Week 12: Homework 1: Project: Facial Recognition on Raspberry Pi with AWS Rekognition

Md Shahadat Hossain Khan (19609)

In this exercise we are going to detect face from live feed of Raspberry Pi. But as we are using Raspbian OS in VirtualBox, we don't using Pi Camera. So, we need to use our webcam from host OS. So, we need such solution that will work in both like VirtualBox environment and in real hardware i.e. Pi Camera. For VirtualBox environment we have other challenge like we need to attach host OS with guest OS.

To achieve our goal we are going to use following technology -

1. Python: Make the entire project happen
2. OpenCV: Handle camera stream
3. boto3: Communicate with AWS

Here we are using OpenCV due to portability of project. Like when we use real hardware we don't need to change our code much. Also OpenCV is a very powerful project, we can add many feature in our project with the help of OpenCV.

Here is the steps we are going to follow to achieve our goal -

1. Prepare OS to attach host camera
2. Install required module into Raspbian OS
3. Configure AWS
4. Code our project
5. Execute and test

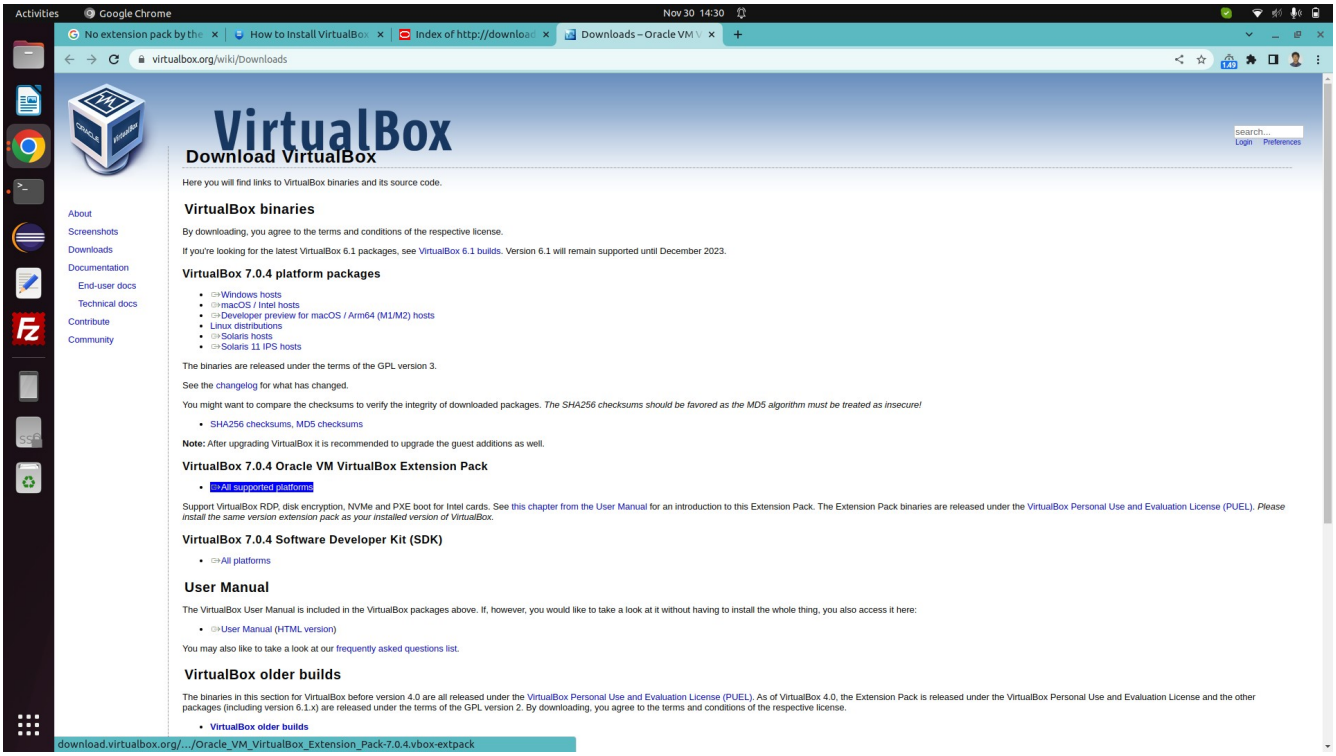
Please visit https://github.com/mkhan-sfbu/AWS-IoT-Raspberry-Pi-Emulator-Image-and-Video-Analysis for complete code

Prepare OS to attach host camera

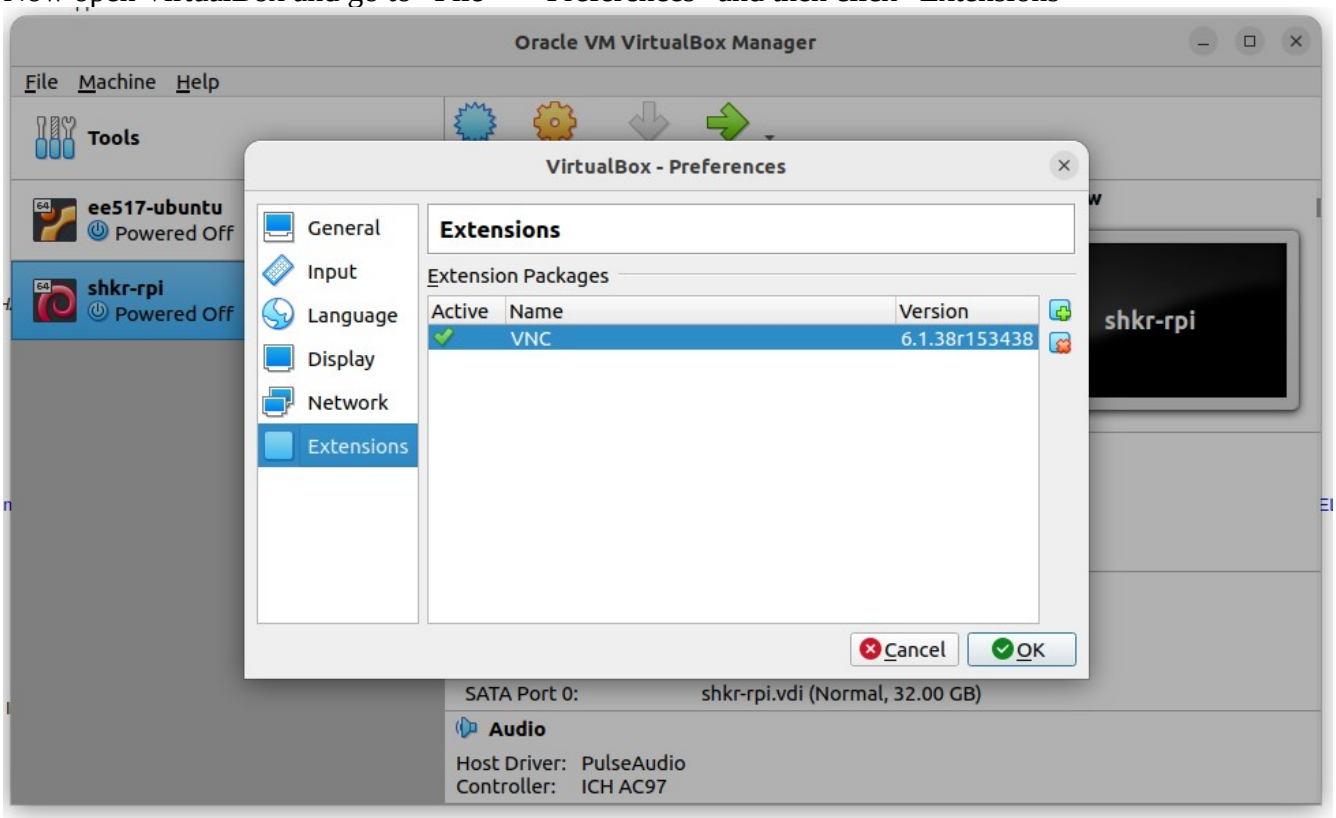
As we are using VirtualBox and we are going to use our host OS camera at guest, we need to tell VirtualBox that we are going to borrow that hardware. Issue is, we need to execute a command each time we start Raspbian OS to attach camera. I can't find any automatic solution that can attach camera when we start Raspbian OS.

Also we need to load / install "Oracle VM Virtual Box Extension Pack" if we already don't have that extension pack. To install it we need to download it from Oracle Virtual Box site. Please note that we must matched our Virtual Box version with extension pack version.

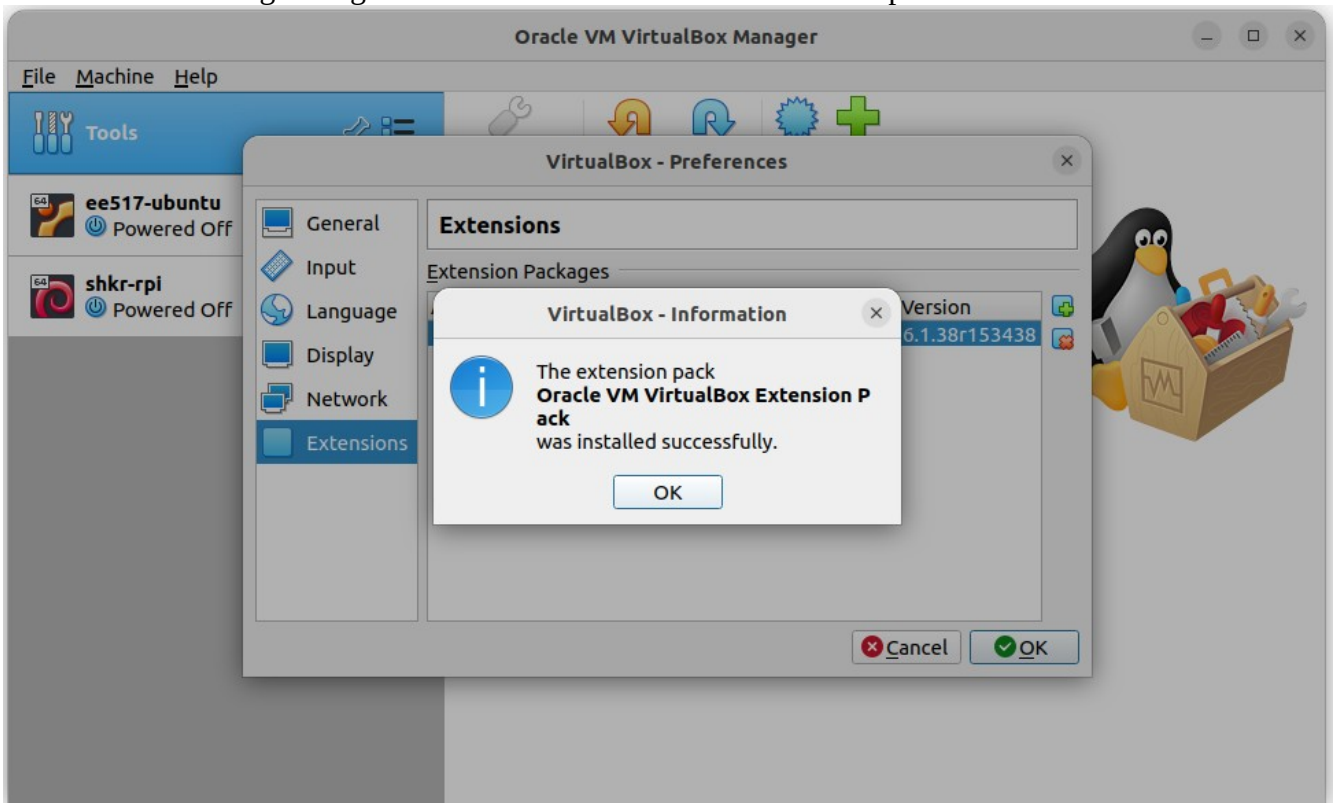
Here we downloading extension pack from virtualbox.org



Now open VirtualBox and go to “File” > “Preferences” and then click “Extensions”



Now click on “+” sign at right side and select downloaded extension pack to install.



Now we have extension pack installed into our VirtualBox, we need to know our webcam location to attach.

```
$ VBoxManage list webcams
```

```
shahadat@shkr-HP-Pavilion-Laptop-15t-eg100:/mnt/shkr-data/shkr/shkr-study/mscs-sfbu/fall22/ee517__introduction-to-the-internet-of-things-iot/week12/hw1$ VBoxManage list webcams
Video Input Devices: 1
.1 "HP Wide Vision HD Camera: HP Wi"
/dev/video0
```

Now we know our camera location which is “/dev/video0”. Now we need to attach this camera with our guest OS i.e. Raspbian OS. Please note, we need to attach this device every time after running our virtual machine. So, run Raspbian and execute following command at host OS when guest OS completely stable state.

```
$ VBoxManage controlvm shkr-rpi webcam attach /dev/video0
```

Here -
- “controlvm” is the sub-command
- “shkr-rpi” is my virtual machine name. [to find VM name apply “VBoxManage list vms”]
- “webcam” hardware type we are going to attach
- “attach” command to attach [when we need to detach camera, we can use “detach” command here]
- “/dev/video0” is our webcam location. We can also use number like “.1” instead of this location.

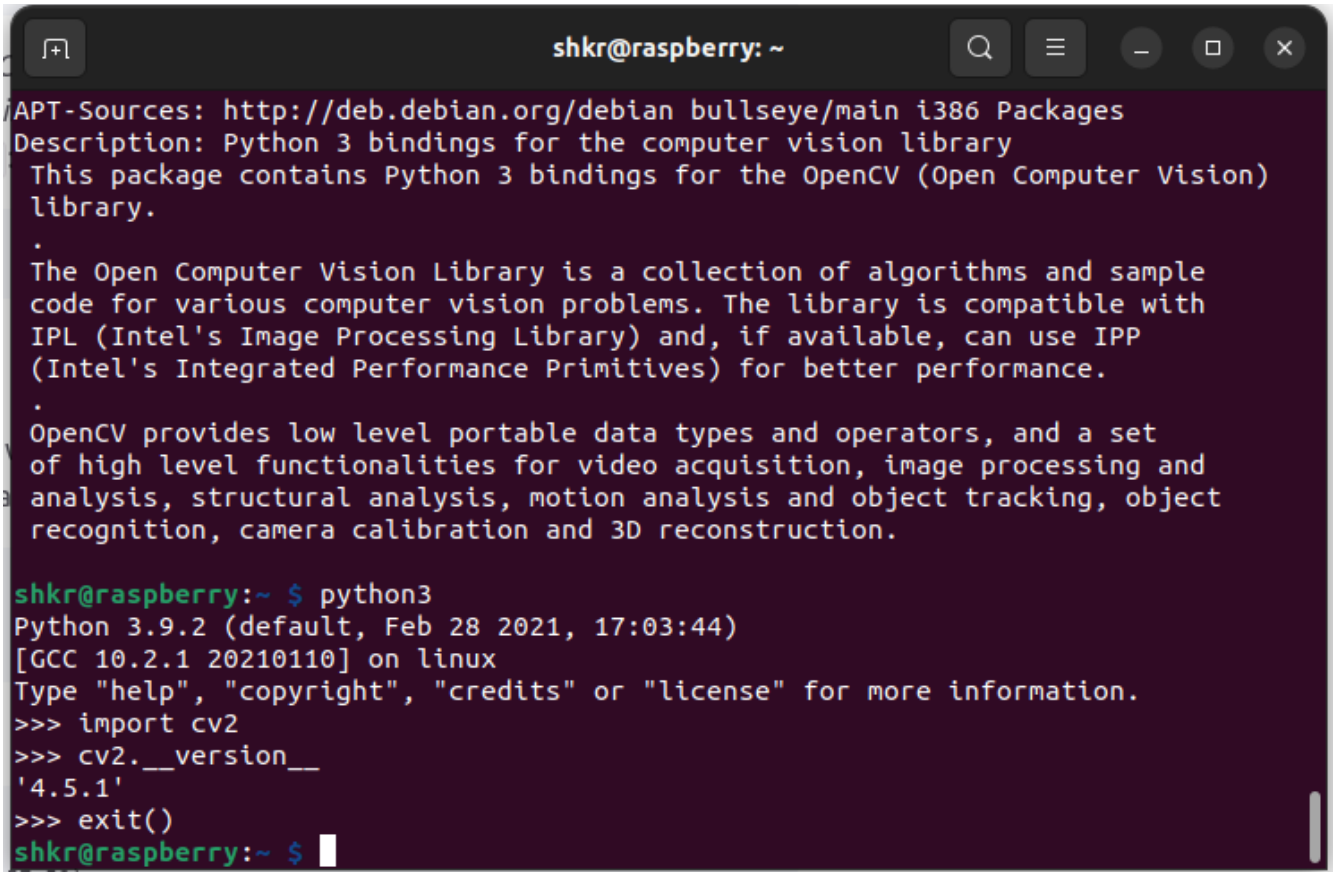
Install required module into Raspbian OS

Now we need to prepare our Raspberry Pi to make it happen. We need to install following modules -
1. OpenCV
2. boto3

But we already install boto3 for our previous project, so we are going to install only OpenCV. To install OpenCv we need to follow these steps -

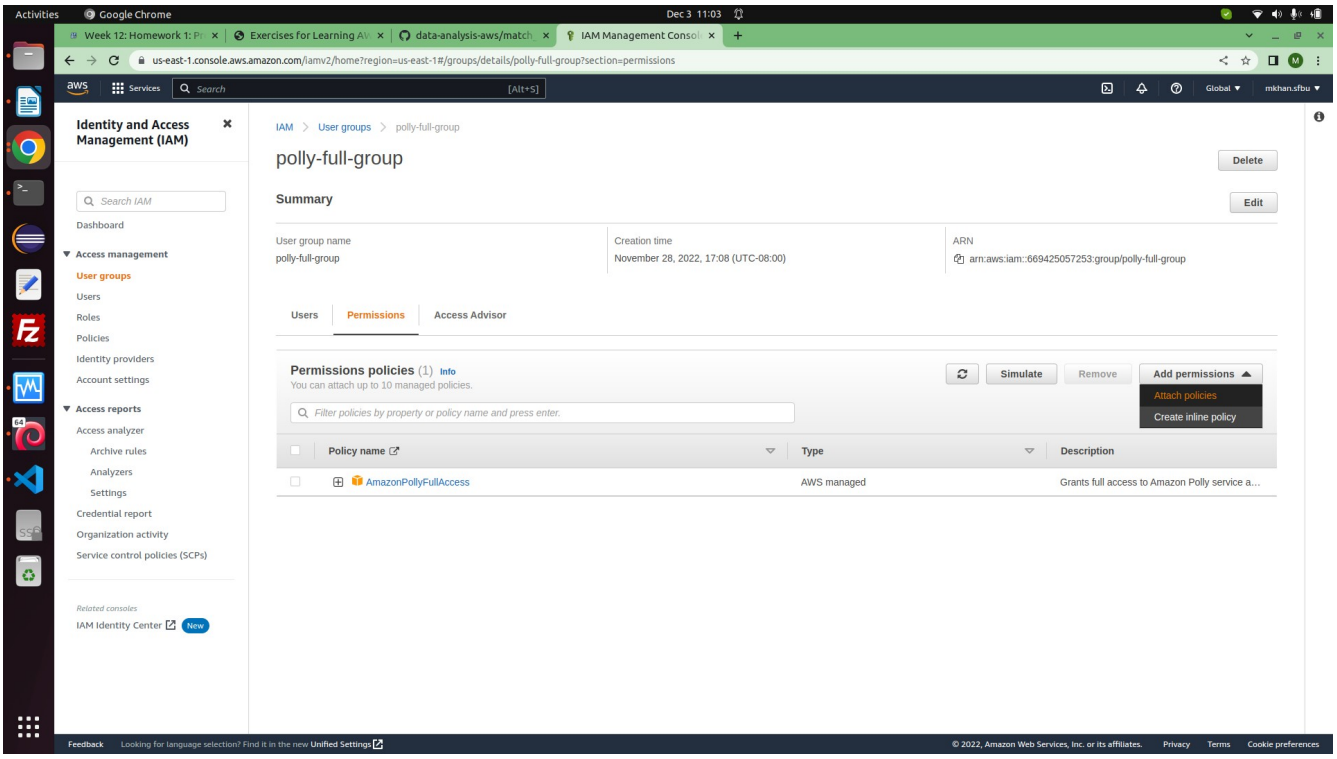
```
$ sudo apt-get install software-properties-common
$ sudo apt-get install build-essential cmake pkg-config libjpeg-dev libtiff5-
dev libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
libxvidcore-dev libx264-dev libfontconfig1-dev libcairo2-dev libgdk-pixbuf2.0-
dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libatlas-base-dev gfortran
libhdf5-dev libhdf5-serial-dev libhdf5-103 python3-pyqt5 python3-dev -y
$ pip install numpy
$ pip install cmake
$ pip install --upgrade pip setuptools wheel
$ sudo apt install python3-opencv
```

After applying above commands we can check our OpenCv installation by following commands in screenshot.

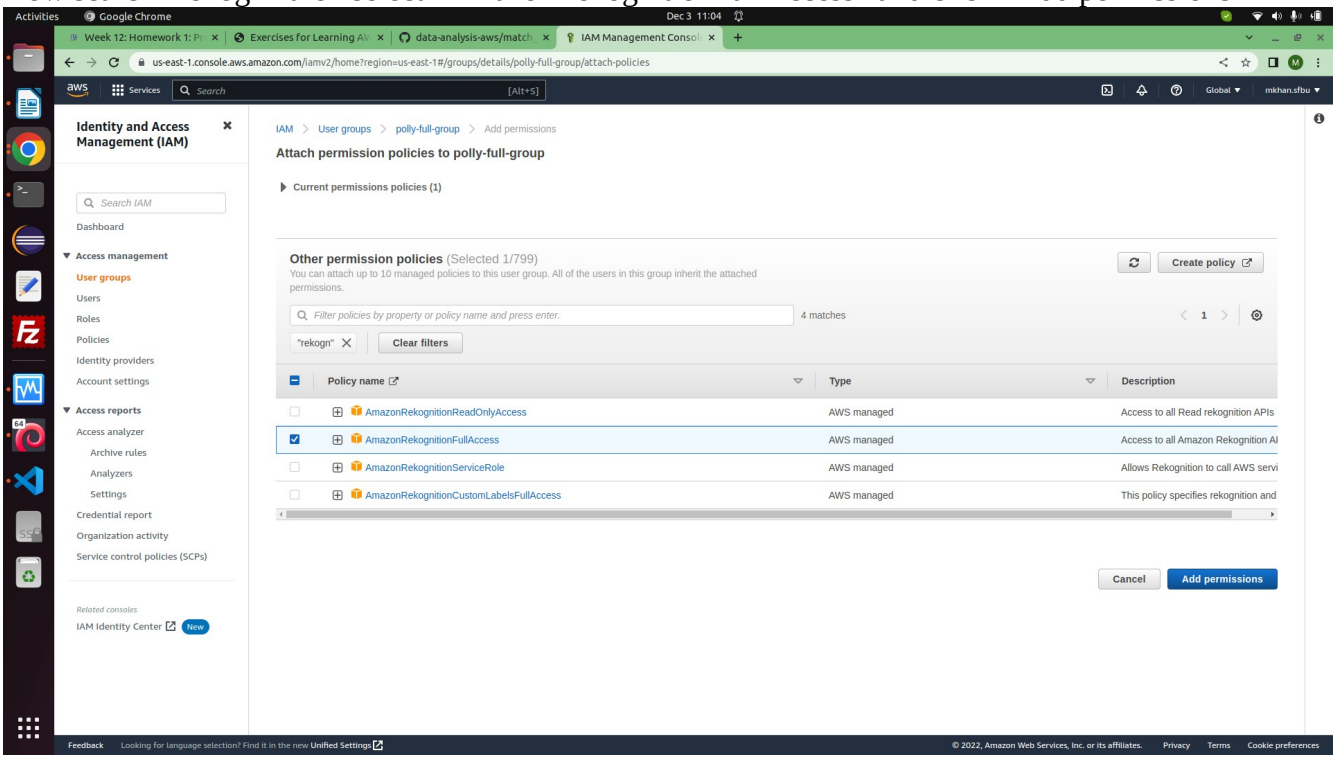


Configure AWS

Now we need to assign our IAM user to access for “rekognition” API. To do this open our “polly-full-group” from AWS IAM control panel’s “User groups” section. There we need to go “Permissions” tab and then we need to click “Add permissions” drop down list to find “Attach policies” link.



Now search “rekogn” then select “AmazonRekognitionFullAccess” and click “Add permissions”



Code our project

Now we are going to read our camera stream frame by frame with the help of OpenCV. After getting a frame we are going to analysis it by built-in feature of OpenCV for any face into that frame/image. If we found any face we then ask AWS to detect that face. If AWS tells us that the face doesn't match with any face into the active collection, we will do following task as per parameter.

1. In learn Mode: We will ask user to provide is name of that face. And store that face into AWS collection. Note that, we run a web-server to show the detected face to user.

2. Otherwise: We will tell that the face doesn’t match.

Here is the scripts (rekognition.py) where all above task happened -

```
import signal
import cv2
import boto3
from contextlib import closing
import tempfile
import pygame
from botocore.exceptions import BotoCoreError, ClientError
import sys
import time
import os
import socket
import fcntl
import struct
from http.server import BaseHTTPRequestHandler, HTTPServer
import threading
from urllib.parse import parse_qs
import base64


# This will return video from the first webcam on your computer.
cap = cv2.VideoCapture(0)
#setting the buffer size and frames per second, to reduce frames in buffer
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
cap.set(cv2.CAP_PROP_FPS, 2)

session = boto3.Session(aws_access_key_id="AKIAZXXGXXXSZBQHVOWZ",
aws_secret_access_key="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",
region_name='us-east-1')

polly = session.client('polly')
rekognition = session.client('rekognition')

action = "detect"
if len(sys.argv)>2 : action = sys.argv[2]

interface='eth1'

def getIpAddress(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
```

```
        0x8915,  # SIOCGIFADDR
        struct.pack(b'256s', bytes(ifname[:15], 'utf-8'))
    )[20:24])

ipAddress=getIpAddress(interface)
print('Server IP Address: '+ipAddress)
serverPort=7328

dirname = os.path.dirname(__file__)
tmpDirectory = os.path.join(dirname, 'tmp-images')


class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        path, _, query_string = self.path.partition('?')
        if path != '/favicon.ico':
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(bytes("<html><head><title>Simple Web Server to
Show Image, Developed By SHKR</title></head><body>", "utf-8"))
            self.wfile.write(bytes("<p>Request: %s</p>" % self.path, "utf-8"))
            self.wfile.write(bytes("<p>Parsed Path: %s</p>" % path, "utf-8"))
            if path == '/':
                self.wfile.write(bytes("<p>This is a simple web server to show
image.</p>", "utf-8"))
                self.wfile.write(bytes("<p>Temporary Directory: %s</p>" %
tmpDirectory, "utf-8"))
            else:
                # need to show image!!!
                with open('{0}{1}.png'.format(tmpDirectory, path), 'rb') as
image_file:
                    encoded_string = base64.b64encode(image_file.read())
                    self.wfile.write(bytes("<img
src=\"%s\" />" % encoded_string.decode('utf-8'), "utf-
8"))
            self.wfile.write(bytes("</body></html>", "utf-8"))
        else:
            self.send_response(404)
            self.end_headers()


webServer = None

if action=='learn':
    webServer = HTTPServer((ipAddress, serverPort), MyServer)
    print("Server configured at http://%s:%s" % (ipAddress, serverPort))
    daemon = threading.Thread(name='daemon_server',
target=webServer.serve_forever)
    daemon.setDaemon(True) # Set as a daemon so it will be killed once the
main thread is dead.
    daemon.start()

def destroyAndRelease():
    # Close the window / Release webcam
    cap.release()
```

```
    # De-allocate any associated memory usage
    cv2.destroyAllWindows()

    # close rekognition
    rekognition.close()

    # close polly
    polly.close()

    if webServer is not None: webServer.server_close()

    ##print("", end="\r", flush=True)
    ##print(" " * len(msg), end="", flush=True) # clear the printed line
    print("    ", end="\r", flush=True)


def handleExit(signum, frame):
    destroyAndRelease()
    sys.exit(1)


signal.signal(signal.SIGINT, handleExit)



def speakUpPolly(text2speak):
    try:
        # Request speech synthesis
        response = polly.synthesize_speech(Text=text2speak,
OutputFormat="mp3", VoiceId="Joanna")
    except (BotoCoreError, ClientError) as error:
        # The service returned an error, exit gracefully
        print(error)
        sys.exit(-1)

    if "AudioStream" in response:
        with closing(response["AudioStream"]) as stream:
            output = os.path.join(tempfile.gettempdir(), "speech.mp3")
            # print("file stored into "+output)

            try:
                # Open a file for writing the output as a binary stream
                with open(output, "wb") as file:
                    file.write(stream.read())

                print("Polly speaking: "+text2speak)
                pygame.mixer.init()
                pygame.mixer.music.load(output)
                pygame.mixer.music.play()
                while pygame.mixer.music.get_busy() == True:
                    continue

            except IOError as error:
                # Could not write to file, exit gracefully
                print(error)
                sys.exit(-1)
```

```
    else:
        # The response didn't contain audio data, exit gracefully
        print("Could not stream audio")
        sys.exit(-1)


if not cap.isOpened:
    speakUpPolly('Camera NOT Found!')
    print('--(!)Error opening video capture')
    destroyAndRelease()
    sys.exit(-1)



collectionId = "shkrtest"
if len(sys.argv)>1 : collectionId = sys.argv[1]
collections=rekognition.list_collections(MaxResults=10)
if not collectionId in collections['CollectionIds']:
    speakUpPolly('Collection '+collectionId+' NOT Found!')
    if action!='learn':
        destroyAndRelease()
        sys.exit(-1)
    # as we learning we need to create collection
    rekognition.create_collection(CollectionId=collectionId)
    speakUpPolly('Successfully Created Collection '+collectionId)


faceDirectory = os.path.join(dirname, 'faces')

faceCascade = cv2.CascadeClassifier(os.path.join(dirname,
'haarcascade_frontalface_default.xml'))


# loop runs if capturing has been initialized.
while(True):
    milli = int(round(time.time() * 1000))
    # reads frames from a camera
    frame = {}
    #calling read() twice as a workaround to clear the buffer.
    cap.read()
    cap.read()
    # ret checks return at each frame
    ret, frame = cap.read()

    if frame is None:
        speakUpPolly('No captured frame')
        print('--(!) No captured frame -- Break!')
        break

    timestr = time.strftime("%Y%m%d-%H%M%S")
    imgLocation = '{0}/image_{1}.png'.format(tmpDirectory, timestr)
    cv2.imwrite(imgLocation, frame)
    # Read the input image
    img = cv2.imread(imgLocation)
    # Convert into grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Detect faces
```

```
    faces = faceCascade.detectMultiScale(gray, 1.1, 4)
    if len(faces)>0:
        # imdata=cv2.imencode('.jpg', img)[1].tobytes()
        try: #match the captured imges against the indexed faces
            match_response =
rekognition.search_faces_by_image(CollectionId=collectionId, Image={'Bytes':
cv2.imencode('.jpg', img)[1].tobytes()}, FaceMatchThreshold=85)
            matches={}
            for p in match_response['FaceMatches']:
                matches[p['Face']['ExternalImageId']]=p['Face']
['ExternalImageId']
            print('FaceMatches: '+str(len(matches))+' | cv2:
'+str(len(faces)))
            if action=='learn' and len(matches)!=len(faces):
                # new face found! need to update our collection...
                for (x, y, w, h) in faces:
                    ROI = img[y:y+h, x:x+w]
                    nImgLoc = 'ni-{0}_x{1}y{2}w{3}h{4}'.format(timestr, x, y,
w, h)
                    cv2.imwrite('{0}/{1}.png'.format(tmpDirectory, nImgLoc),
ROI)
                    print("Visit http://%s:%s/%s" % (ipAddress, serverPort,
nImgLoc))
                    speakUpPolly('Please input name for new person')
                    label=input('Name: ')
                    if len(label)>0:

idxResponse=rekognition.index_faces(CollectionId=collectionId,
                                    Image={'Bytes': cv2.imencode('.jpg', ROI)
[1].tobytes()},
                                    ExternalImageId=label,
                                    MaxFaces=1,
                                    QualityFilter="AUTO",
                                    DetectionAttributes=['ALL'])
                        print('FaceId: ', idxResponse['FaceRecords'][0]
['Face']['FaceId'])
                    else:
                        print('%s ignored' % nImgLoc)


            if len(matches)>0:
                for prsn in matches:
                    faceImgLocation = '{0}/{1}_{2}.png'.format(faceDirectory,
prsn, timestr)
                    cv2.imwrite(faceImgLocation, frame)
                    print('Hello, ', prsn)
                    speakUpPolly('Hello, '+prsn+'! How are you?')
            else:
                speakUpPolly('No faces matched!')
                print('No faces matched')
        except:
            print('No face detected '+str(milli)+' | file: '+imgLocation)
    else:
        print('No Face Found. Waiting for someone...')


destroyAndRelease()
```

We do face detection using Haar cascades which is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc.. Today we will be using the face classifier.

We need to download the trained classifier XML file (haarcascade_frontalface_default.xml), which is available in OpenCv's GitHub repository - https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

Execute and test

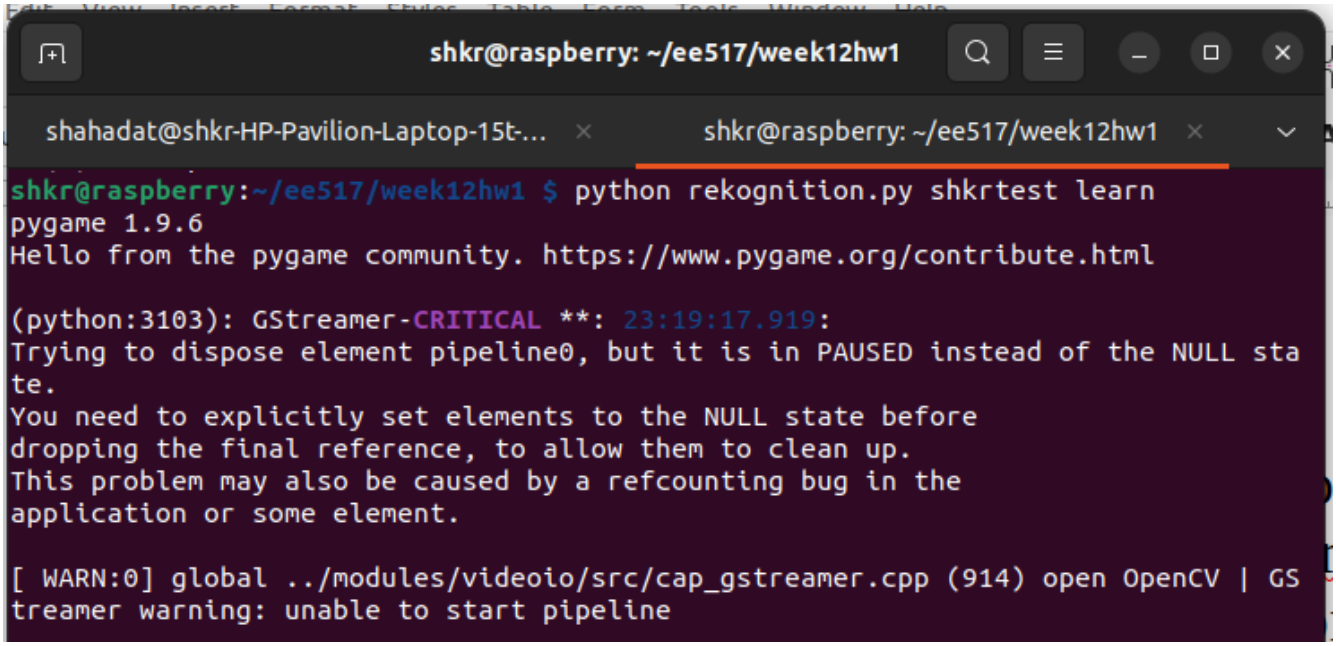
Before execute our program we need to create two folder which required for our code which are "tmp-images" and "faces". We need to create these directory into where our code will store and execute. We can create those directory by mkdir command as follows -

```
$ cd <directory/contain/above/code/file>
$ mkdir tmp-images
$ mkdir faces
```

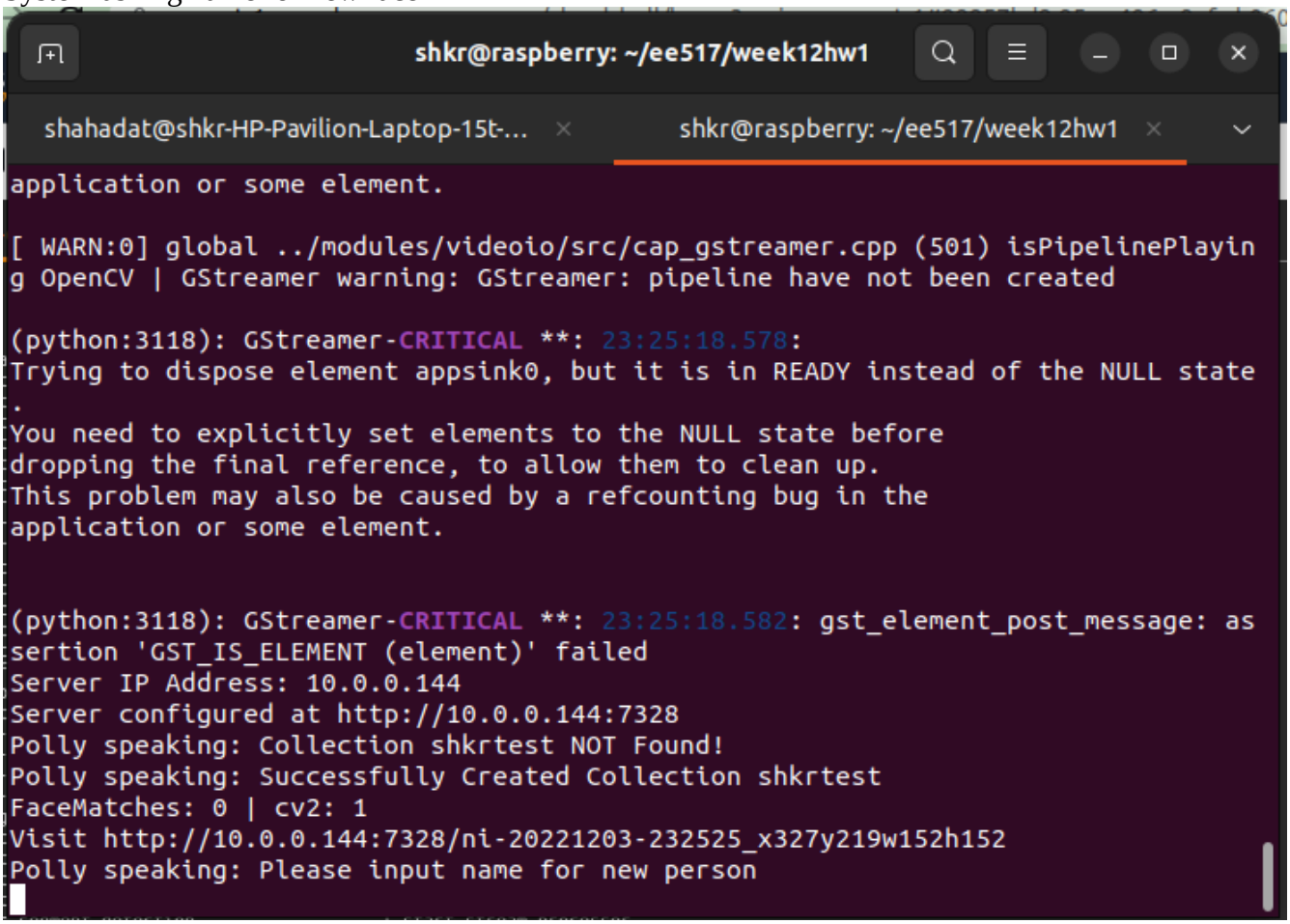
Now we are ready to execute our program. We will use following command -

```
$ python rekognition.py sfbu learn
```

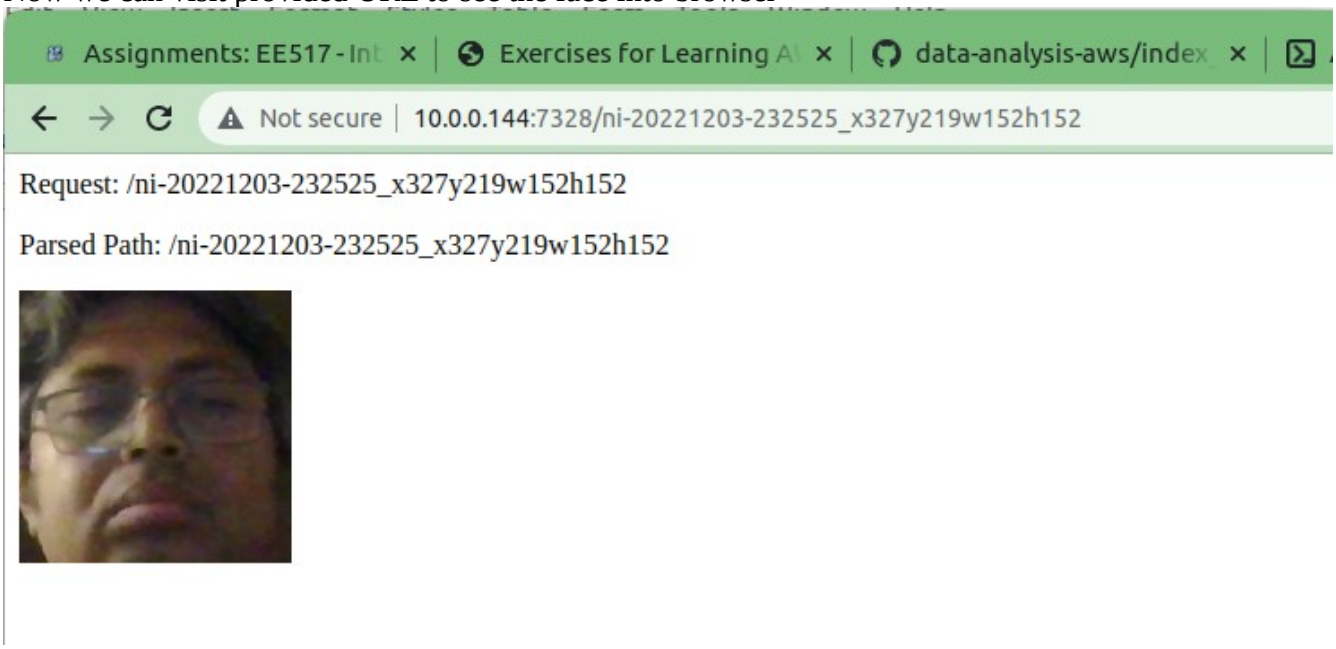
Here -
- "python" is the compiler and executer
- "rekognition.py" our script to execute
- "sfbu" collection ID where we are going to store our known face. This parameter is optional, if nothing provide system will assume as "shkrtest" collection
- "learn" command (optional parameter) if provide, our program will ask name for NOT matched faces and add into AWS collection



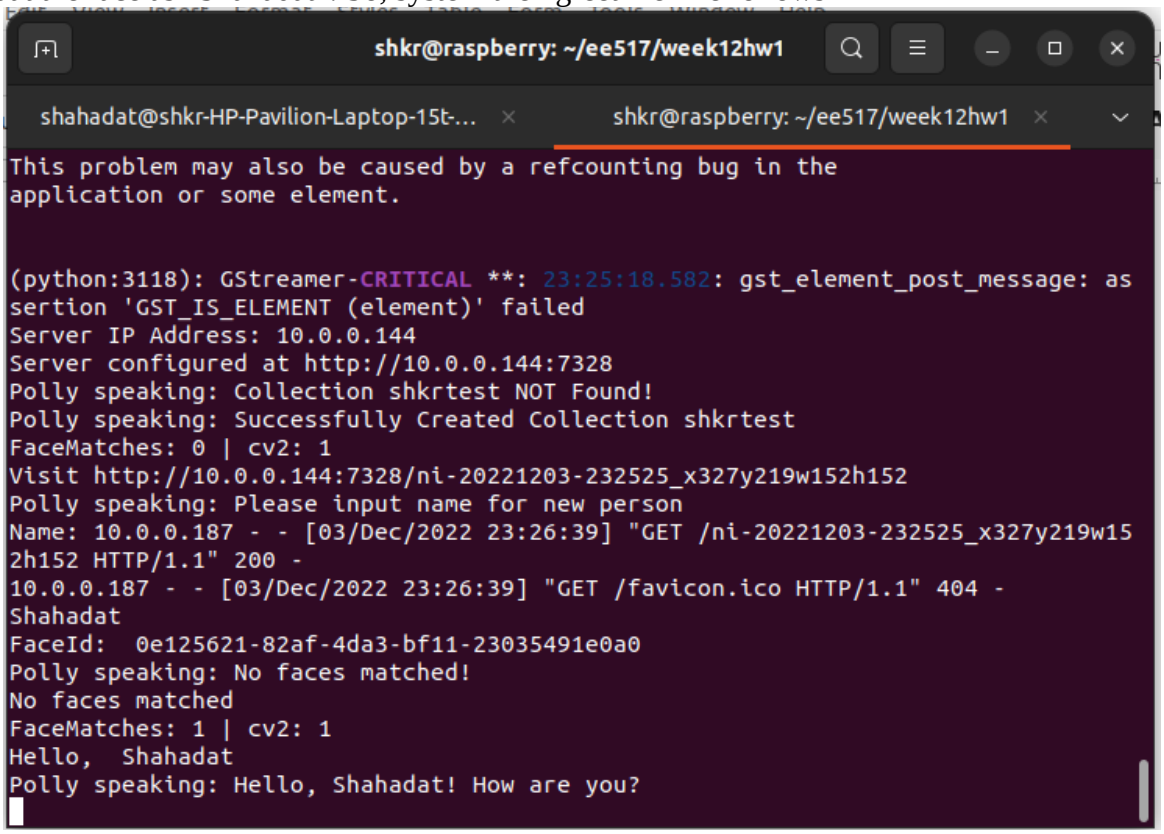
System asking name for new face -



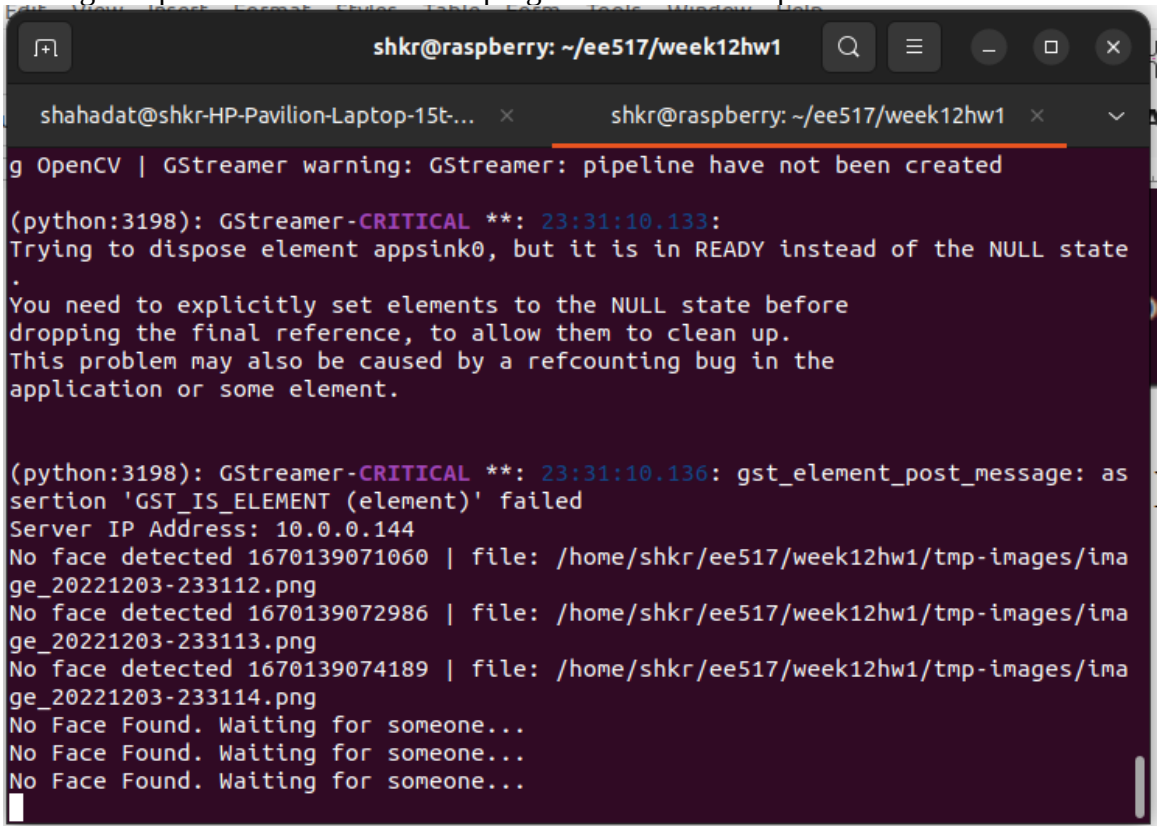
Now we can visit provided URL to see the face into browser -



We input the face as “Shahadat”. So, system then greet me like follows -



After learning complete we can execute our program without learn parameter -



Our program will NOT ask name for new faces, instead it will tell “No face detected” also note that we primarily detect face by OpenCV, if no face in our camera, system will tell “No Face Found. Waiting for someone…” Otherwise system will greet for known faces.

Finally we can terminate our program by pressing “Ctrl + C” together.

Thank You