = 3, bd = 0, bg
= "#fff", cursor = "hand2", command = lambda: btn_click_2(1)).grid(row = 3, column = 0, padx
= 1, pady = 1)

        two_2 = Button(btns_frame_2, text = "2", fg = "black", width = 11, height = 3, bd = 0, bg
= "#fff", cursor = "hand2", command = lambda: btn_click_2(2)).grid(row = 3, column = 1, padx
= 1, pady = 1)

        three_2 = Button(btns_frame_2, text = "3", fg = "black", width = 11, height = 3, bd = 0,
bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(3)).grid(row = 3, column = 2,
padx = 1, pady = 1)


        astk_2 = Button(btns_frame_2, text = "*", fg = "black", width = 11, height = 3, bd = 0, bg
= "#fff", cursor = "hand2", command = lambda: btn_click_2("*")).grid(row = 4, column =
0,padx = 1, pady = 0)

        zero_2 = Button(btns_frame_2, text = "0", fg = "black", width = 11, height = 3, bd = 0,
bg = "#fff", cursor = "hand2", command = lambda: btn_click_2(0)).grid(row = 4, column =
1,padx = 1, pady = 0)

        hstg_2 = Button(btns_frame_2, text = "#", fg = "black", width = 11, height = 3, bd = 0,
bg = "#fff", cursor = "hand2", command = lambda: btn_click_2("#")).grid(row = 4, column = 2,
padx = 1, pady = 0)



def Program_Keyboard():

        global expression_3

        global dirname

        expression_3 = ""
```

```
        dirname=""
        input_text_6.set("Enter name of the person and Press Submit")
        input_text_5.set("firstname_lastname")
        newwin4 = Toplevel(window)
        newwin4.geometry("645x260")
    #newwin4.resizable(0,0)
        input_frame_3 = Frame(newwin4, width = 80, height = 20, bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)
        input_frame_3.pack(side = TOP,ipady=2)


        input_field_3 = Entry(input_frame_3, font = ('arial', 30, 'bold'), textvariable = input_text_5, width = 30, bg = "#eee", bd = 0, justify = RIGHT)
        input_field_3.grid(row = 0, column = 0,columnspan=8)
        input_field_3.pack(ipady=2) # 'ipady' is internal padding to increase the height of input field


        Label3 = Label(newwin4,textvariable=input_text_6)
        Label3.pack()


        btns_frame_3 = Frame(newwin4, width = 80, height = 100, bg = "grey",bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)
        btns_frame_3.pack(side=LEFT,ipadx=1)


        clear_3 = Button(btns_frame_3, text = "Clear", fg = "black", width = 49, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_clear_3()).grid(row = 0, column = 0, columnspan = 5, padx = 0, pady = 1)
        submit_bt_3 = Button(btns_frame_3, text = "Submit", fg = "black", width = 39, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: submit_keyboard()).grid(row = 0, column = 5, columnspan = 4, padx = 0, pady = 1)
        ## First row
        q_3 = Button(btns_frame_3, text = "q", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('q')).grid(row = 1, column = 0, padx = 1, pady = 1)
```

```
        w_3 = Button(btns_frame_3, text = "w", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('w')).grid(row = 1, column = 1, padx = 1, pady = 1)

        e_3 = Button(btns_frame_3, text = "e", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('e')).grid(row = 1, column = 2, padx = 1, pady = 1)

        r_3 = Button(btns_frame_3, text = "r", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('r')).grid(row = 1, column = 3, padx = 1, pady = 1)

        t_3 = Button(btns_frame_3, text = "t", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('t')).grid(row = 1, column = 4, padx = 1, pady = 1)

        y_3 = Button(btns_frame_3, text = "y", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('y')).grid(row = 1, column = 5, padx = 1, pady = 1)

        u_3 = Button(btns_frame_3, text = "u", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('u')).grid(row = 1, column = 6, padx = 1, pady = 1)

        i_3 = Button(btns_frame_3, text = "i", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('i')).grid(row = 1, column = 7, padx = 1, pady = 1)

        o_3 = Button(btns_frame_3, text = "o", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('o')).grid(row = 1, column = 8, padx = 1, pady = 1)




        ## Second Row

        p_3 = Button(btns_frame_3, text = "p", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('p')).grid(row = 2, column = 0, padx = 1, pady = 1)

        a_3 = Button(btns_frame_3, text = "a", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('a')).grid(row = 2, column = 1, padx = 1, pady = 1)

        s_3 = Button(btns_frame_3, text = "s", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('s')).grid(row = 2, column = 2, padx = 1, pady = 1)

        d_3 = Button(btns_frame_3, text = "d", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('d')).grid(row = 2, column = 3, padx = 1, pady = 1)
```

```
        f_3 = Button(btns_frame_3, text = "f", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('f')).grid(row = 2, column = 4, padx = 1, pady = 1)

        g_3 = Button(btns_frame_3, text = "g", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('g')).grid(row = 2, column = 5, padx = 1, pady = 1)

        h_3 = Button(btns_frame_3, text = "h", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('h')).grid(row = 2, column = 6, padx = 1, pady = 1)

        j_3 = Button(btns_frame_3, text = "j", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('j')).grid(row = 2, column = 7, padx = 1, pady = 1)

        k_3 = Button(btns_frame_3, text = "k", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('k')).grid(row = 2, column = 8, padx = 1, pady = 1)


        ## Third Row

        l_3 = Button(btns_frame_3, text = "l", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('l')).grid(row = 3, column = 0, padx = 1, pady = 1)

        z_3 = Button(btns_frame_3, text = "z", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('z')).grid(row = 3, column = 1, padx = 1, pady = 1)

        x_3 = Button(btns_frame_3, text = "x", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('x')).grid(row = 3, column = 2, padx = 1, pady = 1)

        c_3 = Button(btns_frame_3, text = "c", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('c')).grid(row = 3, column = 3, padx = 1, pady = 1)

        v_3 = Button(btns_frame_3, text = "v", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('v')).grid(row = 3, column = 4, padx = 1, pady = 1)

        b_3 = Button(btns_frame_3, text = "b", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('b')).grid(row = 3, column = 5, padx = 1, pady = 1)

        n_3 = Button(btns_frame_3, text = "n", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('n')).grid(row = 3, column = 6, padx = 1, pady = 1)

        m_3 = Button(btns_frame_3, text = "m", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('m')).grid(row = 3, column = 7, padx = 1, pady = 1)
```

```
        underscore_3 = Button(btns_frame_3, text = "_", fg = "black", width = 9, height = 2, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click_3('_')).grid(row = 3, column = 8, padx = 1, pady = 1)









##

expression = ""

expression_1 = ""

expressionn_2 = ""

expression_3 = ""

dirname = ""

input_text = StringVar()

input_text_1 = StringVar()

input_text_2 = StringVar()

input_text_3 = StringVar()

input_text_4 = StringVar()

input_text_5 = StringVar()

input_text_6 = StringVar()









fun_frame = Frame(window, width = 100, height = 375, bg = "grey",bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

fun_frame.pack(side=LEFT,ipady=0)

# first row
```

```
Camera = Button(fun_frame, text = "Camera", fg = "black", width = 20, height = 3, bd = 0, bg = "#fff",activebackground="green",cursor = "hand2", command = lambda: detect(lckdwn)).grid(row = 0, column = 0, columnspan = 1, padx = 1, pady = 1)

Lock = Button(fun_frame, text = "Full Lock", fg = "black", width = 20, height = 3, bd = 0, bg = "#fff" ,activebackground="green", cursor = "hand2", command = lambda: btn_Lock(expression)).grid(row = 1, column = 0, padx = 1, pady = 1,columnspan=1)

SLock = Button(fun_frame, text = "System Lock", fg = "black", width = 20, height = 3, bd = 0, bg = "#fff",activebackground="green", cursor = "hand2", command = lambda: btn_SystemLock(expression)).grid(row = 2, column = 0, padx = 1, pady = 1,columnspan=1)

Pchange = Button(fun_frame, text = "Password Change", fg = "black", width = 20, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: Pass_window()).grid(row = 3, column = 0, padx = 1, pady = 1,columnspan=1)

Program = Button(fun_frame, text = "Program", fg = "black", width = 20, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: Program_Keyboard()).grid(row = 4, column = 0,padx = 1, pady = 1,columnspan=1)

logout = Button(fun_frame, text = "Logout",fg = "black", width = 20, height = 3, bg = "#fff",bd=0,command=window.quit).grid(row = 5, column = 0, columnspan = 1, padx = 1, pady = 1)


input_frame = Frame(window, width = 70, height = 30, bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

input_frame.pack(side = TOP,ipady=2)



# creating a input field inside the 'Frame'

input_field = Entry(input_frame, font = ('arial', 30, 'bold'), textvariable = input_text, width = 70, bg = "#eee", bd = 0, justify = RIGHT)

input_field.grid(row = 0, column = 0,columnspan=3)

input_field.pack(ipady=2) # 'ipady' is internal padding to increase the height of input field



# creating another 'Frame' for the button below the 'input_frame'

btns_frame = Frame(window, width = 70, height = 330, bg = "grey",bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

btns_frame.pack(side=LEFT,ipadx=50)


# first row
```

```
clear = Button(btns_frame, text = "Clear", fg = "black", width = 30, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_clear()).grid(row = 0, column = 0, columnspan = 3, padx = 3, pady = 0)


# second row

seven = Button(btns_frame, text = "7", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(7)).grid(row = 1, column = 0, padx = 1, pady = 1)

eight = Button(btns_frame, text = "8", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(8)).grid(row = 1, column = 1, padx = 1, pady = 1)

nine = Button(btns_frame, text = "9", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(9)).grid(row = 1, column = 2, padx = 1, pady = 1)


# third row

four = Button(btns_frame, text = "4", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(4)).grid(row = 2, column = 0, padx = 1, pady = 1)

five = Button(btns_frame, text = "5", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(5)).grid(row = 2, column = 1, padx = 1, pady = 1)

six = Button(btns_frame, text = "6", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(6)).grid(row = 2, column = 2, padx = 1, pady = 1)


# fourth row

one = Button(btns_frame, text = "1", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(1)).grid(row = 3, column = 0, padx = 1, pady = 1)

two = Button(btns_frame, text = "2", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(2)).grid(row = 3, column = 1, padx = 1, pady = 1)

three = Button(btns_frame, text = "3", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(3)).grid(row = 3, column = 2, padx = 1, pady = 1)



# fourth row
```

```
astk = Button(btns_frame, text = "*", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click("*")).grid(row = 4, column = 0,padx = 1, pady = 0)

zero = Button(btns_frame, text = "0", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click(0)).grid(row = 4, column = 1,padx = 1, pady = 0)

hstg = Button(btns_frame, text = "#", fg = "black", width = 11, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn_click("#")).grid(row = 4, column = 2, padx = 1, pady = 0)


window.mainloop()
```

Code 2: Detection.py

```
import face_recognition

import cv2

import os

from datetime import datetime

# Get a reference to webcam #0 (the default one)


def intrusion_fn(detect1):

    x = 0 labels =

    [] check = []

    image_pic = []

    known_face_encodings=[]

    known_face_names=[]


    for root, dirs, files in os.walk("IntrusionDetected"):

        num = len(files)


    for root, dirs, files in os.walk("Images"):

        for file in files:
```

```
        if file.endswith("jpg") or file.endswith("JPG"):
            path = os.path.join(root, file)
            label = os.path.basename(file).replace(" ", "-").lower()
            labels.append(label)
    for i in range (0,len(labels)):
        image_pic.append(face_recognition.load_image_file(str("Images/")+str(labels[i])))
    for i in range (0,len(labels)):
        known_face_encodings.append(face_recognition.face_encodings(image_pic[i])[0])
        known_face_names.append(labels[i][0:-4])

    # Initialize some variables
    face_locations = []
    face_encodings = []
    face_names = []
    process_this_frame = True
    video_capture = cv2.VideoCapture(0)
    while True:
        # Grab a single frame of video
        ret, frame = video_capture.read()

        # Resize frame of video to 1/4 size for faster face recognition processing
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
        rgb_small_frame = small_frame[:, :, ::-1]

        # Only process every other frame of video to save time
        if process_this_frame:
            # Find all the faces and face encodings in the current frame of video
```

```
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # If a match was found in known_face_encodings, just use the first one.
        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]

        face_names.append(name)

process_this_frame = not process_this_frame

# Display the results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    # Scale back up face locations since the frame we detected in was scaled to 1/4 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # Draw a box around the face
    cv2.rectangle(frame, (left, top), (right+35, bottom), (0, 0, 255), 2)

    # Draw a label with a name below the face
```

```
        cv2.rectangle(frame, (left, bottom -35), (right+35, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, str(datetime.now()), (5, 450), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)


    # Display the resulting image
    cv2.imshow('Video', frame)
    # print(face_names)
    item = "Unknown"
    if detect1 == 1:
        if x == 0:
            if len(face_names) != 0 and item not in face_names:
                check = face_names
                x = 1
            elif item in face_names:
                cv2.imwrite(str("IntrusionDetected/Intrusion_") + str(num)+ str(".jpg"), frame)
                num = num+1
        elif x == 1:
            if check == face_names:
                x = 2
            else:
                x = 0
        elif x == 2:
            if check == face_names:
                break
            else:
                x = 0
    # Hit 'q' on the keyboard to quit!
    elif detect1 == 0:
```

```
            if item in face_names:
                cv2.imwrite(str("IntrusionDetected/Intrusion_")+str(num)+str(".jpg"),frame)
                num = num+1
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()
```

Code 3: save_photo.py

```
import numpy as np
import cv2

def go_snap(name_label):
    cap = cv2.VideoCapture(0)
    while cap.isOpened():
        [ok, frame] = cap.read()
        frame = cv2.flip(frame, 1)
        cv2.imshow('Photo', frame)

        key = cv2.waitKey(1)
        if key == ord('p'):
            cv2.imwrite(str("Images/")+str(name_label)+str(".jpg"), frame)
            break
    cv2.destroyAllWindows()
```