**Amazon Rekognition - SmartHealth Use Case**

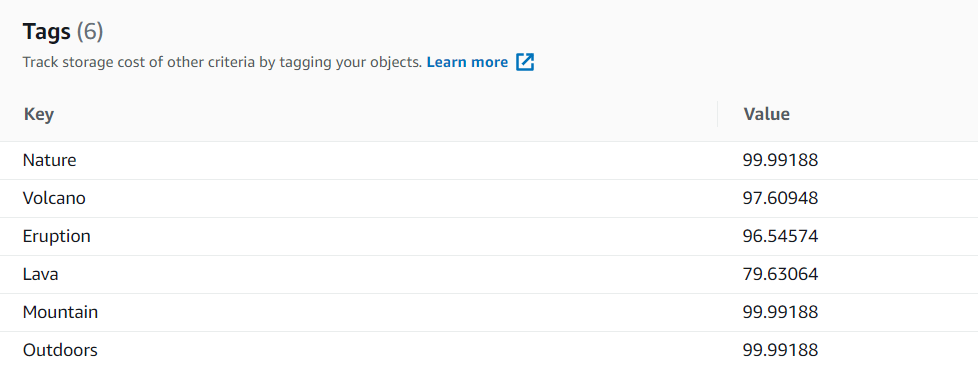
The problem that SmartHealth was facing is that they have a smart glass which was capturing real-time images and the labelling of these images will help the blind to navigate the path conveniently. They are currently using Azure services for computer vision for eg. Person wearing the glasses will get to know a car is in front. The client wanted to replace Azure with AWS so at the place of API calling changes need to be done for this replacement. The default request parameters for this function in AWS are image in base 64 format and S3 location where the image is stored. This base 64 conversion and saving to S3 part is integrated in lambda code only so that from front-end only image parameter would be suffice.

We create an AWS Lambda function that automatically tags digital assets located in an Amazon S3 bucket. The Lambda function reads all objects in a given Amazon S3 bucket. For each object in the bucket, it passes the image to the Amazon Rekognition service to generate a series of labels. Each label is used to create a tag that is applied to the image. After you execute the Lambda function, it automatically creates tags based on all images in a given Amazon S3 bucket and applies them to the images.

For example, assume we run the Lambda function and we have this image in an Amazon S3 bucket.

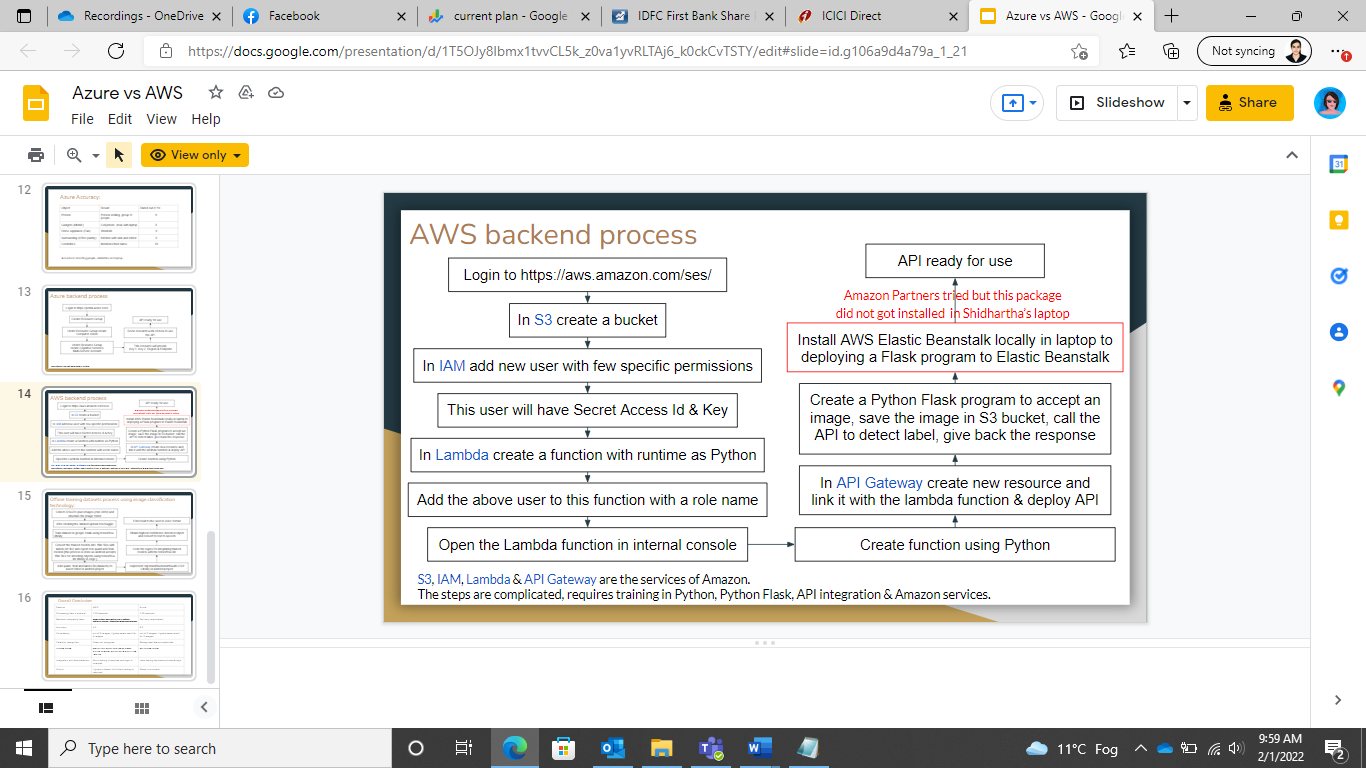


The application then automatically creates tags and applies them to the image.

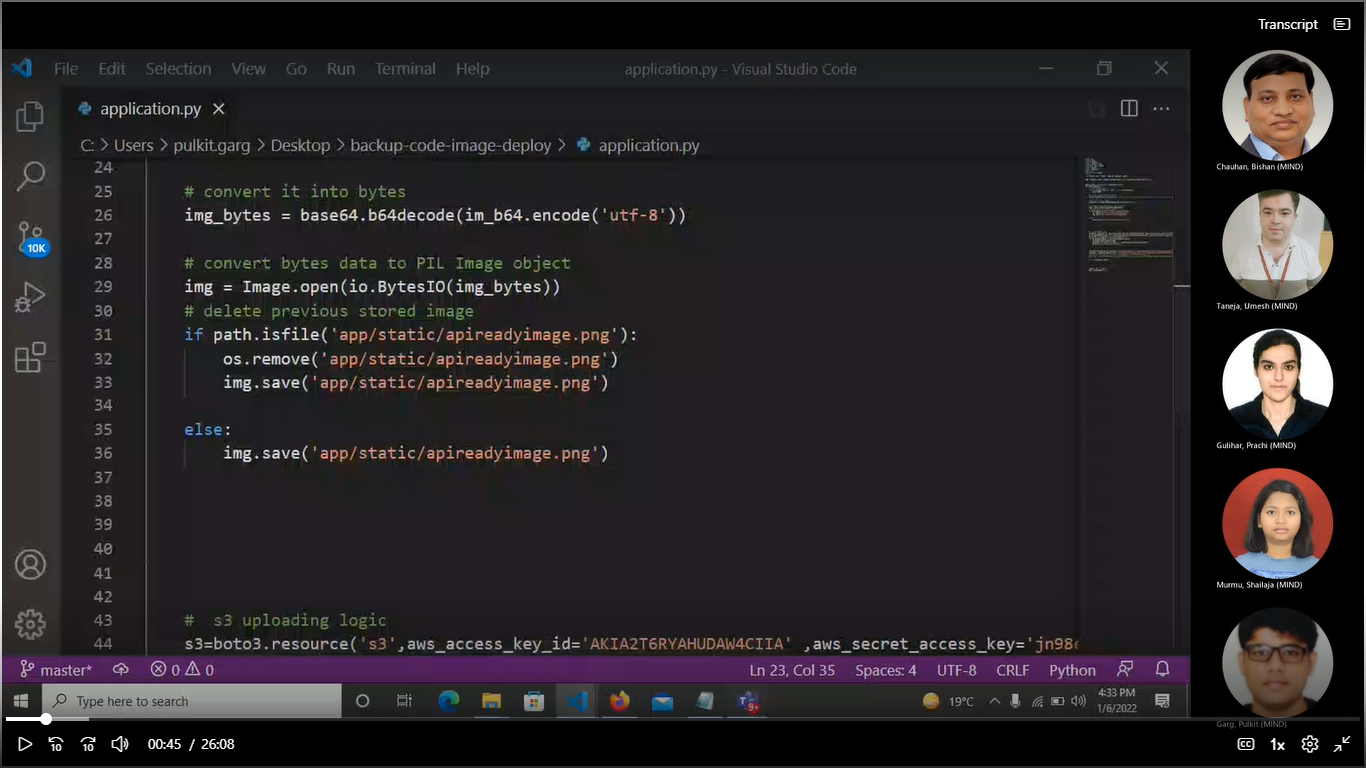


The basic idea is to capture the image and pass it to AWS. And get the detected image result within some milliseconds. AWS API is provided using Amazon API Gateway ( created Lambda function having attached code) – to use for amazon recognition – detect\_label, which accepts an image and returns the required information i.e. Name and confidence. Now, this will work same as that of azure service but with better detection capability, and now this API can be integrated with mobile app code, to detect images name and Azure is replaced AWS service.

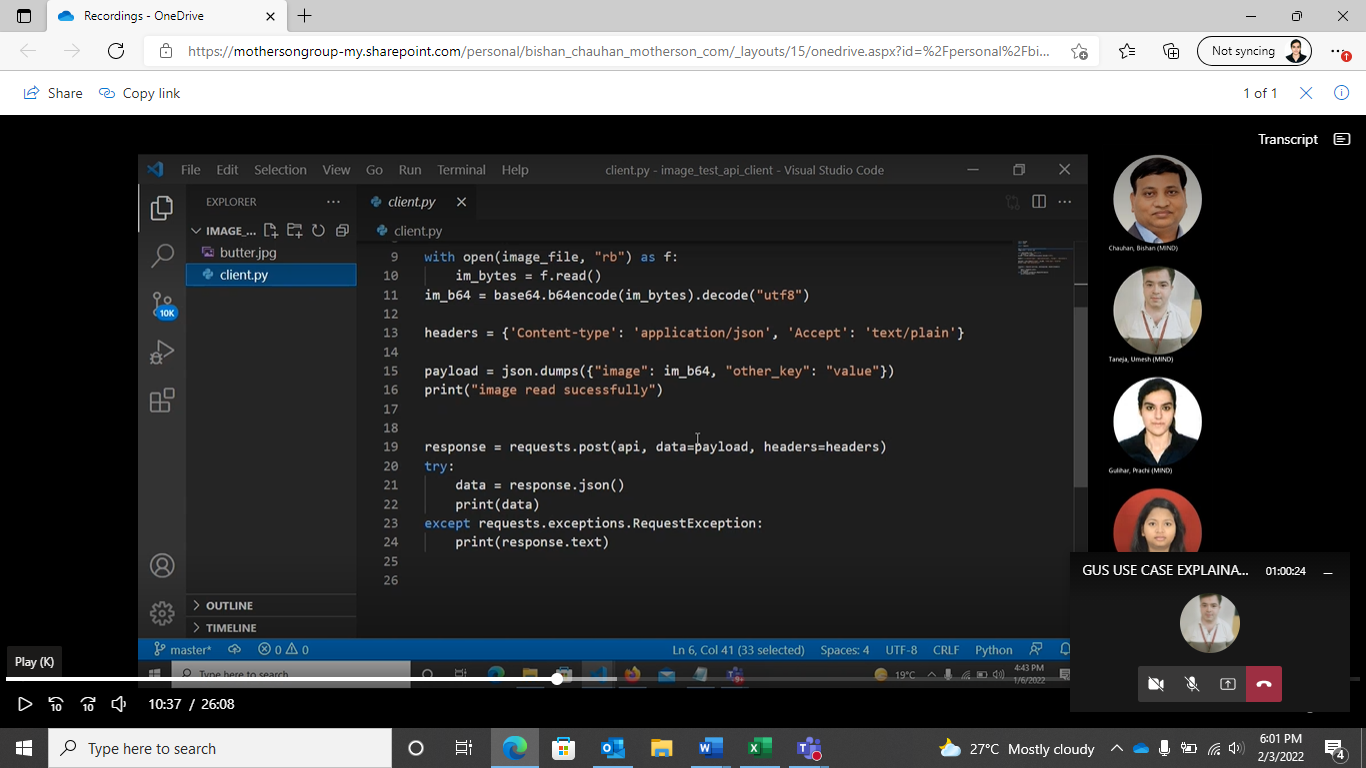
AWS API calling process



Code for conversion of image to base 64 format



Upload to S3 function



The below code is for detecting the image which is on Aws S3 using Aws Rekognition service and return only two features 'labels' and 'confidence' according to client requirements and give it in API response.

import json

def event\_handler(event,context):

bucket='demobucketimagetest'

my\_image\_name = '4129iYx95yL.\_SX425\_.jpg'

#my\_image\_name = event['queryStringParameters']['imageid']

client=boto3.client('rekognition',aws\_access\_key\_id='AKIA2T6RYAHUDAW4CIIA' , aws\_secret\_access\_key='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'))

response = client.detect\_labels(Image={'S3Object':{'Bucket':bucket,'Name':my\_image\_name}}, MaxLabels=1)

print('Detected labels for ' + photo)

return len(response['Labels'])

json\_body = {}

for label in response['Labels']:

temp\_dict = {}

label\_name = label['Name'])

confidence = label['Confidence']

temp\_dict['Label'] =label\_name

temp\_dict['Confidence'] = confidence

json\_body['Results'] = temp\_dict

apiresponse = {}

apiresponse['statusCode'] = 200

apiresponse['headers'] = {}

apiresponse['headers']['Content-Type'] = 'application/json'

apiresponse['body'] = json.dumps(json\_body)

return apiresponse

Some of the most frequently used APIs for Rekognition are as follows:

1. **Indexfaces function-**

IndexFaces operation is used to detect faces in an image and add them to a collection. For each face detected, Amazon Rekognition extracts facial features and stores the feature information in a database (collection in DynamoDB)

def index\_faces(bucket, key):

    response = rekognition.index\_faces(

        Image={"S3Object":

            {"Bucket": bucket,

            "Name": key}},

            CollectionId="family\_collection")

    return response

1. **Compare faces function call-**

Two parameters- source and target are there.

|  |  |
| --- | --- |
| response = rekognition.compare\_faces( |  |
|  | SourceImage={ |
|  | "Bytes": image1Bytes, |
|  | }, |
|  | TargetImage={ |
|  | "Bytes": image2Bytes, |
|  | }, |
|  | SimilarityThreshold=90, |
|  | QualityFilter="HIGH", |
|  | ) |

1. **Detect face function call-**

Single image is passed as Bytearray.

|  |
| --- |
| imageBytes = bytearray(document.read()) |
|  |  |
|  | # Amazon Rekognition client |
|  | rekognition = boto3.client("rekognition") |

|  |
| --- |
| response = rekognition.detect\_faces( |
|  | Image={"Bytes": imageBytes}, |
|  | ) |

Detectface API responses include AgeRange, Beard, Emotions, Eyeglasses, EyesOpen, Gender, MouthOpen, Mustache, Smile, Sunglasses

The CompareFaces operation and the search faces operations differ as follows:

* The CompareFaces operation compares a face in a source image with faces in the target image. The scope of this comparison is limited to the faces that are detected in the target image.
* SearchFaces and SearchFacesByImage compare a face (identified either by a FaceId or an input image) with all faces in a given face collection. Therefore, the scope of this search is much larger.