Marathwada Shikshan Prasarak Mandal’s

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

**Project Report**

**on**

**Automatic Door Unlock System Using Face Recognition**

Submitted By

**Krushna Kulkarni (36122)**

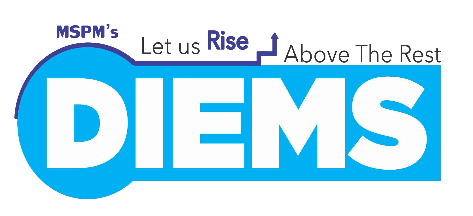
for

**Continuous Assessment of**

**Machine Learning (TY CSE)**

**Dr. Babasaheb Ambedkar Technological University**

**Lonere (M.S.)**

****

Department of Computer Science and Engineering

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

(2020- 2021)

**Project Report**

**on**

**Automatic Door Unlock System Using Face Recognition**

Submitted By

**Krushna Kulkarni (36122)**

**Ganesh Kekan (36129)**

**Vaibhav Lale (36158)**

**In partial fulfillment of**

**Bachelor of Engineering**

**(Computer Science & Engineering)**

Guided By

**Ms. Sugandha Nandedkar**

Department of Computer Science & Engineering

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

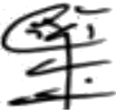
(2020- 2021)

**CERTIFICATE**

This is to certify that, the Project entitled “**Automatic Door Unlock System Using Face Recognition**” submitted by **Krushna Kulkarni (36122) is** a bonafide work completed under my supervision and guidance in partial fulfillment for award of Bachelor of Engineering (Computer Science and Engineering) Degree of Dr. Babasaheb Ambedkar Technological University, Lonere.

Place: Aurangabad

Date: 22/12/2020

****

**Ms. Sugandha Nandedkar Mr. Sanjay B. Kalyankar**

**Guide Head**

****

**Dr. Ulhas D. Shiurkar**

**Director,**

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

|  |  |  |
| --- | --- | --- |
| Contents | | |
|  | List of Figures | i |
|  | List of Screen Shots | ii |
| 1. | INTRODUCTION  * 1. Introduction   2. Problem Statement   3. Objectives | 1 |
| 2. | DATA COLLECTION AND TRAINING **2.1** Dataset  **2.2** Features  2.3 Training | 2 |
| 3. | FINAL DESIGN AND IMPLEMENTATION | 10 |
| 4. | PERFORMANCE ANALYSIS | 15 |
| 5. | CONCLUSION AND FUTURE SCOPE  * 1. Conclusion   2. Future Scope | 16 |

|  |  |
| --- | --- |
| **List of Figures**  **Figure 2.1: Haar Cascade Overview**  **Figure 2.2: Theoretical Face Model using Haar Features**  **Figure 2.3: LBP Operation**  **Figure 2.4: Bilinear Interpolation**  **Figure 2.5: Histogram Formation from LBP** |  |

**List of Screen Shots**

**Screenshot 2.1: Overview of dataset**

**Screenshot 2.1: Overview of User Dataset Training**

**Screenshot 3.1: Home Page of Application**

**Screenshot 3.2: Admin Portal of Application**

**Screenshot 3.4: New User Registration Page of Application**

**Screenshot 3.5: Process of Capturing Images and Creation of Dataset**

**Screenshot 3.6: Model Training Successful**

**Screenshot 3.7 Authorized Person Recognized**

**Screenshot 3.8 Authorized Person Recognized Notification**

###### 1. INTRODUCTION

* 1. Introduction

Face is the representation of one’s unique identity. So here we are proposing an application using face recognition for purpose of Automatic door unlocking.

* 1. Problem statement

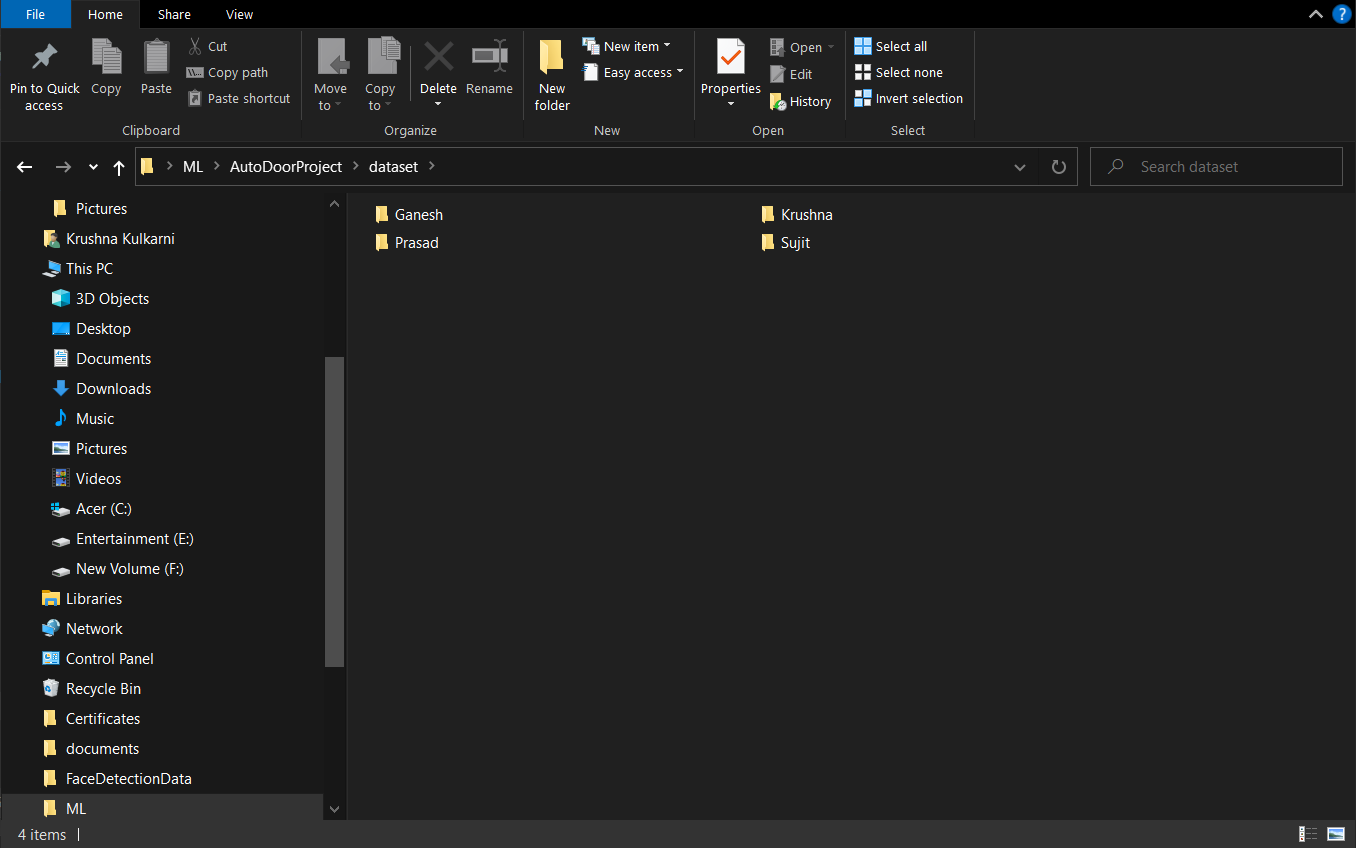
An old woman lives alone in her house. Beside the main door she is having a small window through which she can talk with the person standing at her house footsteps. Her hands are shaking so badly that she cannot open the door immediately. Can you help her by designing such a system where the door will be opened only for her maid? Other than her maid if somebody else is standing behind the door whom she finds as a trustworthy person then also the door should be opened.

* 1. Objectives

1. Door should be unlocked automatically.
2. Door should be unlocked when there is authorized person on doorsteps.
3. Visitors entry data should be recorded.

**2. DATA COLLECTION AND TRAINING**

* 1. Dataset:

 **Screenshot 2.1: Overview of dataset**

Details:

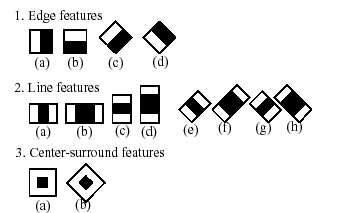
Whenever registering as a new authorized user, 300 images of user will be captured, which forms our dataset

We are using Haar Cascades Classifier for face detection and and LBPH algorithm for recognition

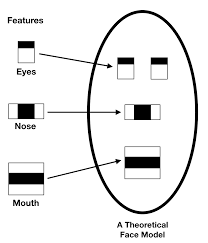
Haar cascades is a machine learning based approach where a cascade function is trained with a set of input data.

Haar Cascade is a machine learning object detection classifier used to identify objects in an image or video.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other image



**Figure 2.1: Haar Cascade Overview**



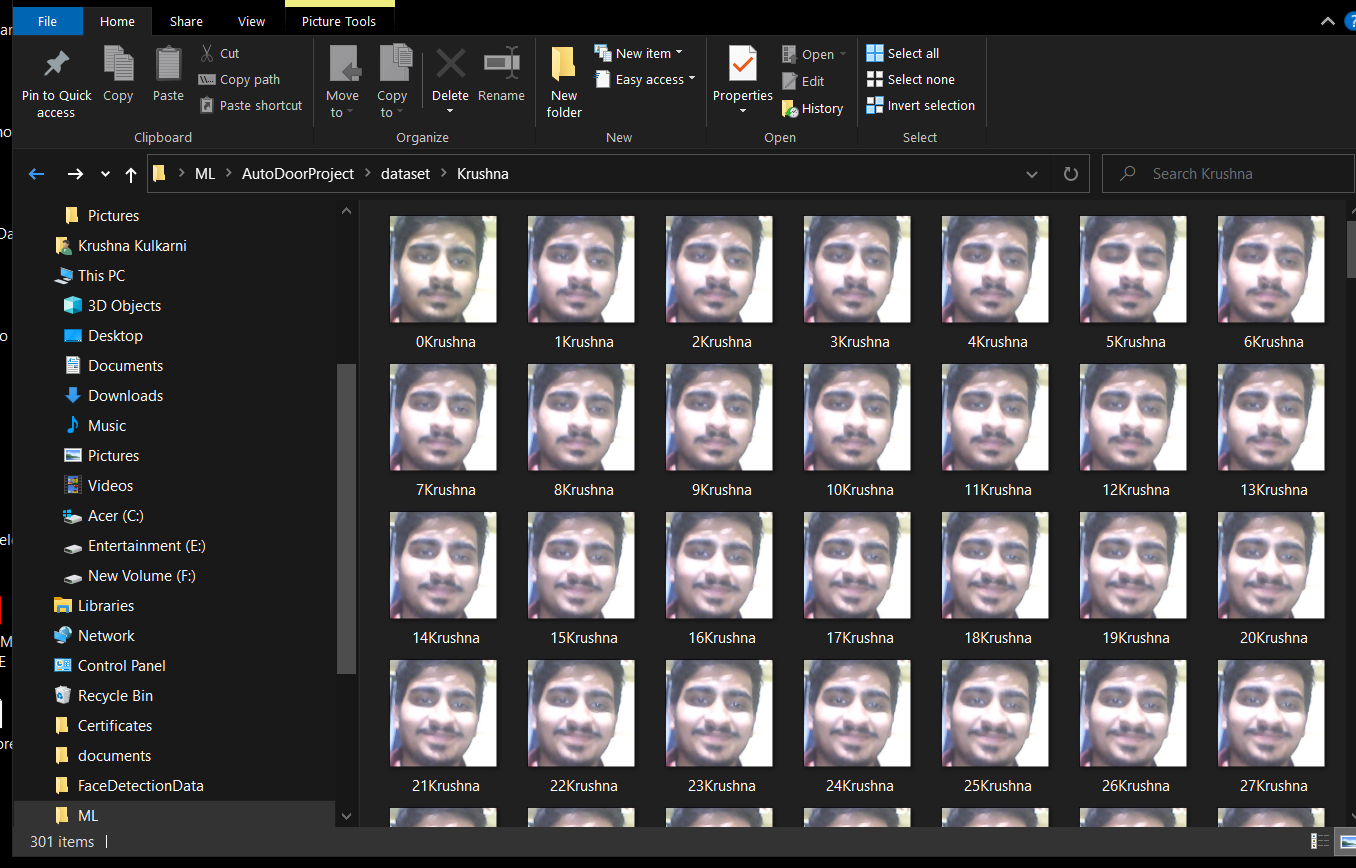
**Figure 2.2: Theoretical Face Model using Haar Features**

* It is a classifier in which the cascade function is trained by superimposing the positive images over set of negative images
* It uses HAAR Features and Integral Images.
* It helps in the stages of Face Detection and Feature Extraction.
  1. Features:

1. We can create as many users as we want.
2. Each authorized person will have separate set of images as a part of dataset.
3. Name of the authorized user will be stored in an excel file.

* 1. Training:

1. For the purpose of training we are using LBPH (Local Binary Pattern Histogram) algorithm.
2. Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.
3. LBPH is one of the easiest face recognition algorithms. It can represent local features in the images. It is possible to get great results (mainly in a controlled environment). It is robust against monotonic grey scale transformations. It is provided by the OpenCV library (Open Source Computer Vision Library).



**Screenshot 2.1: Overview of User Dataset Training**

There are 5 stages for whole Process

**1.** **Parameters**: the LBPH uses 4 parameters:

* Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
* Neighbours: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
* Grid X and Y: the number of cells in the horizontal and vertical direction respectively. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.

**2. Applying the LBP operation**: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbours**.

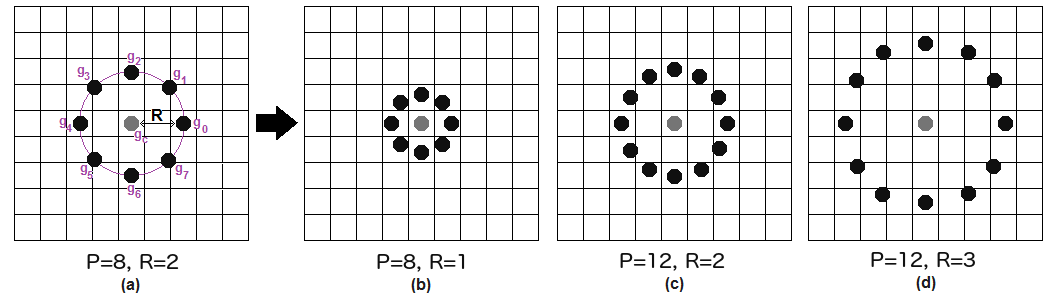
The image below shows this procedure:



**Figure 2.3: LBP Operation**

Based on the image above, let’s break it into several small steps so we can understand it easily:

* Suppose we have a facial image in grayscale.
* We can get part of this image as a window of 3x3 pixels.
* It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
* Then, we need to take the central value of the matrix to be used as the threshold.
* This value will be used to define the new values from the 8 neighbours.
* For each neighbour of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
* Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
* Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
* At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
* **Note**: The LBP procedure was expanded to use a different number of radius and neighbours, it is called Circular LBP.

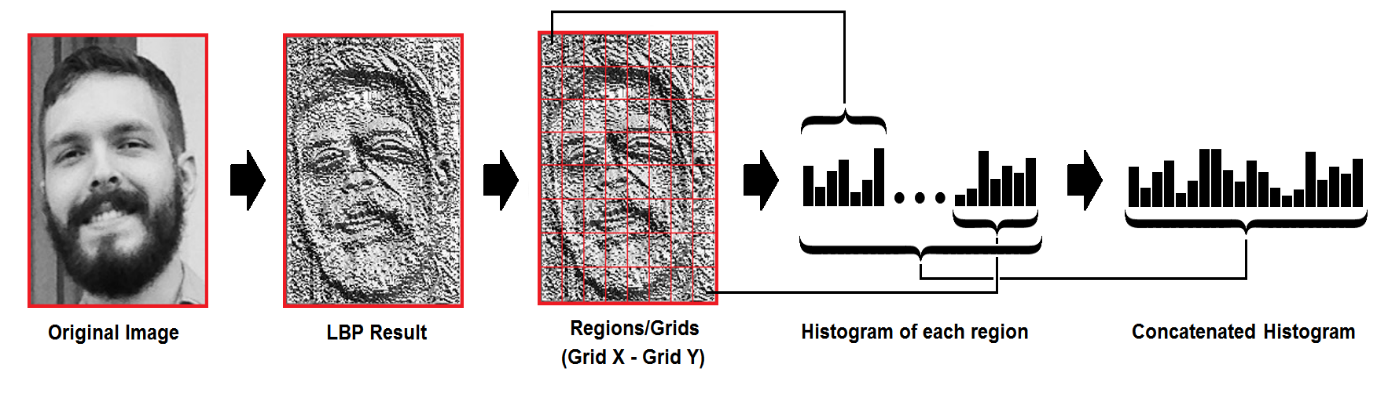


**Figure 2.4: Bilinear Interpolation**

It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest

pixels (2x2) to estimate the value of the new data point.

**4. Extracting the Histograms**: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:



**Figure 2.5: Histogram Formation from LBP**

Based on the image above, we can extract the histogram of each region as follows:

As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.

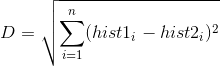
Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

* **Performing the face recognition**: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **Euclidean distance**, **chi-square**, **absolute value**, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:



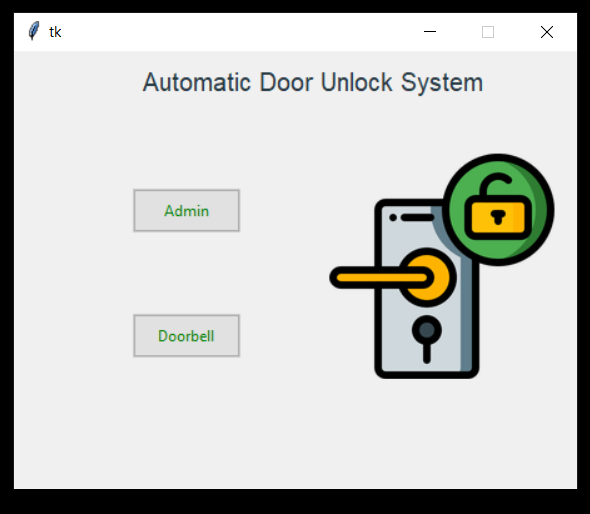
* So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a ‘**confidence**’ measurement. **Note**: don’t be fooled about the ‘confidence’ name, as lower confidences are better because it means the distance between the two histograms is closer.
* We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

**3. FINAL DESIGN AND IMPLEMENTATION**

We created A Desktop Application for the implementation of the project.

There are 2 options for user on Home.

* + - 1. Admin
      2. Doorbell

****

**Screenshot 3.1: Home Page of Application**

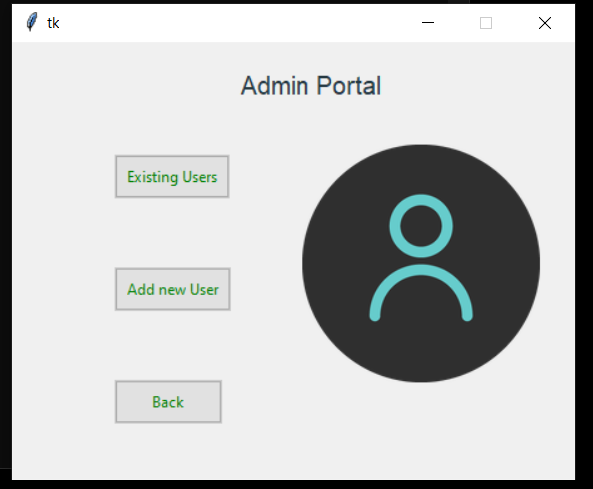
The Admin Portal have 3 Options

1. Existing Users:

It returns the list of Authorized Persons.

1. Add new User:

For creating new user. And training the model.

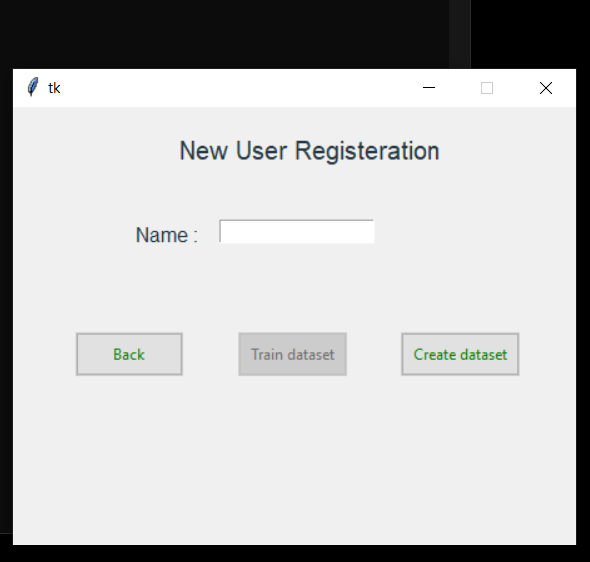
****

**Screenshot 3.2: Admin Portal of Application**

The new user registration for the purpose of authorization:

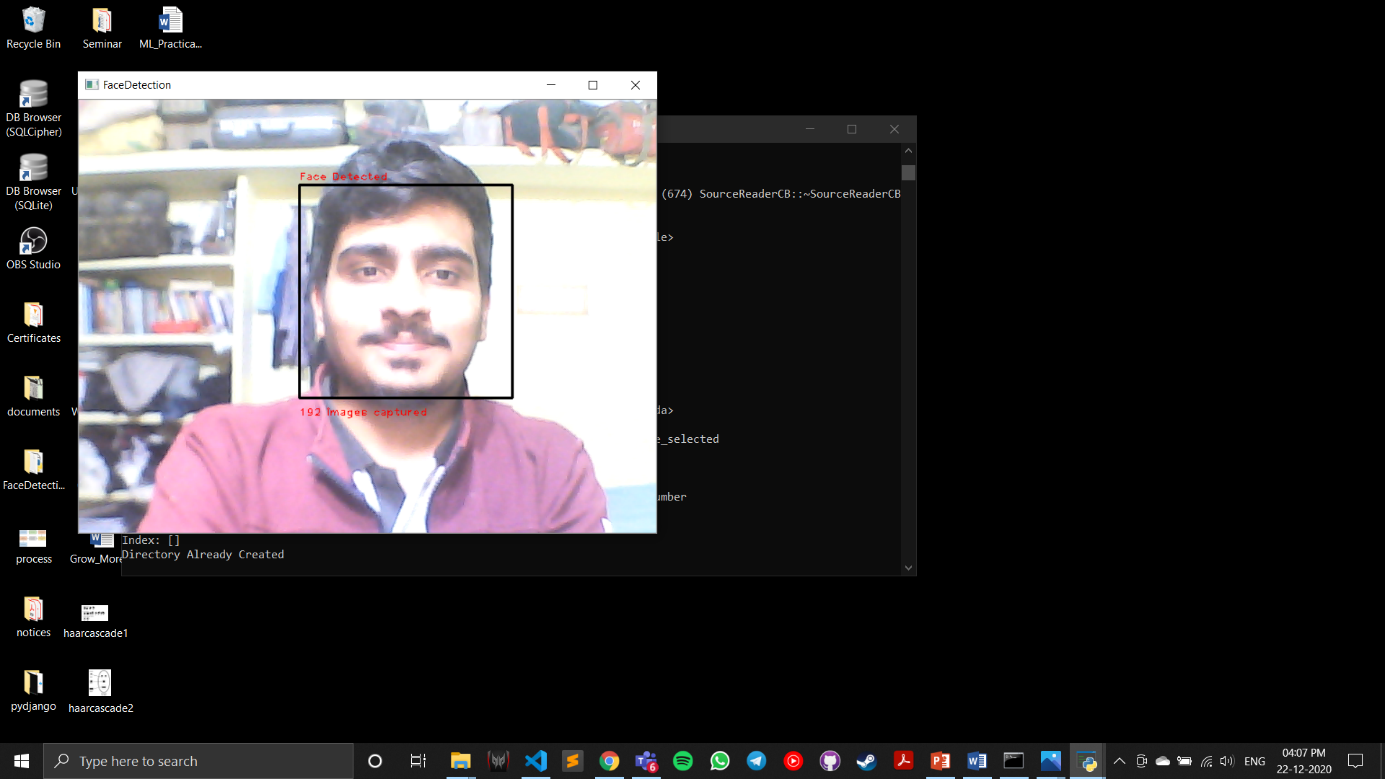
While adding the new user we need to enter name and create the dataset of the images of that person.

.

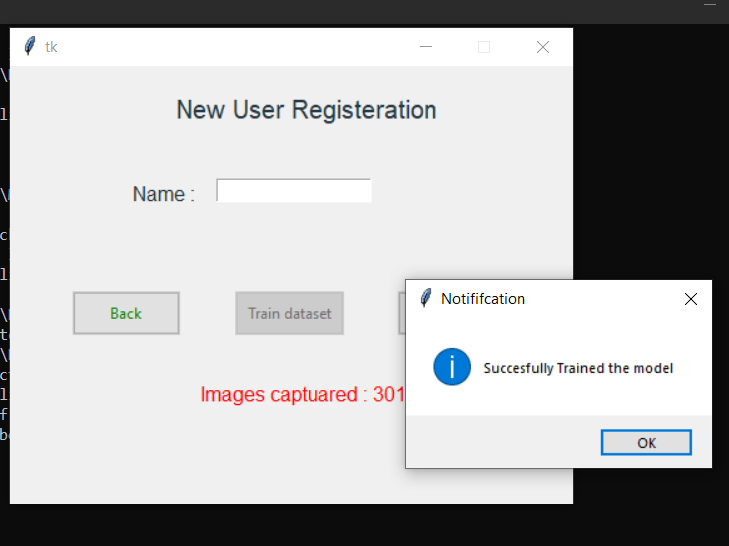
****

**Screenshot 3.4: New User Registration Page of Application**

When you enter name and click on ‘Create dataset’ option, camera application starts and captures the 300 images of that person

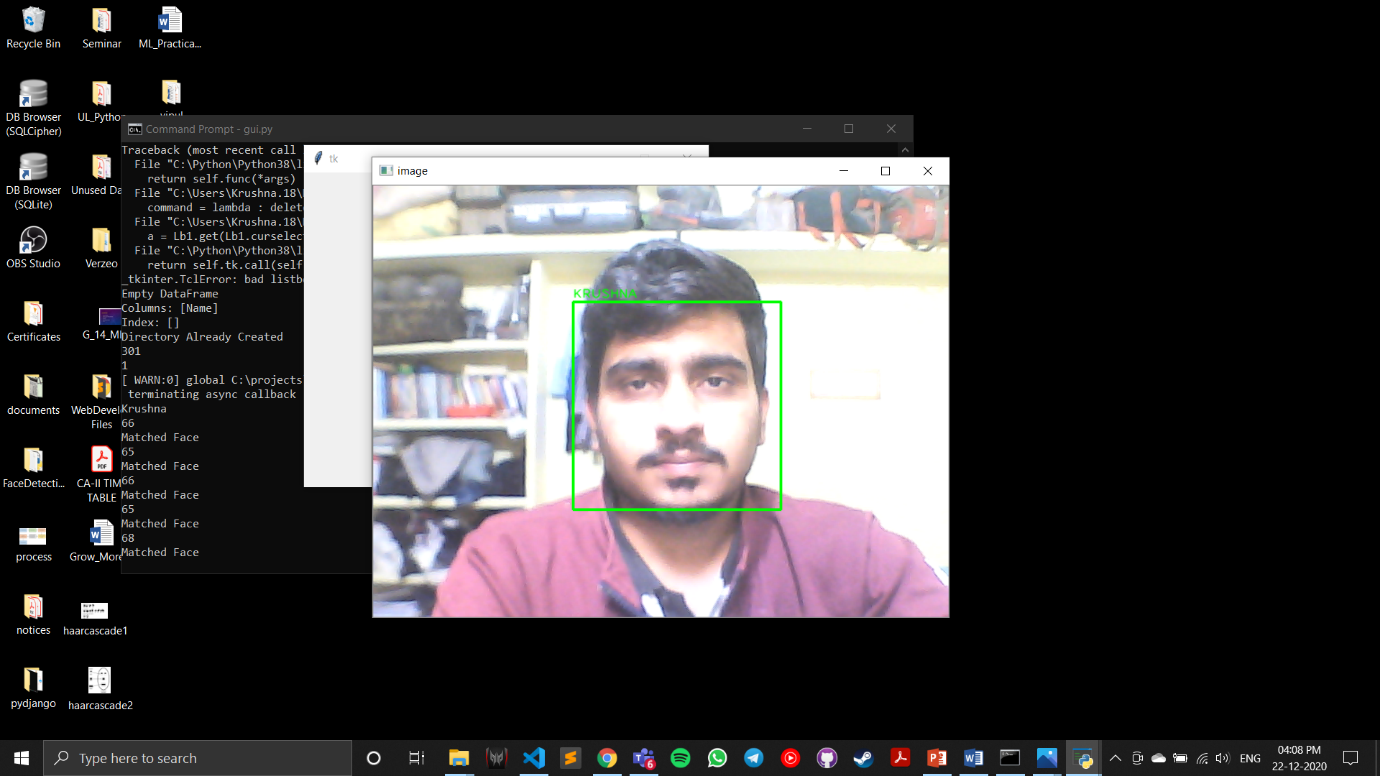
****

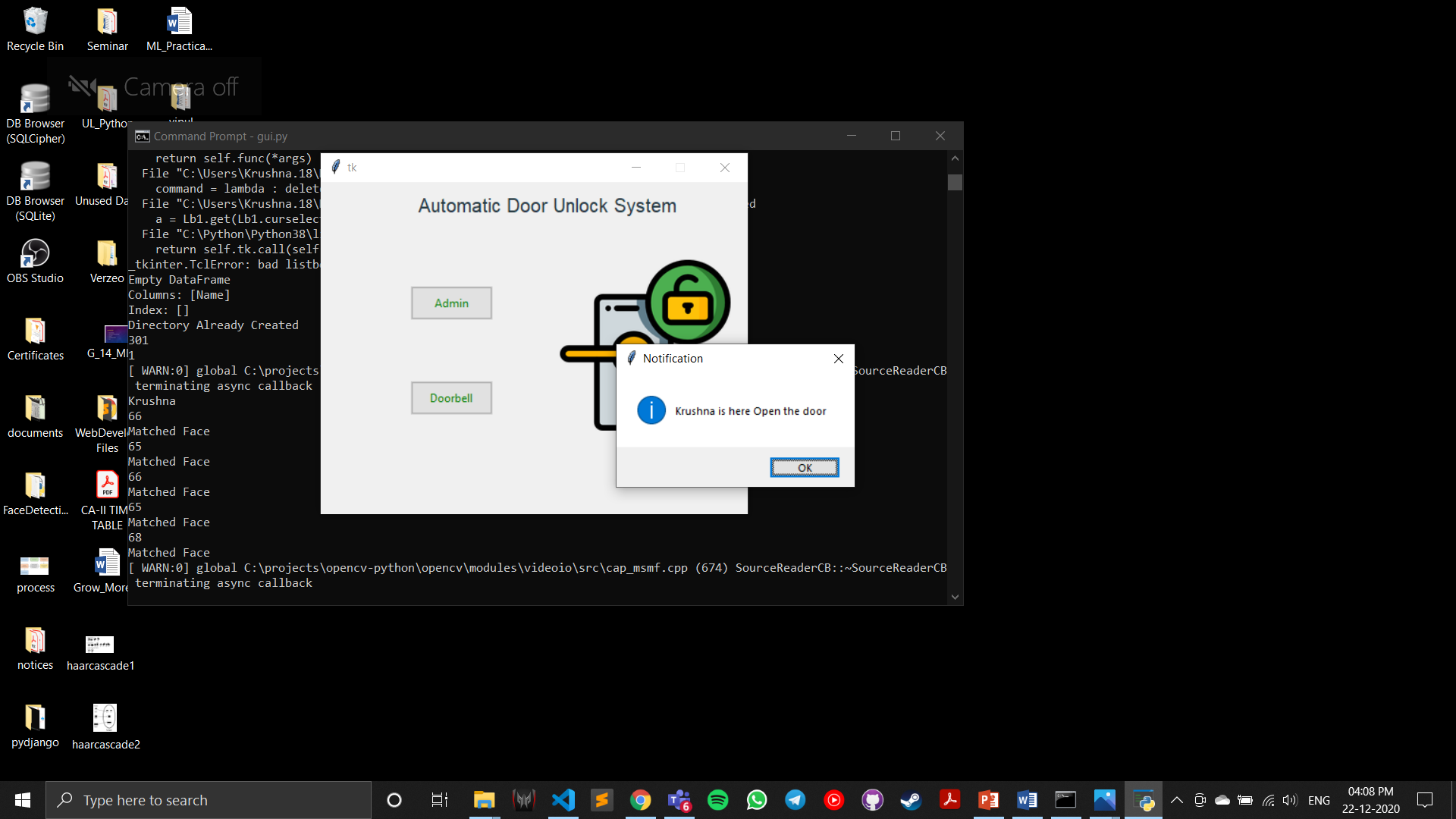
**Screenshot 3.5: Process of Capturing Images and Creation of Dataset**

**** After dataset is created, train the model by clicking on ‘Train dataset’ option.

**Screenshot 3.6: Model Training Successful**

Now when the doorbell is pressed, camera will start and will detect the face of person. If the person is recognized from the dataset we created it will show the notification as bellow and door will be unlocked.

****

** Screenshot 3.7 Authorized Person Recognized**

**Screenshot 3.8 Authorized Person Recognized Notification**

1. **PERFORMANCE ANALYSIS**

* While testing the application with authorized person’s face it detected and recognized user with rate of 100%
* In good lighting conditions (Daylight) it gave recognition with accuracy of 92.77 %
* While in low light conditions it gave recognition with accuracy of 77%
* Although the profile is just for frontal faces it can detect accurately for slight head tilts (about 15 -20 degrees) and at 200 cms distance from webcam.
* The accuracy varies between lighting conditions and the quality of the webcam used.
* For recognition with even greater confidence, the equipment or environment (web camera and the room light/natural light) should be higher quality.

1. **CONCLUSION AND FUTURE SCOPE**

5.1 Conclusion

* The Accuracy of for the algorithm is very high, which shows that the LBPH classifier is an efficient and accurate face recognizer
* The algorithm is able to provide access control during different lighting conditions (during day and night) because Haar Cascades frontal profile.
* Although the profile is just for frontal faces it can detect accurately for slight head tilts (about 15 -20 degrees) and at 200 cms distance from webcam.
* The accuracy between lighting conditions and the quality of the webcam used.
* For recognition with even greater confidence, the equipment or environment (web camera and the room light/natural light) should be higher quality.

5.2 Future Scope

* This project can be integrated with raspberry pi for actual implementation on the door.
* This project can be used in drone cams for criminal detection in crowds.
* This project can be implemented over large sectors.
* Can be used for attendance purpose of students.
* Can be used for employee recognition in corporate sectors.
* Can be used to detect security flaws and trespassers.

**Project Code**

import tkinter as tk

from tkinter import messagebox

from tkinter import ttk

from PIL import ImageTk, Image

import pytesseract

import cv2

import numpy as np

#import face\_recognition

import pandas as pd

import os

import csv

import time

import datetime

import sys

import shutil

#from Detector import test\_model

def startpage(container):

label = tk.Label(container, text ="Automatic Door Unlock System", font = "Helvetica", foreground="#263942")

label.config(font=("Helvetica", 15))

label.place(x = 100,y = 10)

def admin\_clear\_frame(frame):

for widget in frame.winfo\_children():

widget.destroy()

admin(frame)

# opens the image

img = Image.open('static/door.png')

img = img.resize((180, 180), Image.ANTIALIAS)

# PhotoImage class is used to add image to widgets, icons etc

img = ImageTk.PhotoImage(img)

# create a label

panel = tk.Label(container, image = img)

# set the image as img

panel.image = img

panel.place(x = 250 , y = 80)

ttk.Style().configure("TButton", padding=6, relief="flat",

background="#ccc",foreground='green')

button1 = ttk.Button(container, text ="Admin",command = lambda : admin\_clear\_frame(container))

button1.place(x = 95, y = 110)

button2 = ttk.Button(container, text ="Doorbell",command = lambda : doorbell())

button2.place(x = 95,y = 210)

def admin(container):

label = tk.Label(container, text ="Admin Portal", font = "Helvetica", foreground="#263942")

label.config(font=("Helvetica", 15))

label.place(x = 180,y = 20)

img = Image.open('static/login.png')

img = img.resize((190, 190), Image.ANTIALIAS)

img = ImageTk.PhotoImage(img)

panel = tk.Label(container, image = img)

panel.image = img

panel.place(x = 230 , y = 80)

ttk.Style().configure("TButton", padding=6, relief="flat",

background="#ccc",foreground='green')

def user\_list\_clear\_frame(frame):

for widget in frame.winfo\_children():

widget.destroy()

user\_list(frame)

def new\_user\_clear\_frame(frame):

for widget in frame.winfo\_children():

widget.destroy()

new\_user(frame)

def back\_menu(frame):

for widget in frame.winfo\_children():

widget.destroy()

startpage(frame)

button1 = ttk.Button(container, text ="Existing Users",command = lambda : user\_list\_clear\_frame(container))

button1.place(x = 82, y = 90)

button2 = ttk.Button(container, text ="Add new User",command = lambda : new\_user\_clear\_frame(container))

button2.place(x = 82,y = 180)

button3 = ttk.Button(container, text ="Back",command = lambda : back\_menu(container))

button3.place(x = 82,y = 270)

def new\_user(container):

new\_user = tk.StringVar()

flag = tk.IntVar()

flag.set(0)

num\_images = tk.IntVar()

label = tk.Label(container, text ="New User Registeration", font = "Helvetica", foreground="#263942")

label.config(font=("Helvetica", 15))

label.place(x = 130,y = 20)

name\_label = tk.Label(container, text ="Name :", font = "Helvetica", foreground="#263942")

name\_label.config(font=("Helvetica", 12))

name\_label.place(x = 95,y = 90)

def clear(frame):

for widget in frame.winfo\_children():

widget.destroy()

admin(frame)

def check(container,name,flag,button1,button2,button3,num\_images):

data = pd.read\_csv('User.csv')

print(data)

if(name in list(data.Name)):

messagebox.showerror("Error","User already Exists")

return

create\_dataset(container,name,flag,button1,button2,button3,num\_images)

return

def build\_model(name,button1,button2,button3):

entry\_name.delete(0,'end')

train\_model(name,button1,button2,button3)

entry\_name = tk.Entry(container,textvariable = new\_user)

entry\_name.place(x = 165, y = 90)

ttk.Style().configure("TButton", padding=6, relief="flat",

background="#ccc",foreground='green')

button3 = ttk.Button(container, text ="Back",command = lambda : clear(container),state = tk.NORMAL)

button2 = ttk.Button(container, text ="Train dataset",state = tk.DISABLED,command = lambda : build\_model(new\_user.get(),button1,button2,button3))

button1 = ttk.Button(container, text ="Create dataset",command = lambda : check(container,new\_user.get(),flag,button1,button2,button3,num\_images))

button1.place(x = 310, y = 180)

button2.place(x = 180,y = 180)

button3.place(x = 50,y = 180)

def create\_dataset(container,name,flag,button1,button2,button3,num\_images):

path = "./dataset/" + name

num\_of\_images = 0

detector = cv2.CascadeClassifier("static/haarcascade\_frontalface\_default.xml")

try:

os.makedirs(path)

except:

print('Directory Already Created')

vid = cv2.VideoCapture(0)

while True:

ret, img = vid.read()

new\_img = None

grayimg = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

face = detector.detectMultiScale(image=grayimg, scaleFactor=1.1, minNeighbors=5)

key = 0

for x, y, w, h in face:

cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 0), 2)

cv2.putText(img, "Face Detected", (x, y-5), cv2.FONT\_HERSHEY\_PLAIN, 0.8, (0, 0, 255))

cv2.putText(img, str(str(num\_of\_images)+" images captured"), (x, y+h+20), cv2.FONT\_HERSHEY\_PLAIN, 0.8, (0, 0, 255))

new\_img = img[y:y+h, x:x+w]

cv2.imshow("FaceDetection", img)

key = cv2.waitKey(1) & 0xFF

try :

cv2.imwrite(str(path+"/"+str(num\_of\_images)+name+".jpg"), new\_img)

num\_of\_images += 1

except :

pass

if num\_of\_images > 300:

break

cv2.destroyAllWindows()

print(num\_of\_images)

button2['state'] = "normal"

button3['state'] = 'disabled'

button1['state'] = 'disabled'

flag.set(1)

num\_images.set(num\_of\_images)

print(flag.get())

app.protocol("WM\_DELETE\_WINDOW",disable\_event)

s = f"Images captuared : {num\_images.get()}"

label1 = tk.Label(container, text = s, font = "Helvetica", foreground="red")

label1.config(font=("Helvetica", 12))

label1.place(x = 150,y = 250)

return

def train\_model(name,button1,button2,button3):

path = os.path.join(os.getcwd()+"/dataset/"+name+"/")

faces = []

ids = []

labels = []

pictures = {}

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

pictures = files

for pic in pictures :

imgpath = path+pic

img = Image.open(imgpath).convert('L')

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

id = int(pic.split(name)[0])

#names[name].append(id)

faces.append(imageNp)

ids.append(id)

ids = np.array(ids)

#Train and save classifier

clf = cv2.face.LBPHFaceRecognizer\_create()

clf.train(faces, ids)

clf.write("./classifiers/"+name+"\_classifier.xml")

button2['state'] = 'disabled'

button3['state'] = "normal"

button1['state'] = "normal"

app.protocol("WM\_DELETE\_WINDOW",close)

data = pd.read\_csv('User.csv')

data.loc[len(data.Name)] = [name]

data.set\_index('Name',inplace=True)

data.to\_csv('User.csv')

messagebox.showinfo("Notififcation","Succesfully Trained the model")

def delete\_selected(frame,Lb1):

a = Lb1.get(Lb1.curselection()).split(' ')

path = os.getcwd()

print("Path : ",path)

path1 = path + f"\\dataset\\{a[1]}"

path2 = path + f'\\classifiers\\{a[1]}\_classifier.xml'

print(path,path1,path2)

shutil.rmtree(path1)

os.remove(path2)

data = pd.read\_csv('User.csv')

print(data)

new\_data = data[data.Name != a[1]]

print(new\_data)

new\_data.set\_index('Name',inplace = True)

print("New Dataset : ",new\_data)

new\_data.to\_csv('User.csv')

for widget in frame.winfo\_children():

widget.destroy()

user\_list(frame)

def user\_list(container):

label = tk.Label(container, text ="List of Existing Users", font = "Helvetica", foreground="#263942")

label.config(font=("Helvetica", 15))

label.place(x = 140,y = 20)

names = []

Lb1 = tk.Listbox(container,selectbackground = "lightblue",yscrollcommand = True,bg = "#ccc")

data = pd.read\_csv('User.csv')

z = list(data.Name)

print(z)

for i in range(len(z)):

Lb1.insert(i+1, f"{i+1}. {z[i]}")

Lb1.place(x = 90,y = 80)

ttk.Style().configure("TButton", padding=6, relief="flat",

background="#ccc",foreground='green')

def back\_clear\_frame(frame):

for widget in frame.winfo\_children():

widget.destroy()

admin(frame)

button1 = ttk.Button(container, text ="Delete",

command = lambda : delete\_selected(container,Lb1))

button1.place(x = 300, y = 120)

button1 = ttk.Button(container, text ="Back",

command = lambda : back\_clear\_frame(container))

button1.place(x = 300, y = 180)

def doorbell():

data = pd.read\_csv("User.csv")

names = list(data.Name)

face\_cascade = cv2.CascadeClassifier('./static/haarcascade\_frontalface\_default.xml')

recognizer = cv2.face.LBPHFaceRecognizer\_create()

cap = cv2.VideoCapture(0)

for i in names:

print(i)

name = i

recognizer.read(f"./classifiers/{name}\_classifier.xml")

pred = 0

for i in range(50):

ret, frame = cap.read()

#default\_img = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray,1.3,5)

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

roi\_gray = gray[y:y+h,x:x+w]

id,confidence = recognizer.predict(roi\_gray)

confidence = 100 - int(confidence)

if confidence > 50:

# if u want to print confidence level

# confidence = 100 - int(confidence)

pred += +1

text = name.upper()

font = cv2.FONT\_HERSHEY\_PLAIN

frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

frame = cv2.putText(frame, text, (x, y-4), font, 1, (0, 255, 0), 1, cv2.LINE\_AA)

print (confidence)

print("Matched Face")

if(pred == 5):

time\_now = datetime.datetime.now()

path = os.getcwd() + f"\\results\\{name}{time\_now}.jpg"

#print(frame)

#print(path)

s = ".\\results\\"+ str(name) + str(time\_now.date()) + "-" + str(time\_now.hour) + "-" +str(time\_now.minute) + "-" +str(time\_now.second)

cv2.imwrite(s+".jpg", frame)

cv2.waitKey(2000)

cap.release()

cv2.destroyAllWindows()

excel\_data = pd.read\_excel('entries.xlsx',engine='openpyxl')

excel\_data.loc[len(excel\_data)] = [name,datetime.datetime.now()]

excel\_data.to\_excel('entries.xlsx',index = False)

messagebox.showinfo("Notification",str(name) + " is here Open the door")

return

else:

#pred += -1

text = "UnknownFace"

font = cv2.FONT\_HERSHEY\_PLAIN

frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

frame = cv2.putText(frame, text, (x, y-4), font, 1, (0, 0,255), 1, cv2.LINE\_AA)

cv2.imshow("image", frame)

if cv2.waitKey(20) & 0xFF == ord('q'):

print(pred)

messagebox.showerror("Error","Unauthorized Person doors are closed")

cap.release()

cv2.destroyAllWindows()

app = tk.Tk()

app.geometry("450x350")

app.resizable(False,False)

container = tk.Frame(app)

container.pack(side = "top", fill = "both", expand = True)

container.grid\_rowconfigure(0, weight = 1)

container.grid\_columnconfigure(0, weight = 1)

startpage(container)

def close():

app.destroy()

def disable\_event():

pass

app.mainloop()

**ACKNOWLEDGEMENT**

We would like to place on record our deep sense of gratitude to Mr. Sanjay Kalyankar, Head of Department Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad, for his generous guidance, help and useful suggestions.

We express our sincere gratitude to Prof. Sugandha Nandedkar, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We are extremely thankful to Dr. Ulhas Shiurkar, Director, Deogiri Institute of Engineering and management Studies Aurangabad, for providing me infrastructural facilities to work in, without which this work would not have been possible.

**Signature(s) of Students**

Krushna Kulkarni