**College of Applied Business**

**(Tribhuvan University)**

**A REPORT ON PYTHON PROJECT**

**(FACIAL RECOGNITION)**

**Submitted by:**

Anil Raj Rimal

Bibhav Pokharel

Pawan Sapkota

Sabin Maharjan

College of Applied Business

Department of Computer Science and Technology

Gangahiti, Kathmandu

**Submitted to:**

College of Applied Business

Department of Computer Science and Technology

Gangahiti, Kathmandu

**ACKNOWLEDGMENT**

"It's not possible to prepare a project report without the assistance and encouragement of other people. This one is certainly no exception."

On the very outset of this report, we would like to extend our sincere and heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, cooperation and encouragement, we would not have made headway in this project.

We are ineffably indebted to Mr. Kamal Acharya, our python instructor, for conscientious guidance and encouragement to accomplish this project.

We extend our gratitude towards College of Applied Business for giving us this opportunity.

We also acknowledge a deep sense of reverence, our gratitude towards our parents and other members of our family, who have helped us morally.

At last, but not least gratitude goes to all of our friends who directly or indirectly helped us to complete this project report.

Any omission in this brief acknowledgement does not mean lack of gratitude.

**ABSTRACT**

The project proposal was given to create any application using python. As we have seen people like doing all stuff all by their own. So, we decided to create a project on AI.

The purpose of this project is to expose the users to overcome their daily workload. They can simply use our program to detect face and take attendance accurately. This project might me helpful for government and non-government offices, schools, hospitals, markets etc. So, we have tried to help whole humanity to decrease their workload.

**Table of Contents**

**Chapter One 1-2**

1. **Project Introduction** 1
2. **Tools Used** 2

**Chapter Two 3**

1. **Flow Chart** 3

**Chapter Three 4-15**

1. **Implementation** **[code]**4-12
2. **Output**…………………………………………………………….13-15

**Conclusion 16**

**Chapter One**

1. **Project Introduction**

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image. To write a code that recognizes faces needs some training data, we should train our machine so that it knows the faces and who are they.

Recognizing people by their faces in pictures and video feeds is seen everywhere starting from social media to phone cameras. A face recognition system is built for matching human faces with a digital image. Ultimately what a computer recognizes is pixel values ranging from 0-255.

In Computer Vision face recognition has been in since ages and has evolved over the years. Many researchers have come up with many new techniques to efficiently identify and tell apart faces. There are many use cases such as authentication and verification of users.

**Face Detection vs. Face Recognition**

These two things might sound very similar but actually, they are not the same. Let’s understand the difference so that we don’t miss the point.

Face Detection is the process of detecting faces, from an image or a video that doesn’t matter. The program doesn’t do anything more than finding the faces. But on the other hand, face recognition, the program that finds the faces and also it can tell which face belongs to whom. So, it is more informational than just detecting them.

1. **Tools Used**

We have to install some libraries so that our program works. Here is a list of the libraries we have installed:

**Cmake:**

Cmake is a prerequisite library so that face recognition library installation doesn’t give us an error.

**Face-recognition:**

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

**NumPy:**

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

**OpenCV-python:**

OpenCV is used for all sorts of image and video analysis, like facial recognition and detection.

**Tkinter:**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.

**Pandas:**

Panda is a software library written for the Python programming language for data manipulation and analysis.

Other tools like VS Code, GitHub are also use in this project.

**Chapter Two**

1. **Flowchart**

**Chapter Three**

1. **Implementation**

This project uses two python files:

setup.py(python setup build) and train.py (main file)

**Code:**

*setup.py*

from cx\_Freeze import setup, Executable

import sys,os

PYTHON\_INSTALL\_DIR = os.path.dirname(os.path.dirname(os.\_\_file\_\_))

os.environ['TCL\_LIBRARY'] = os.path.join(PYTHON\_INSTALL\_DIR, 'tcl', 'tcl8.6')

os.environ['TK\_LIBRARY'] = os.path.join(PYTHON\_INSTALL\_DIR, 'tcl', 'tk8.6')

base = None

if sys.platform == 'win32':

base = None

executables = [Executable("train.py", base=base)]

packages = ["idna","os","sys","cx\_Freeze","tkinter","cv2","setup",

"numpy","PIL","pandas","datetime","time"]

options = {

'build\_exe': {

'packages':packages,

},

}

setup(

name = "ToolBox",

options = options,

version = "0.0.1",

description = 'Vision ToolBox',

executables = executables

)

*train.py*

import tkinter as tk

from tkinter import Message ,Text

import cv2,os

import shutil

import csv

import numpy as np

from PIL import Image, ImageTk

import pandas as pd

import datetime

import time

import tkinter.ttk as ttk

import tkinter.font as font

window = tk.Tk()

window.title("Face Recogniton Project")

dialog\_title = 'QUIT'

dialog\_text = 'Are you sure?'

window.geometry('1280x720')

window.configure(background='blue')

#window.attributes('-fullscreen', True)

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

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

message = tk.Label(window, text="COLLEGE ATTENDANCE SYSTEM" ,bg="Green" ,fg="white" ,width=50 ,height=3,font=('times', 30, 'italic bold underline'))

message.place(x=100, y=20)

lbl = tk.Label(window, text="Enter ID",width=20 ,height=2 ,fg="red" ,bg="yellow" ,font=('times', 15, ' bold ') )

lbl.place(x=100, y=200)

txt = tk.Entry(window,width=20 ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))

txt.place(x=400, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20 ,fg="red" ,bg="yellow" ,height=2 ,font=('times', 15, ' bold '))

lbl2.place(x=100, y=300)

txt2 = tk.Entry(window,width=20 ,bg="yellow" ,fg="red",font=('times', 15, ' bold ') )

txt2.place(x=400, y=315)

lbl3 = tk.Label(window, text="Notification : ",width=20 ,fg="red" ,bg="yellow" ,height=2 ,font=('times', 15, ' bold underline '))

lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="yellow" ,fg="red" ,width=30 ,height=2, activebackground = "yellow" ,font=('times', 15, ' bold '))

message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance : ",width=20 ,fg="red" ,bg="yellow" ,height=2 ,font=('times', 15, ' bold underline'))

lbl3.place(x=400, y=650)

message2 = tk.Label(window, text="" ,fg="red" ,bg="yellow",activeforeground = "green",width=30 ,height=2 ,font=('times', 15, ' bold '))

message2.place(x=700, y=650)

def clear():

txt.delete(0, 'end')

res = ""

message.configure(text= res)

def clear2():

txt2.delete(0, 'end')

res = ""

message.configure(text= res)

def is\_number(s):

try:

float(s)

return True

except ValueError:

pass

try:

import unicodedata

unicodedata.numeric(s)

return True

except (TypeError, ValueError):

pass

return False

def TakeImages():

Id=(txt.get())

name=(txt2.get())

if(is\_number(Id) and name.isalpha()):

cam = cv2.VideoCapture(0)

harcascadePath = "haarcascade\_frontalface\_default.xml"

detector=cv2.CascadeClassifier(harcascadePath)

sampleNum=0

while(True):

ret, img = cam.read()

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

faces = detector.detectMultiScale(gray, 1.3, 5)

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

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

sampleNum=sampleNum+1

cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])

#display the frame

cv2.imshow('frame',img)

if cv2.waitKey(100) & 0xFF == ord('q'):

break

elif sampleNum>60:

break

cam.release()

cv2.destroyAllWindows()

res = "Images Saved for ID : " + Id +" Name : "+ name

row = [Id , name]

with open('StudentDetails\StudentDetails.csv','a+') as csvFile:

writer = csv.writer(csvFile)

writer.writerow(row)

csvFile.close()

message.configure(text= res)

else:

if(is\_number(Id)):

res = "Enter first name only"

message.configure(text= res)

if(name.isalpha()):

res = "Enter Numeric Id"

message.configure(text= res)

def TrainImages():

recognizer = cv2.face\_LBPHFaceRecognizer.create()

harcascadePath = "haarcascade\_frontalface\_default.xml"

detector =cv2.CascadeClassifier(harcascadePath)

faces,Id = getImagesAndLabels("TrainingImage")

recognizer.train(faces, np.array(Id))

recognizer.save("TrainingImageLabel\Trainner.yml")

res = "Image Trained"

message.configure(text= res)

def getImagesAndLabels(path):

imagePaths=[os.path.join(path,f) for f in os.listdir(path)]

faces=[]

Ids=[]

for imagePath in imagePaths:

pilImage=Image.open(imagePath).convert('L')

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

Id=int(os.path.split(imagePath)[-1].split(".")[1])

faces.append(imageNp)

Ids.append(Id)

return faces,Ids

def TrackImages():

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read("TrainingImageLabel\Trainner.yml")

harcascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(harcascadePath);

df=pd.read\_csv("StudentDetails\StudentDetails.csv")

cam = cv2.VideoCapture(0)

font = cv2.FONT\_HERSHEY\_SIMPLEX

col\_names = ['Id','Name','Date','Time']

attendance = pd.DataFrame(columns = col\_names)

while True:

ret, im =cam.read()

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

faces=faceCascade.detectMultiScale(gray, 1.2,5)

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

cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)

Id, conf = recognizer.predict(gray[y:y+h,x:x+w])

if(conf < 50):

ts = time.time()

date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

aa=df.loc[df['Id'] == Id]['Name'].values

tt=str(Id)+"-"+aa

attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]

else:

Id='Unknown'

tt=str(Id)

if(conf > 75):

noOfFile=len(os.listdir("ImagesUnknown"))+1

cv2.imwrite("ImagesUnknown\Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])

cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)

attendance=attendance.drop\_duplicates(subset=['Id'],keep='first')

cv2.imshow('im',im)

if (cv2.waitKey(1)==ord('q')):

break

ts = time.time()

date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

Hour,Minute,Second=timeStamp.split(":")

fileName="Attendance\Attendance\_"+date+"\_"+Hour+"-"+Minute+"-"+Second+".csv"

attendance.to\_csv(fileName,index=False)

cam.release()

cv2.destroyAllWindows()

res=attendance

message2.configure(text= res)

clearButton = tk.Button(window, text="Clear", command=clear ,fg="red" ,bg="yellow" ,width=20 ,height=2 ,activebackground = "Red" ,font=('times', 15, ' bold '))

clearButton.place(x=950, y=200)

clearButton2 = tk.Button(window, text="Clear", command=clear2 ,fg="red" ,bg="yellow" ,width=20 ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))

clearButton2.place(x=950, y=300)

takeImg = tk.Button(window, text="Take Images", command=TakeImages ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

takeImg.place(x=100, y=500)

trainImg = tk.Button(window, text="Train Images", command=TrainImages ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

trainImg.place(x=400, y=500)

trackImg = tk.Button(window, text="Track Images", command=TrackImages ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

trackImg.place(x=700, y=500)

quitWindow = tk.Button(window, text="Quit", command=window.destroy ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

quitWindow.place(x=1000, y=500)

copyWrite = tk.Text(window, background=window.cget("background"), borderwidth=0,font=('times', 30, 'italic bold underline'))

copyWrite.tag\_configure("superscript", offset=10)

copyWrite.insert("insert", "Developed by Team","", "TEAM", "superscript")

#Sabin Maharjan

#Anil Raj Rimal

#Pawan Sapkota

#Bibhav Pokhrel

copyWrite.configure(state="disabled",fg="red" )

copyWrite.pack(side="left")

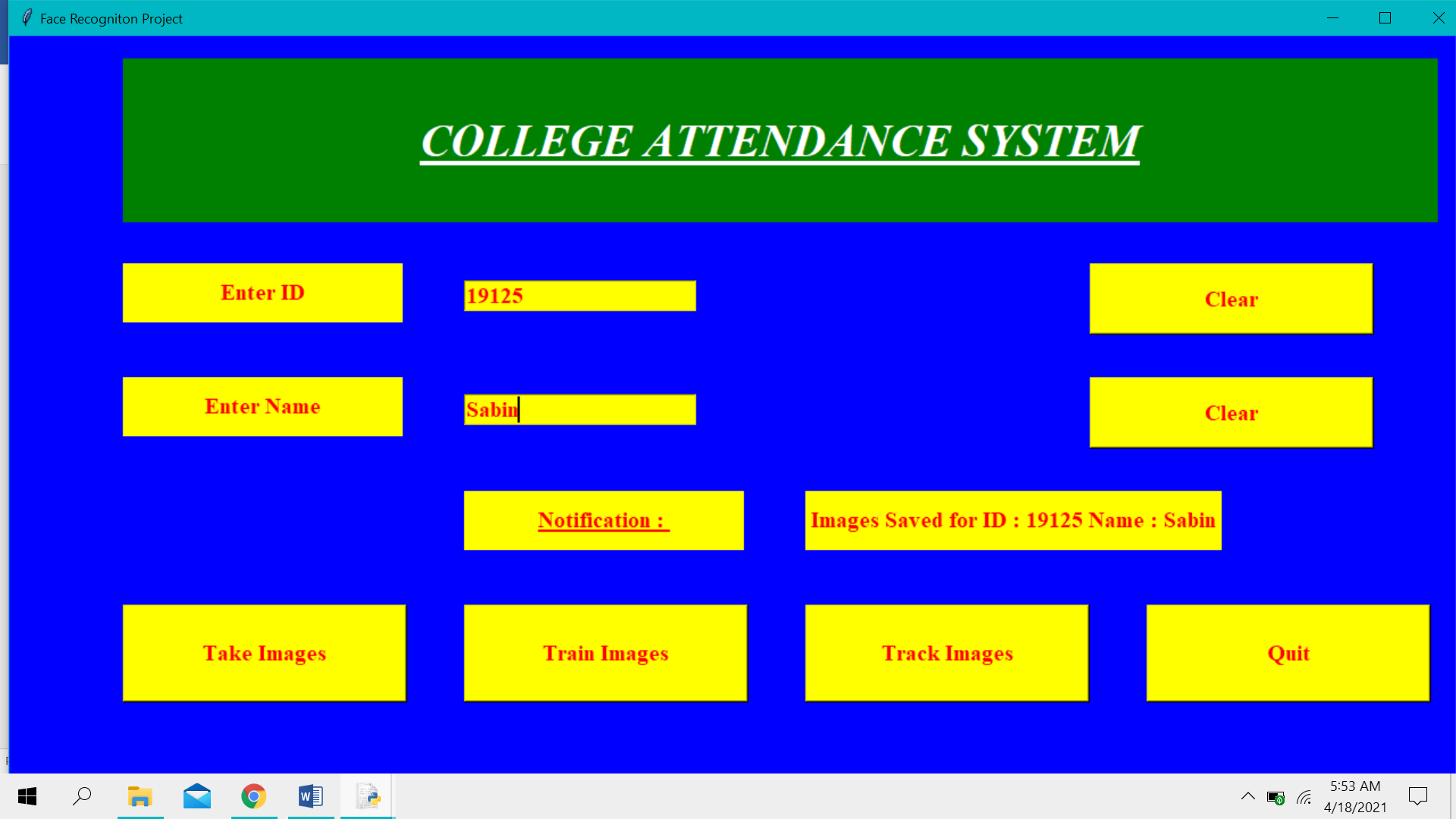
copyWrite.place(x=800, y=750)

window.mainloop()

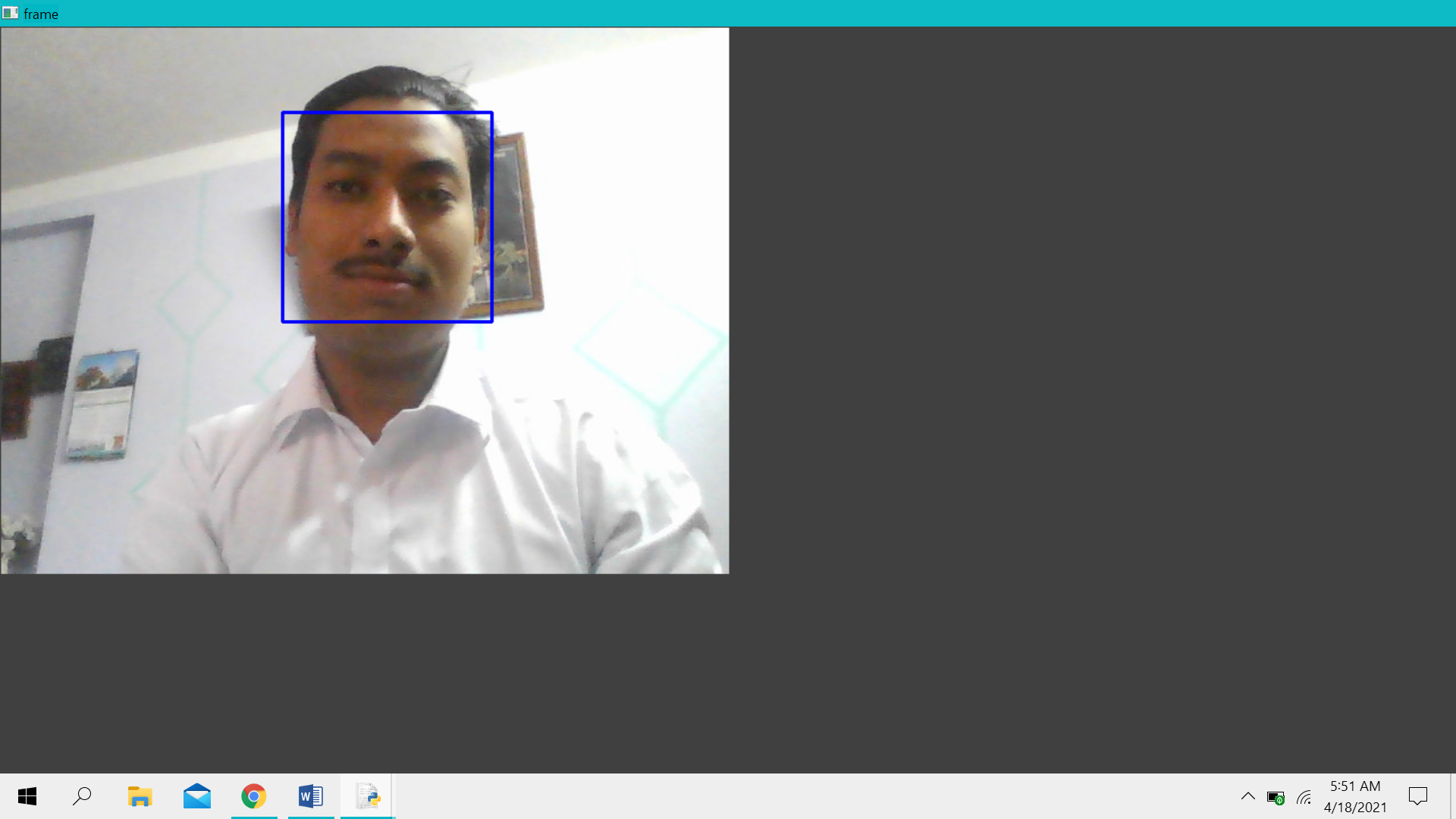
**Output:**

The output is categorized into different parts:

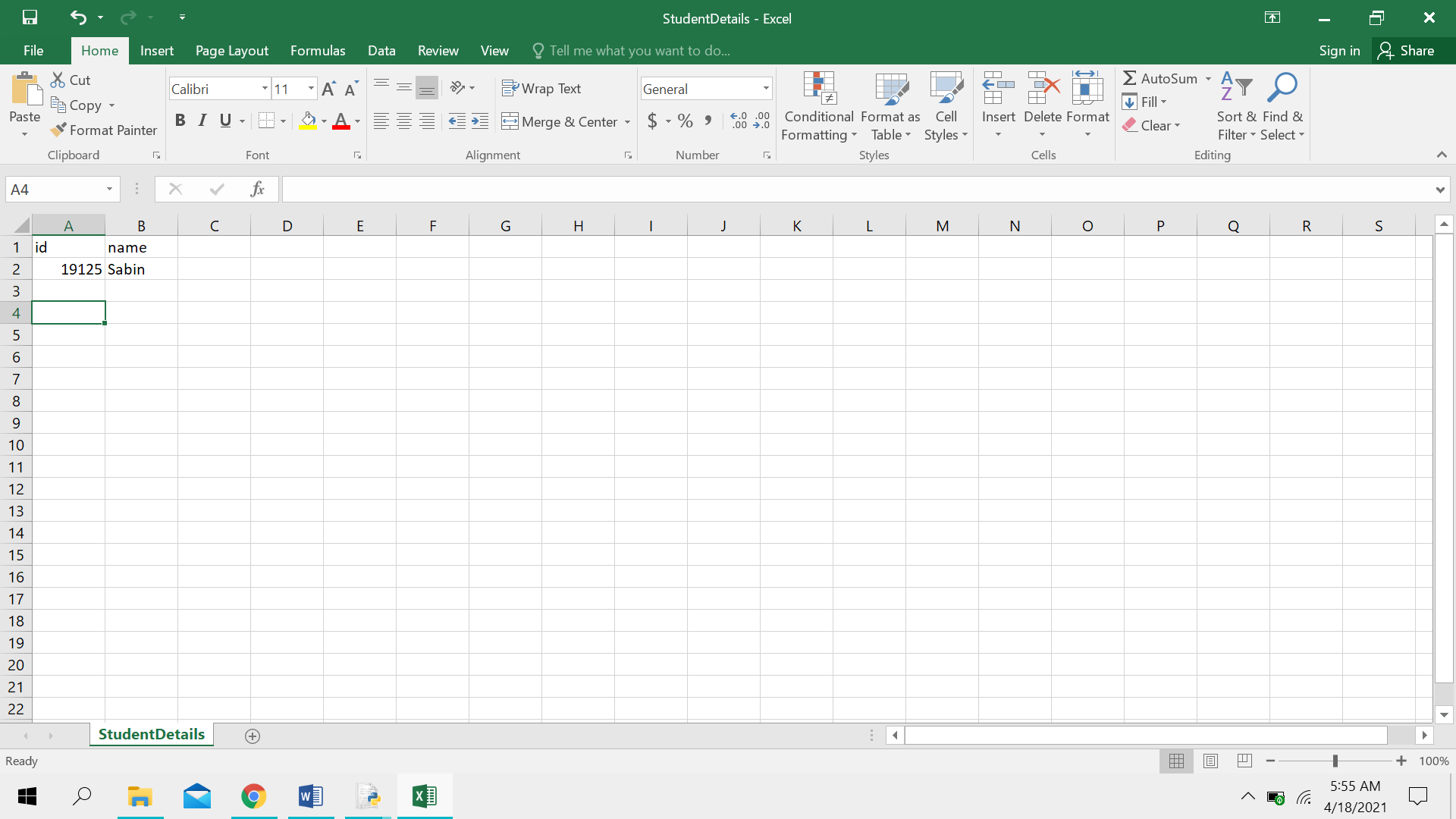
1. Main window for the users:



1. Face being recognized by the program:



1. Attendance being recorded into an excel(csv) file with id and name:



**Conclusion**

Face recognition library being a high-level deep learning library helps in identifying faces accurately. We’ve then used this to build a face attendance system which can be helpful in offices, schools or any other place reducing manual labor and automatically updating the attendance records in day-to-day life. This also notes down the time of arrival thus can acquire information about people coming in late after a specified time.