**1.INTRODUCTION**

1.1 **Overview**

* Purpose of the Project
* Existing Problem
* Proposed Solution
* Block Diagram
* Hardware/Software Designing
* Experimental Investigations
* Flowchart
* Result
* Advantages & Disadvantages
* Applications
* Conclusions
* Future Scope
* Bibliography

1.2 **Purpose**

* Access control is done by using smart Analytic device. It verifies the entry of the person.
* The Smart device verifies the persons entering into the industry.
* The details of the person are being taken and uploaded into the cloud.
* We can Restrict the entry of the unknown persons and we can restrict the persons who are not following the safety measures by using this IoT device.

**2.LITERATURE SURVEY**

2.1  **Existing Problem**

The problem with the present existing device is it cannot able to identifies the safety measures of the persons it just identifies the entry of the persons.

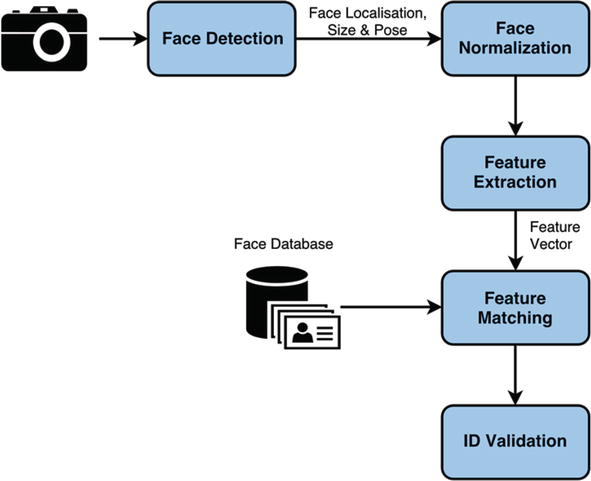
2.2 **Proposed Solution**

We can make use of IoT Analytics in Access Control, such that during working hours in the industry we can identify the persons who are following the safety measures and who are not following.

Also, with the usage of IoT, automatically, the details of the person are taken and we can restrict them.

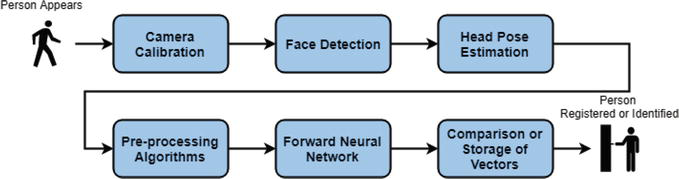
**3.Theoretical Analysis**

**3.1 Block Diagram**



**3.2 Hardware/Software Designing**

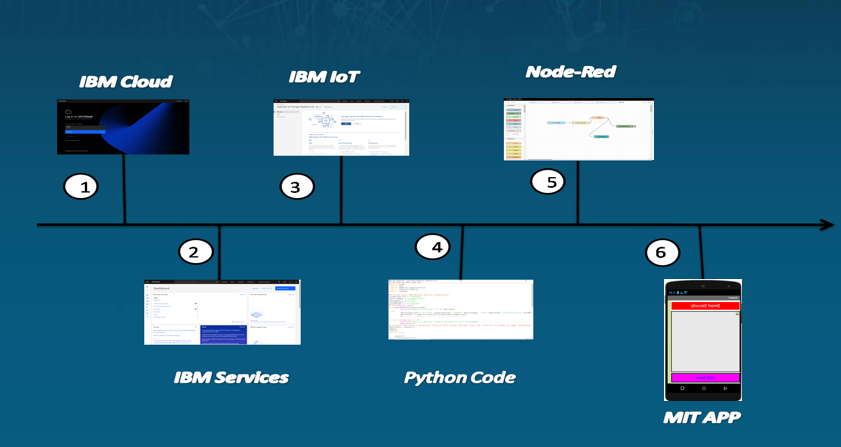
The Software designing involves genera We used IBM Cloud Services to create Internet of Things platform. In IoT platform we create a virtual Raspberry Pi device. After creating the design we get the device credentials. We use these credentials in Python program then we integrated the Node-Red platform with IoT. With the help of MIT APP Inverter we designed the app & integrated with the Node-Red to observe the values.



**4.Experiment Investigation**

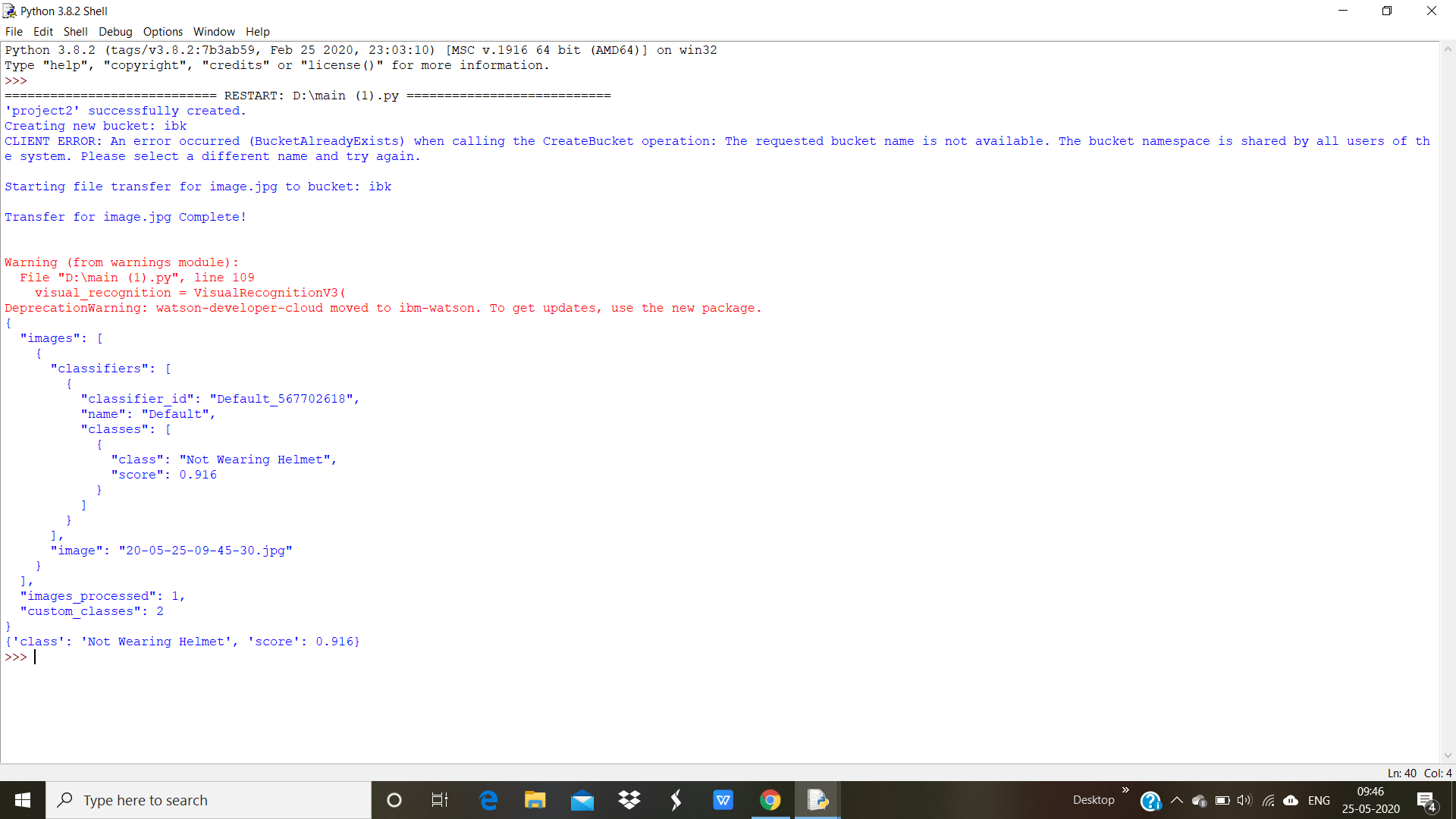
To complete our project work we collected the required data from Google & research papers. After getting the complete knowledge we work according to our roles in the project. At first we create the IBM Cloud account then we created the Internet of Things Platform after we wrote a python code in IDLE to connect IBM IoT Platform. Next we created the Node-Red Services. This service helps us to show virtual flow graphs. We connect Node-Red to IBM IoT to get the current, voltage and calculated bills. From Node-Red we send values to the MIT APP. From app we can view the details of the person .

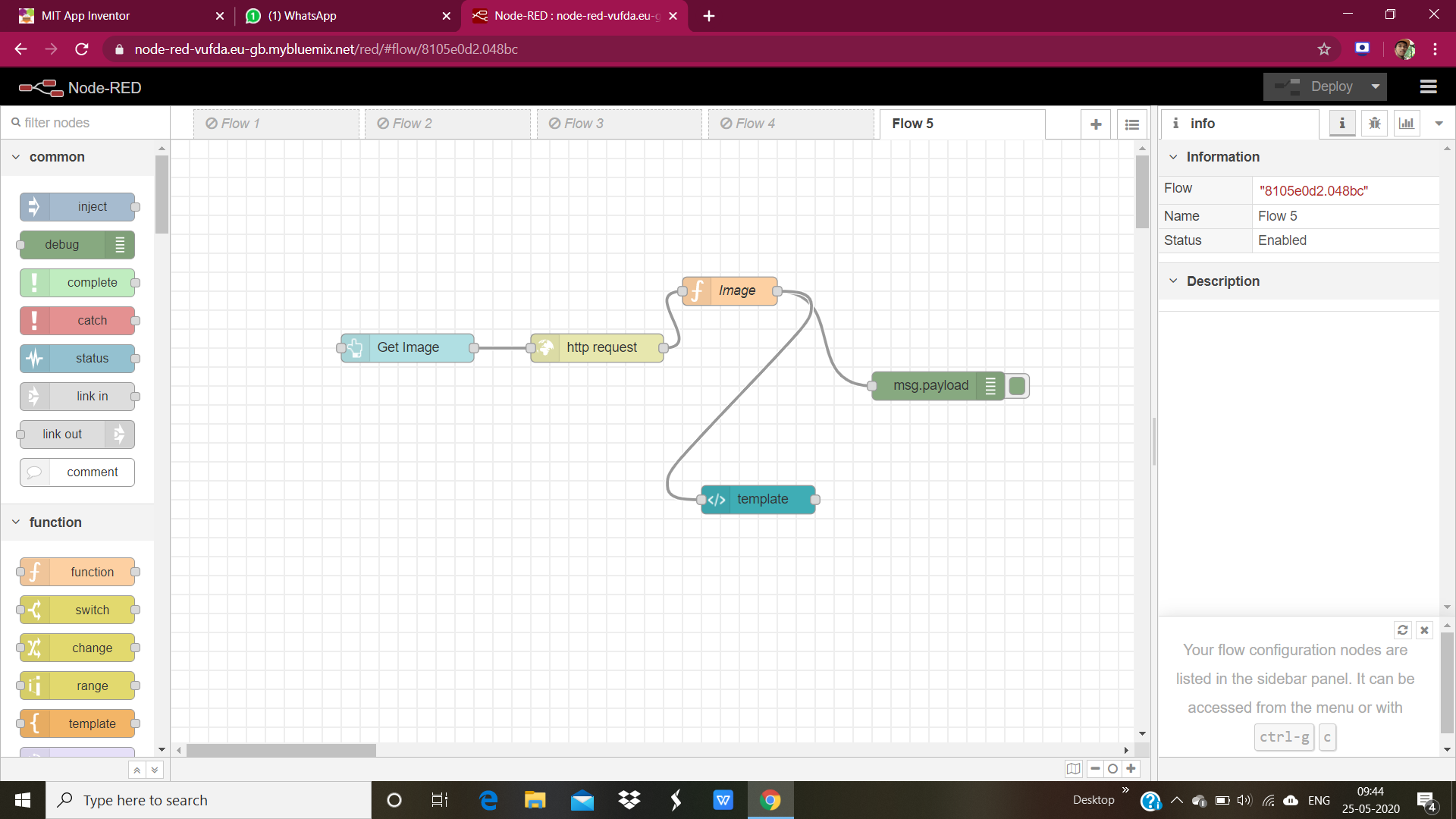
**5.FLOWCHART**

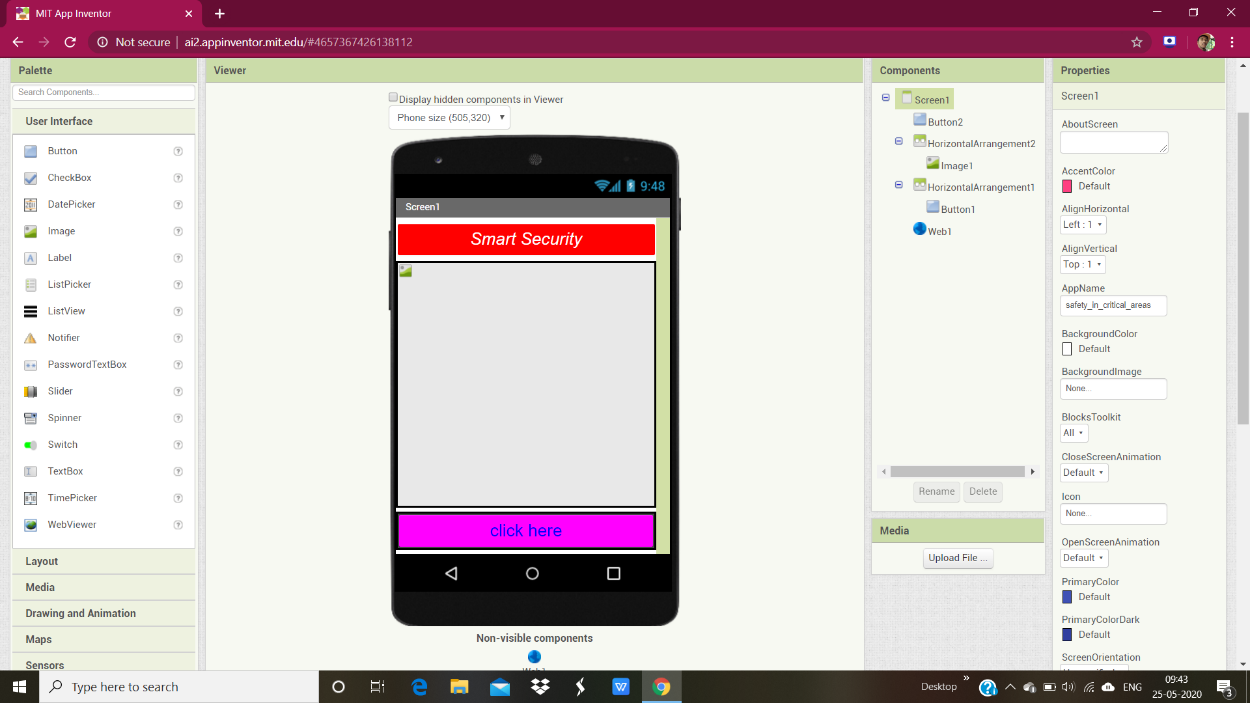
****

**6 RESULT**

**Python Code:**

****

**Node-Red:****MIT APP:**

****

**7 ADVANTAGES & DISADVANTAGES**

**Advantages:**

1) Increase ease of access for employers

2) Keep tack of who comes and goes

3) Protect against unwanted visitors

4) create safe work Environment

5) Reduce Theft and Accidents

6) Easy Monitoring

**Disadvantages:**

1) Access control systems can be hacked.

**8. APPLICATIONS**

1) Large Industries

2) In Airports

3) Government Sectors.

**9. CONCLUSIONS**

Regarding the camera and its calibration, the industrial camera had a better performance compared to the webcam as the calibration method presented focus on the best face image that can be acquired. As for software, the detection algorithm presented a good performance.

**10. FUTURE SCOPE**

The future work goes through the implementation of the solution in larger scales where more people would use it. Until then, the training of new neural networks using the preprocessing techniques is presented, and the study of new alternatives for cameras is on the agenda.

**11. BIBLIOGRAPHY**

<https://cloud.ibm.com/registration>

<https://cloud.ibm.com/catalog/services/watson-studio>

[http://Ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu/)

<https://flows.nodered.org/node/node-red-dashboard>

<https://developer.ibm.com/recipes/tutorials/ui-dashboard-for-iot-device-data-using-node-red/>

<https://appinventor.mit.edu/>

**APPENDIX**

**A. Source Code**

import cv2

import datetime

import numpy as np

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import time

#Provide CloudantDB credentials such as username,password and url

client = Cloudant("8f03c090-ff65-4bfb-8ff7-7bb3f526fd5b-bluemix", "9f35def67e3f66d942b35c52ce21e86c1396e081a735578e228a9b509f7297d6", url="https://8f03c090-ff65-4bfb-8ff7-7bb3f526fd5b-bluemix:9f35def67e3f66d942b35c52ce21e86c1396e081a735578e228a9b509f7297d6@8f03c090-ff65-4bfb-8ff7-7bb3f526fd5b-bluemix.cloudantnosqldb.appdomain.cloud")

client.connect()

#Provide your database name

database\_name = "project2"

my\_database = client.create\_database(database\_name)

if my\_database.exists():

print(f"'{database\_name}' successfully created.")

img=cv2.VideoCapture(0)

while True:

ret,frame=img.read()

global imgname

cv2.imshow("Employee\_Face",frame)

imgname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M-%S")

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

k=cv2.waitKey(1)

#waitKey(1)- for every 1 millisecond new frame will be captured

if k==ord('q'):

#release the camera

img.release()

#destroy all windows

cv2.destroyAllWindows()

break

# Constants for IBM COS values

COS\_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints

COS\_API\_KEY\_ID = "P7vluJ26j6F1Li1\_6ddLIZ8K15kHHrJcaMj5cD75Iy\_-" # eg "W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"

COS\_AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"

COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/58ac0db086aa48898bdfffdd48180291:45fcb933-75ac-4cc7-a7c9-244767458ba2::"

# Create resource

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_RESOURCE\_CRN,

ibm\_auth\_endpoint=COS\_AUTH\_ENDPOINT,

config=Config(signature\_version="oauth"),

endpoint\_url=COS\_ENDPOINT

)

def create\_bucket(bucket\_name):

print("Creating new bucket: {0}".format(bucket\_name))

try:

cos.Bucket(bucket\_name).create(

CreateBucketConfiguration={

"LocationConstraint":"jp-tok-standard"

}

)

print("Bucket: {0} created!".format(bucket\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to create bucket: {0}".format(e))

create\_bucket("ibk")

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

# set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

# set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size

)

# the upload\_fileobj method will automatically execute a multi-part upload

# in 5 MB chunks for all files over 15 MB

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

multi\_part\_upload("ibk", "image.jpg", imgname+".jpg")

json\_document={"link":COS\_ENDPOINT+"/"+"ibk"+"/"+"image.jpg"}

new\_document = my\_database.create\_document(json\_document)

import json

from watson\_developer\_cloud import VisualRecognitionV3

visual\_recognition = VisualRecognitionV3(

'2018-03-19',

iam\_apikey="0MmNQwIZEKyukQygp1lp312e7R0yFdZRHaqqntiOvVGo")

with open(imgname+'.jpg', 'rb') as images\_file:

classes1 = visual\_recognition.classify(

images\_file,

threshold='0.6',

classifier\_ids='Default\_567702618').get\_result()

'print(type(classes1))'

print(json.dumps(classes1, indent=2))

print(classes1["images"][0]["classifiers"][0]["classes"][0])

a=classes1["images"][0]["classifiers"][0]["classes"][0]["class"]

from ibm\_watson import TextToSpeechV1

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

from playsound import playsound

authenticator = IAMAuthenticator("iwnspiBaK117WrVP5eTCljivlC2brGwYg\_PKEt1MlVFP")

text\_to\_speech = TextToSpeechV1(

authenticator=authenticator

)

text\_to\_speech.set\_service\_url("https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/b0fb2355-1377-4242-9015-fccdd17051d8")

with open('project.mp3', 'wb') as audio\_file:

if(a=="Wearing helmet"):

audio\_file.write(

text\_to\_speech.synthesize(

f'Employee is {a} and allow inside',

voice='en-US\_AllisonVoice',

accept='audio/mp3'

).get\_result().content)

else:

audio\_file.write(

text\_to\_speech.synthesize(

f'Employee is {a} and do not allow inside',

voice='en-US\_AllisonVoice',

accept='audio/mp3'

).get\_result().content)

import requests

r = requests.get('https://www.fast2sms.com/dev/bulk?authorization=XrBTQsmZxWGqYn98k5SK0J17zRcufvyE4HeVgPl62hMNdAwj3ob3TEDtGJXY6I4gBAcRxpusyPovONZ1&sender\_id=FSTSMS&message=Employee is not wearing helmet&language=english&route=p&numbers=9676847472')

playsound('project.mp3'