**CREATING COLLECTIONS**

import boto3

def create\_collection(collection\_id):

client=boto3.client('rekognition')

#Create a collection

print('Creating collection:' + collection\_id)

response=client.create\_collection(CollectionId=collection\_id)

print('Collection ARN: ' + response['CollectionArn'])

print('Status code: ' + str(response['StatusCode']))

print('Done...')

def main():

collection\_id='Collection'

create\_collection(collection\_id)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**DELETING COLLECTIONS**

import csv

import boto3

with open("new\_user\_credentials.csv",'r') as input:

next(input)

reader = csv.reader(input)

for line in reader:

access\_key\_id = line[0]

secret\_access\_key = line[1]

session\_token = line[2]

client = boto3.client('rekognition'

#aws\_access\_key\_id = access\_key\_id,

#aws\_secret\_access\_key = secret\_access\_key,

#aws\_session\_token = session\_token,

#region\_name = 'us-east-1'

)

def delete\_collection(collection\_id):

print('Attempting to delete collection ' + collection\_id)

status\_code=0

try:

response=client.delete\_collection(CollectionId=collection\_id)

status\_code=response['StatusCode']

except ClientError as e:

if e.response['Error']['Code'] == 'ResourceNotFoundException':

print ('The collection ' + collection\_id + ' was not found ')

else:

print ('Error other than Not Found occurred: ' + e.response['Error']['Message'])

status\_code=e.response['ResponseMetadata']['HTTPStatusCode']

return(status\_code)

def main():

collection\_id='Collection'

status\_code=delete\_collection(collection\_id)

print('Status code: ' + str(status\_code))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**ADDING FACES INTO COLLECTION**

import boto3

def add\_faces\_to\_collection(bucket,photo,collection\_id):

client=boto3.client('rekognition')

response=client.index\_faces(CollectionId=collection\_id,

Image={'S3Object':{'Bucket':bucket,'Name':photo}},

ExternalImageId=photo,

MaxFaces=1,

QualityFilter="AUTO",

DetectionAttributes=['ALL'])

print ('Results for ' + photo)

print('Faces indexed:')

for faceRecord in response['FaceRecords']:

print(' Face ID: ' + faceRecord['Face']['FaceId'])

print(' Location: {}'.format(faceRecord['Face']['BoundingBox']))

print('Faces not indexed:')

for unindexedFace in response['UnindexedFaces']:

print(' Location: {}'.format(unindexedFace['FaceDetail']['BoundingBox']))

print(' Reasons:')

for reason in unindexedFace['Reasons']:

print(' ' + reason)

return len(response['FaceRecords'])

def main():

bucket='album-manager'

collection\_id='Collection'

photo='test5.jpg'

indexed\_faces\_count=add\_faces\_to\_collection(bucket, photo, collection\_id)

print("Faces indexed count: " + str(indexed\_faces\_count))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**LISTING FACES INTO COLLECTION**

import boto3

def list\_faces\_in\_collection(collection\_id):

maxResults=2

faces\_count=0

tokens=True

client=boto3.client('rekognition')

response=client.list\_faces(CollectionId=collection\_id,

MaxResults=maxResults)

print('Faces in collection ' + collection\_id)

while tokens:

faces=response['Faces']

for face in faces:

print (face)

faces\_count+=1

if 'NextToken' in response:

nextToken=response['NextToken']

response=client.list\_faces(CollectionId=collection\_id,

NextToken=nextToken,MaxResults=maxResults)

else:

tokens=False

return faces\_count

def main():

collection\_id='Collection'

faces\_count=list\_faces\_in\_collection(collection\_id)

print("faces count: " + str(faces\_count))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**SEARCHING IMAGE BY COLLECTION**

import boto3

if \_\_name\_\_ == "\_\_main\_\_":

bucket='album-manager'

collectionId='Collection'

fileName='test5.jpg'

threshold = 70

maxFaces=2

client=boto3.client('rekognition')

response=client.search\_faces\_by\_image(CollectionId=collectionId,

Image={'S3Object':{'Bucket':bucket,'Name':fileName}},

FaceMatchThreshold=threshold,

MaxFaces=maxFaces)

faceMatches=response['FaceMatches']

print ('Matching faces')

for match in faceMatches:

print ('FaceId:' + match['Face']['FaceId'])

print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")

print

**PUTTING IMAGES INTO RESPECTIVE FOLDERS**

import boto3

import csv

with open("new\_user\_credentials.csv",'r') as input:

next(input)

reader = csv.reader(input)

for line in reader:

access\_key\_id = line[0]

secret\_access\_key = line[1]

#session\_token = line[2]

client = boto3.client('rekognition')

#aws\_access\_key\_id = access\_key\_id,

#aws\_secret\_access\_key = secret\_access\_key,

#aws\_session\_token = session\_token,

#region\_name = 'us-east-1')

s3 = boto3.client('s3')

#aws\_access\_key\_id = access\_key\_id,

#aws\_secret\_access\_key = secret\_access\_key,

#aws\_session\_token = session\_token,

#region\_name = 'us-east-1')

def main():

bucket = 'album-manager'

collection\_id = 'Collection'

fileNames=['test1.jpg','test2.jpg','test3.jpg','test4.jpg']

#fileNames = ['test6.jpg']

threshold = 70

maxFaces = 2

for fileName in fileNames:

response=client.search\_faces\_by\_image(CollectionId=collection\_id,

Image={'S3Object':

{'Bucket':bucket,

'Name':fileName}},

FaceMatchThreshold=threshold,

MaxFaces=maxFaces)

faceMatches=response['FaceMatches']

print ('Matching faces')

for match in faceMatches:

print ('FaceId:' + match['Face']['FaceId'])

print ('External Id:' + match['Face']["ExternalImageId"])

print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")

copy\_from = str(bucket+'/'+fileName)

move\_to = str(match['Face']["ExternalImageId"][:-4]+'/'+fileName)

s3.copy\_object(Bucket=bucket, CopySource=copy\_from, Key=move\_to)

print("successfully moved to " + move\_to)

if \_\_name\_\_ == "\_\_main\_\_":

main()

x = int(raw\_input())

**FLASK APP CODE**

import os  
from flask import Flask , request, render\_template  
from gevent.pywsgi import WSGIServer  
import boto3  
import csv  
  
with open("new\_user\_credentials.csv",'r') as input:  
 next(input)  
 reader = csv.reader(input)  
 for line in reader:  
 access\_key\_id = line[0]  
 secret\_access\_key = line[1]  
 session\_token = line[2]  
   
client = boto3.client('rekognition')  
 #aws\_access\_key\_id = access\_key\_id,  
 #aws\_secret\_access\_key = secret\_access\_key,  
 #aws\_session\_token = session\_token,  
 #region\_name = 'us-east-1')  
  
s3 = boto3.client('s3')  
 #aws\_access\_key\_id = access\_key\_id,  
 #aws\_secret\_access\_key = secret\_access\_key,  
 #aws\_session\_token = session\_token,  
 #region\_name = 'us-east-1')  
  
app = Flask(\_\_name\_\_)  
  
def main(filepath):  
 bucket = 'album-manager' #our bucket name in S3  
 collection\_id = 'Collection' #Our collection name of reference faces  
   
 filename = filepath #filepath of uploaded img file by user, passed as an arguement  
   
 relative\_filename = os.path.split(filepath)[1] #the is the name of the file , eg: "test1.jpg"  
 #print(relative\_filename)  
   
 fileNames = [relative\_filename]   
   
 s3.upload\_file(filename, bucket, relative\_filename) #uplaoding the file in S3 bucket  
 print("file Uploaded")  
   
 threshold = 70  
 maxFaces = 2  
  
 #finding the face in uploaded image in the reference  
 for fileName in fileNames:  
 response=client.search\_faces\_by\_image(CollectionId=collection\_id,  
 Image={'S3Object':  
 {'Bucket':bucket,  
 'Name':fileName}},  
 FaceMatchThreshold=threshold,  
 MaxFaces=maxFaces)  
 recognized\_person\_name = "Not Detected"   
 #printing the matched face details in console only   
 faceMatches=response['FaceMatches']  
 print ('Matching faces')  
 for match in faceMatches:  
 print ('FaceId:' + match['Face']['FaceId'])  
 print ('External Id:' + match['Face']["ExternalImageId"])  
 print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")  
   
 #to move the file wihtin s3 bucket, from the bucket to respective folder  
 copy\_from = str(bucket+'/'+fileName)   
 move\_to = str(match['Face']["ExternalImageId"][:-4]+'/'+fileName)  
 recognized\_person\_name = str(match['Face']["ExternalImageId"][:-4])  
 s3.copy\_object(Bucket=bucket, CopySource=copy\_from, Key=move\_to)  
   
 print("successfully moved to " + move\_to)  
 return recognized\_person\_name #returning the name of the person whose face was detected  
 #to be used by flask app, to print on the web page.  
   
   
#opening up the base html template.   
@app.route('/')  
def index():  
 return render\_template('base.html')  
  
#whenever the predict button is clicked, the chosen file is uplaoded to uploads folder  
#it then calls our main function, psases the uploaded file's path as arguement, and print's the  
#returned name as output on web screen  
@app.route('/predict',methods = ['GET','POST'])  
def upload():  
 if request.method == 'POST':  
 f = request.files['image']  
 print("current path")  
 basepath = os.path.dirname(\_\_file\_\_)  
 print("current path", basepath)  
 filepath = os.path.join(basepath,'uploads',f.filename)  
 print("upload folder is ", filepath)  
 f.save(filepath)  
   
 text = main(filepath) #calling our main function, passing stored file's  
 #filepath as arguement to it  
 return text  
   
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug = True, threaded = False)