Smart bridge Guided Project

Smart Guest Identifier With Remote Access Management

Group No: 21

Group members:

P.Sri Bala Harish harish.18bec7019@vitap.ac.in
U.Harshith harshith.18bec7018@vitap.ac.in
D.Sree Charan sreecharan.18bec7096@vitap.ac.in

CONTENTS:

1. 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, and works by pinpointing and measuring facial features from a given image.

Using this technology we can identify the person which is useful for security purposes and to give remote access to the identified person in this project.

a. Overview

Sometimes we will be in our offices or we may go out and guests may visit us. We may miss them as we don't have any idea that someone has come to meet us.

Smart Guest Identifier With Remote Access Management is a device which is integrated at the entrance. There will be video streaming whenever any person comes near the entrance. It will detect the person using IBM Watson visual recognition services. And captures a pic and then that picture is sent to the mobile application.

In the mobile application, the owner can see the person who is in front of their door. And they can also open the door through the mobile application if they want to give access to the visitors.

b. Purpose

Smart guest identifiers can be accessed remotely and makes your home more secure. Using the mobile application the administrator can give access to the guest who ever comes to their home. The door opens when a person is recognised if the guest is a known person. So you can give access only to the people you know and the unknown person cannot enter without the administrator access, this system is more secure to use for security purposes.

2. Literature Survey

a. Existing problem

The security aspect is the highest concern of IoT connected entities. The data can be personal, enterprise or consumer. To reach an acceptable implementation for the smart door lock (SDL), security should be taken as a major challenge. We can summarize the problems into different questions

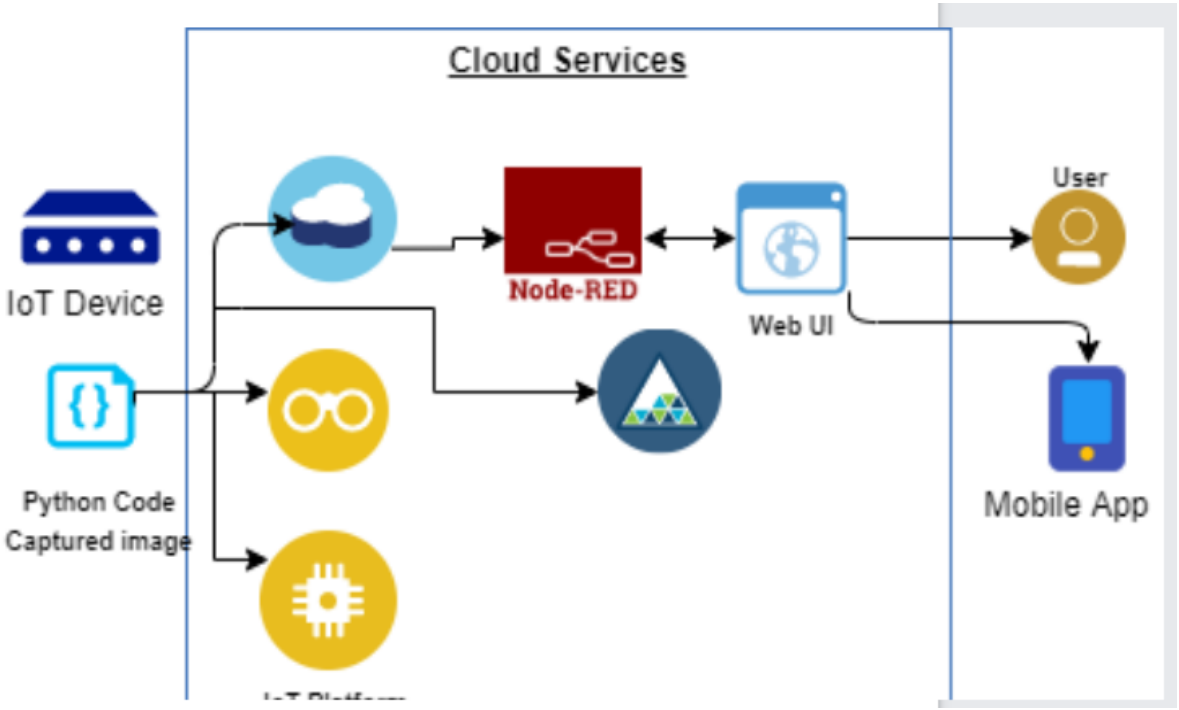
1. How to unlock a door remotely by the administrator when a person wants to enter?
2.How to give access to authorized persons?
3. Which connection protocols can be used in the product and offer the ability to authenticate, and access control?
4. How do we generate an access token for the user that has the privilege to unlock the door and how do we secure this token from being exposed?
5. How to give access to an unknown person if the administrator is not using the mobile application?

b. Proposed solution

- Whenever someone arrives at your door this smart device will detect the face and capture the images.
- The captured images will be sent to the admins mobile app through the cloud platform.
- The communication protocol we used in this project is hypertext transfer protocol.
- If the admin is not in the home and if he wants to open the door, he can open the door by pressing the button in the mobile application
- If the person is recognized as the authorized person the door will be opened automatically
- There is one emergency button if there are any emergencies the person can press the button to send alerts to the concerned persons.

3. Theoretical Analysis

a. Block diagram

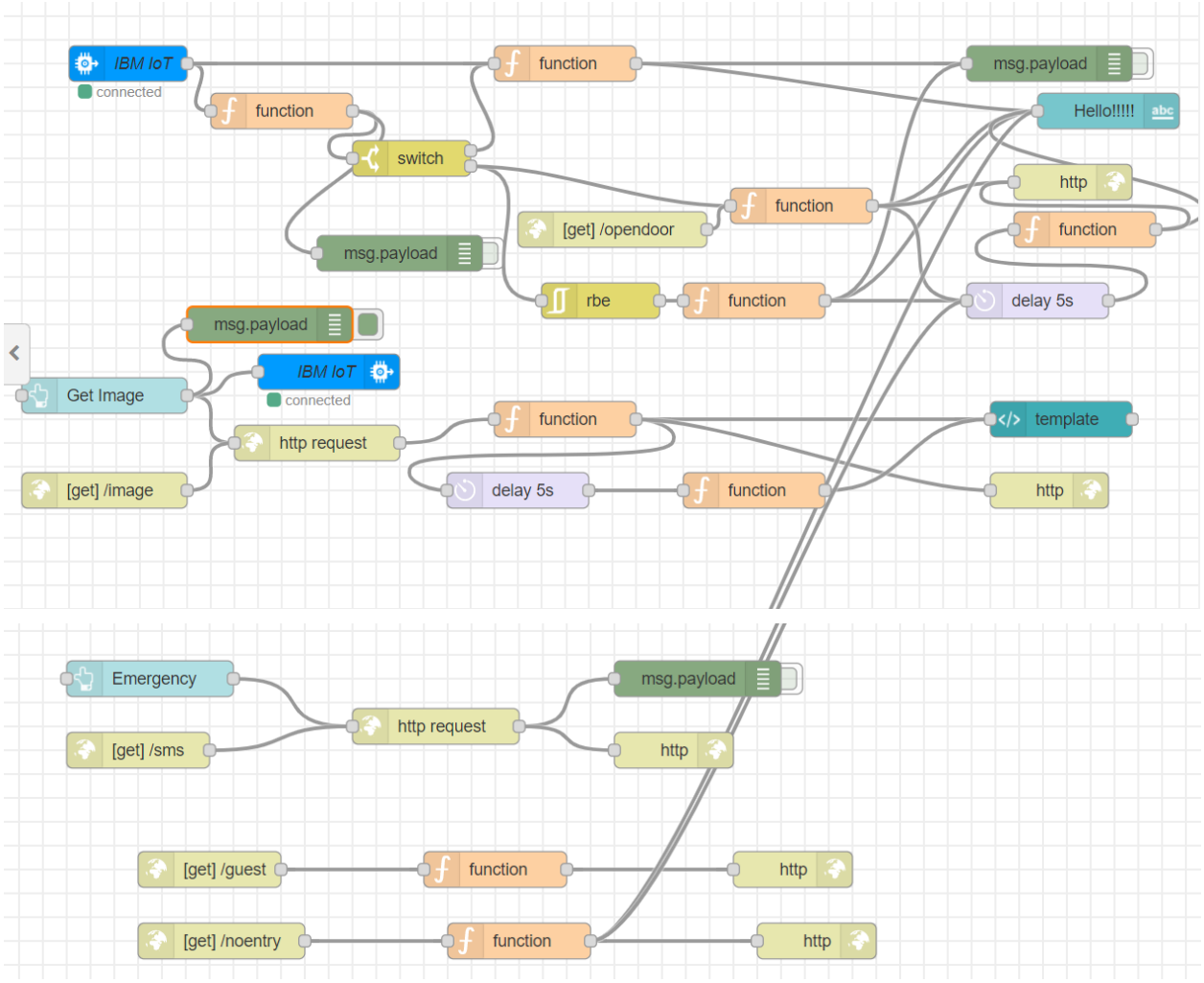


IoT PLATFORM

b. Hardware / Software designing

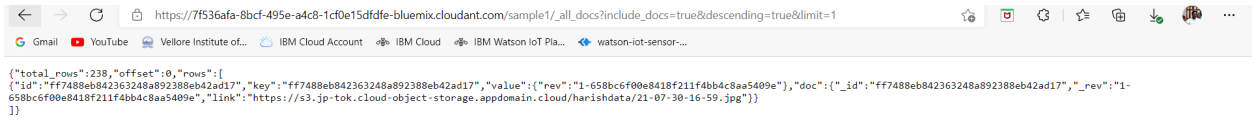
The Python code is basically about when a person is in front of the camera it will capture the image and recognize the face and show if the person was known with his/her name and the door will open automatically. Otherwise it shows the person was unknown and it doesn't give access to open the door.In This code we are using sdks to send the captured images to respective cloud object storage and cloudant database.And Using IBM IoT to send the data which we get output from code to node red flow.

Node Red flow:



Node Red flow:

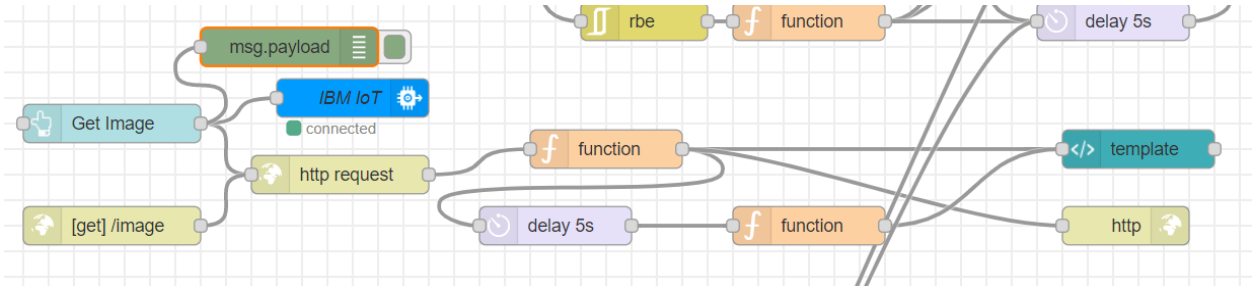
To get the recent image that was captured by primary camera is from these links: https://7f536afa-8bcf-495e-a4c8-1cf0e15dfdfe-bluemix.cloudant.com/sample1/_all_docs?include_docs=true&descending=true&limit=1



Note: only a recent image will come that cloudant db must and should be active.

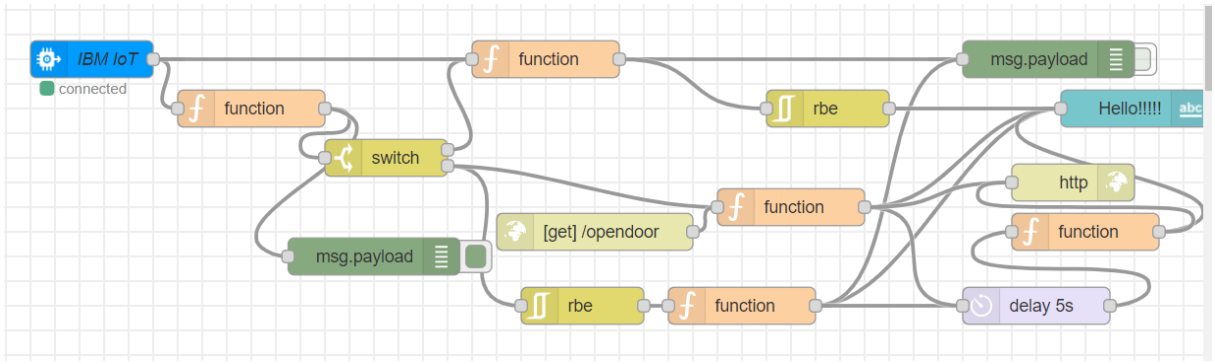
To get Image in Web ui:

Flow was:

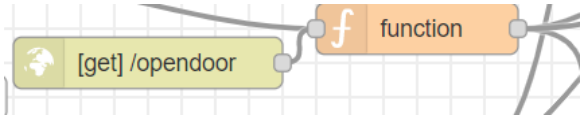


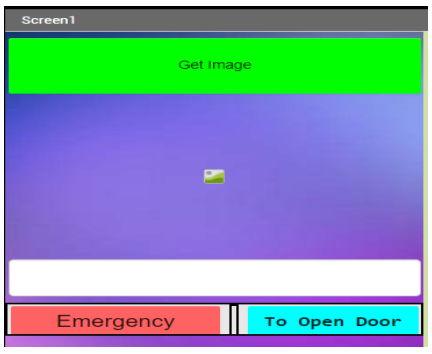
Http in and response nodes for MIT App inventor.

Node Red flow if a person is detected then it needs to open the door otherwise it says unknown person is detected and if you want to open press open button.

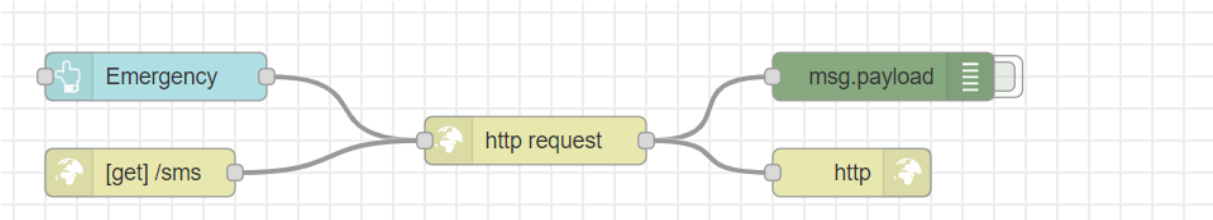


To open the door if you are not at home then build a button in mit app to open door instruction:





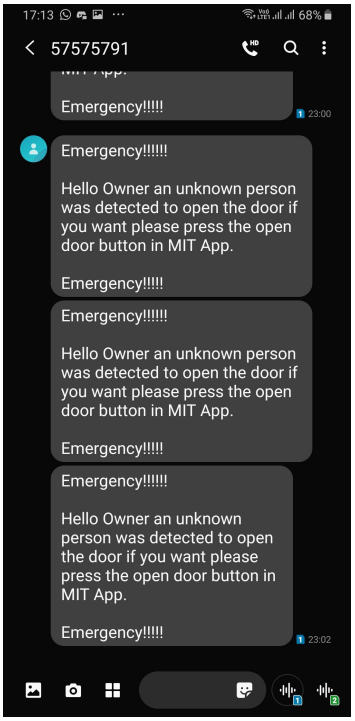
There is one emergency button if there are any emergencies the person can press the button to send alerts to the concerned persons.



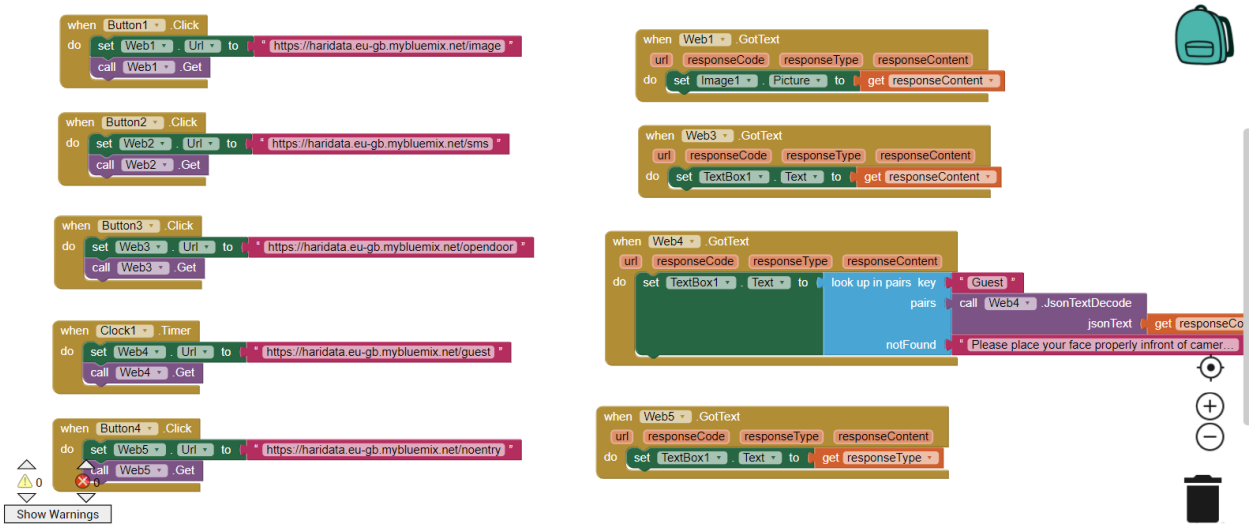
MIT APP:

If emergency Button is clicked in MIT APP:

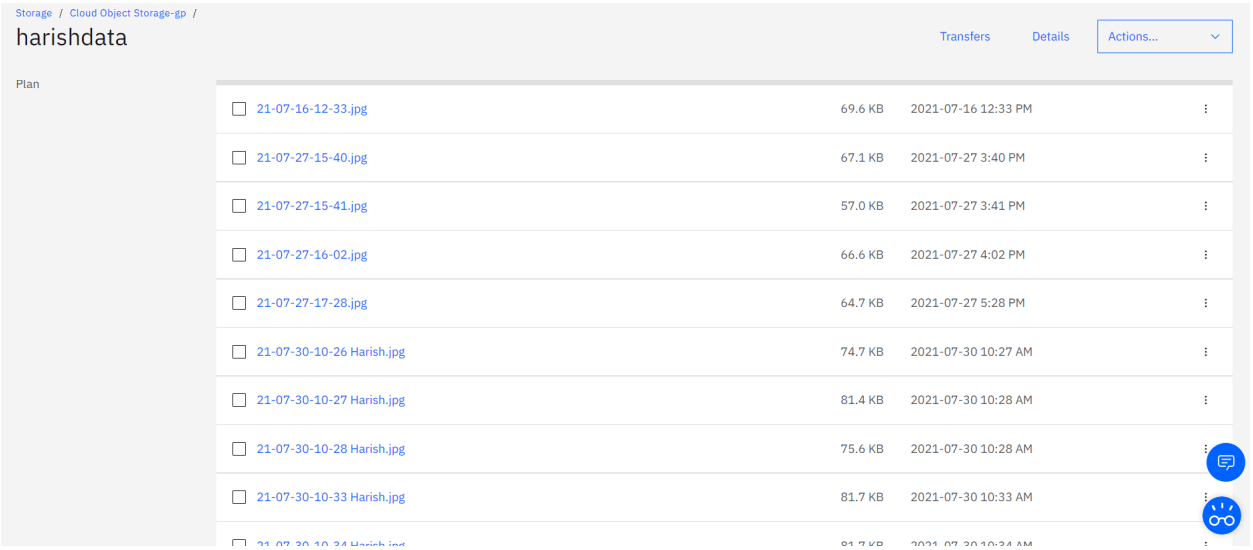
Then the respective owner will get this message.



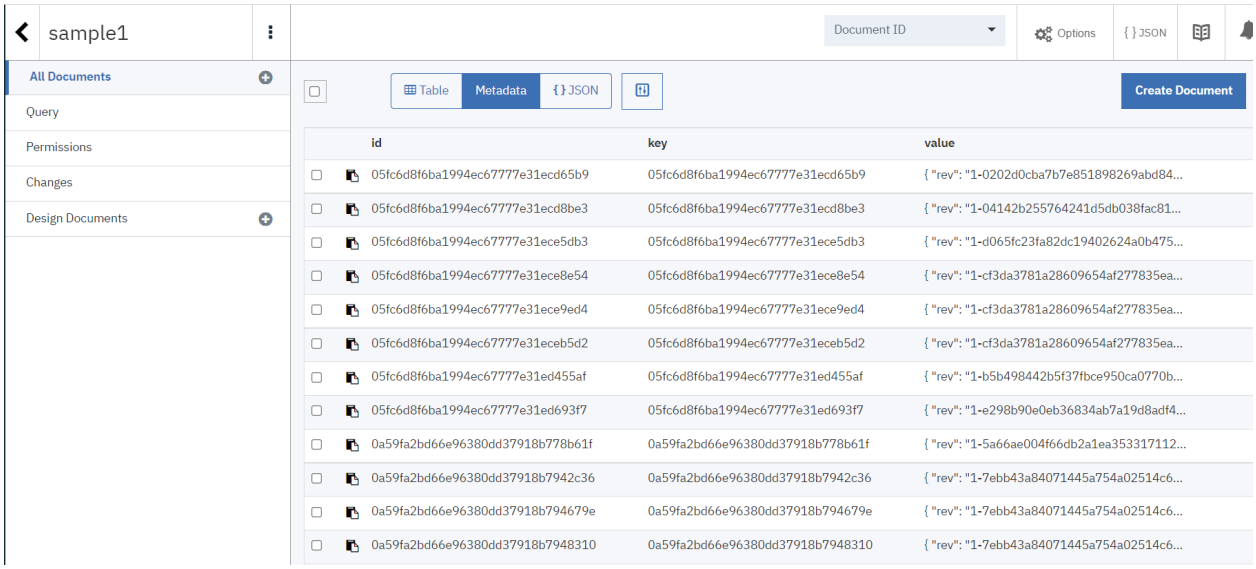
MIT APP Blocks:



Cloudant object storage and Cloudant db:

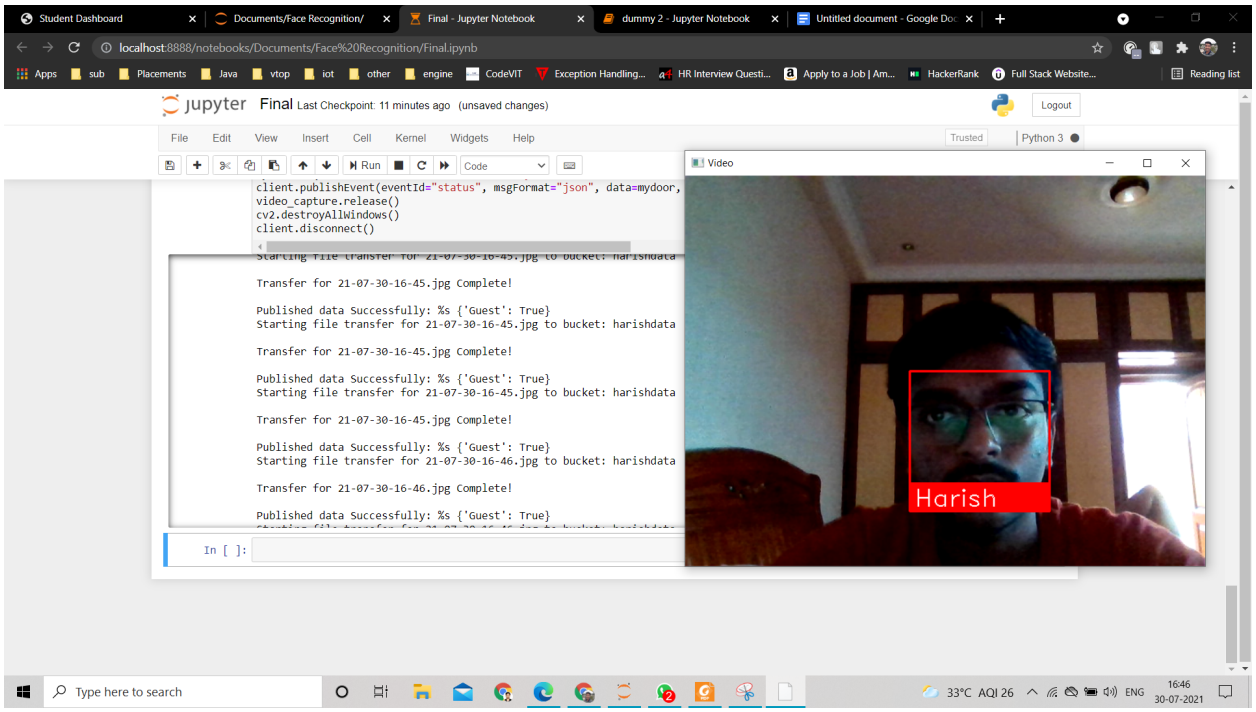


Images captured in cloudant object storage and in cloudant DB.
In cloudant db the data will be stored in Json format.

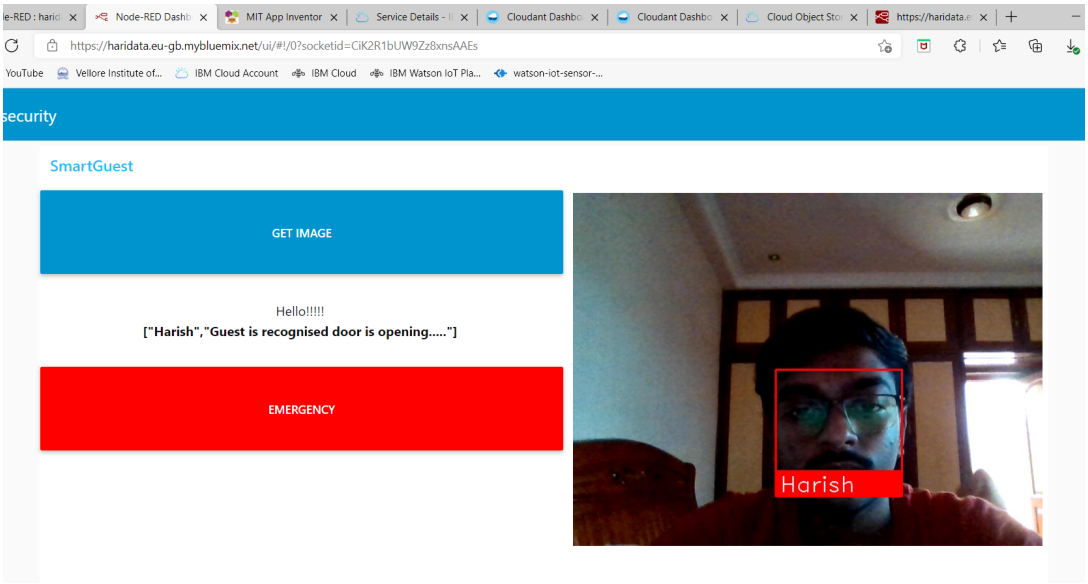


4. Experimental Investigations

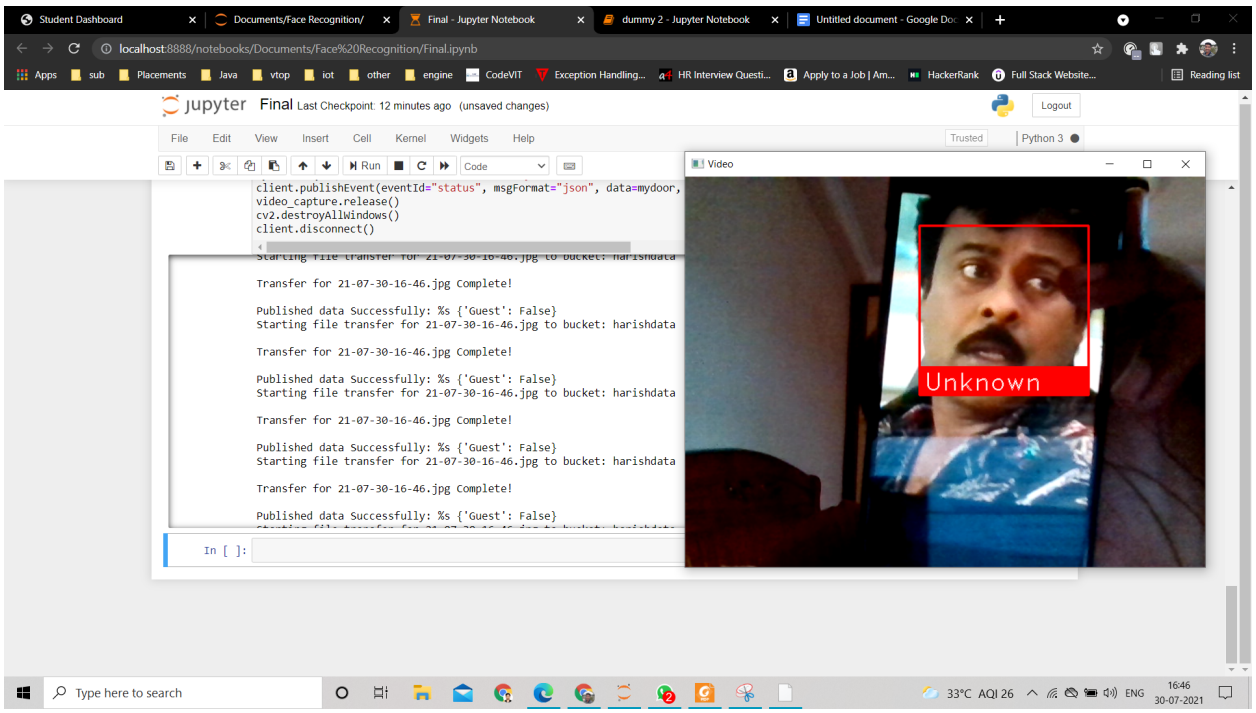
If Known Guest Comes:



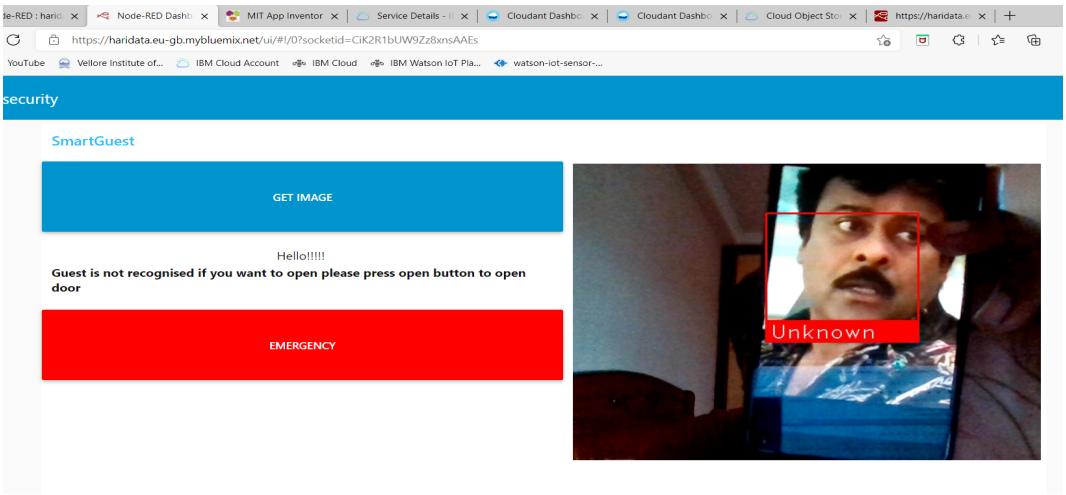
In Web UI:



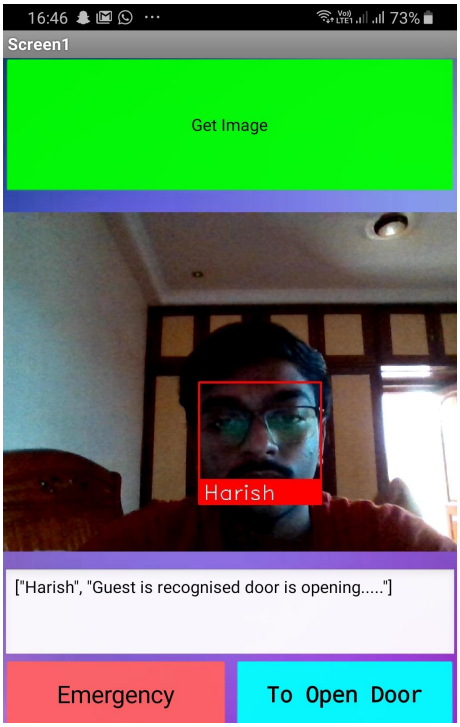
For unknown guest:



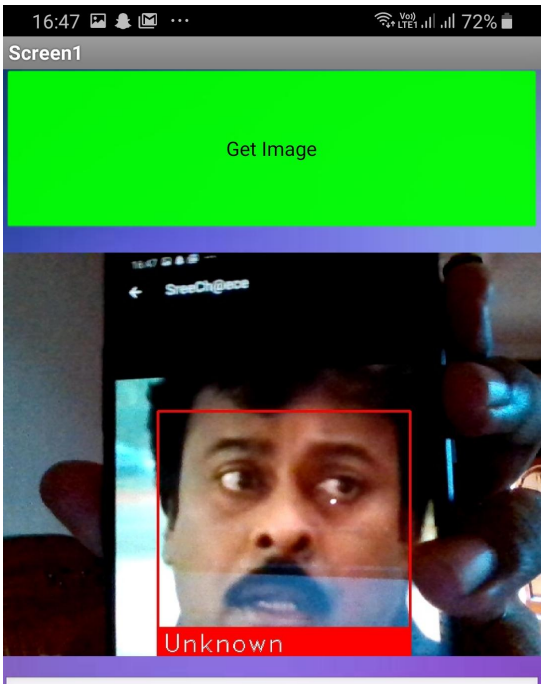
In Web UI:



In MIT APP:

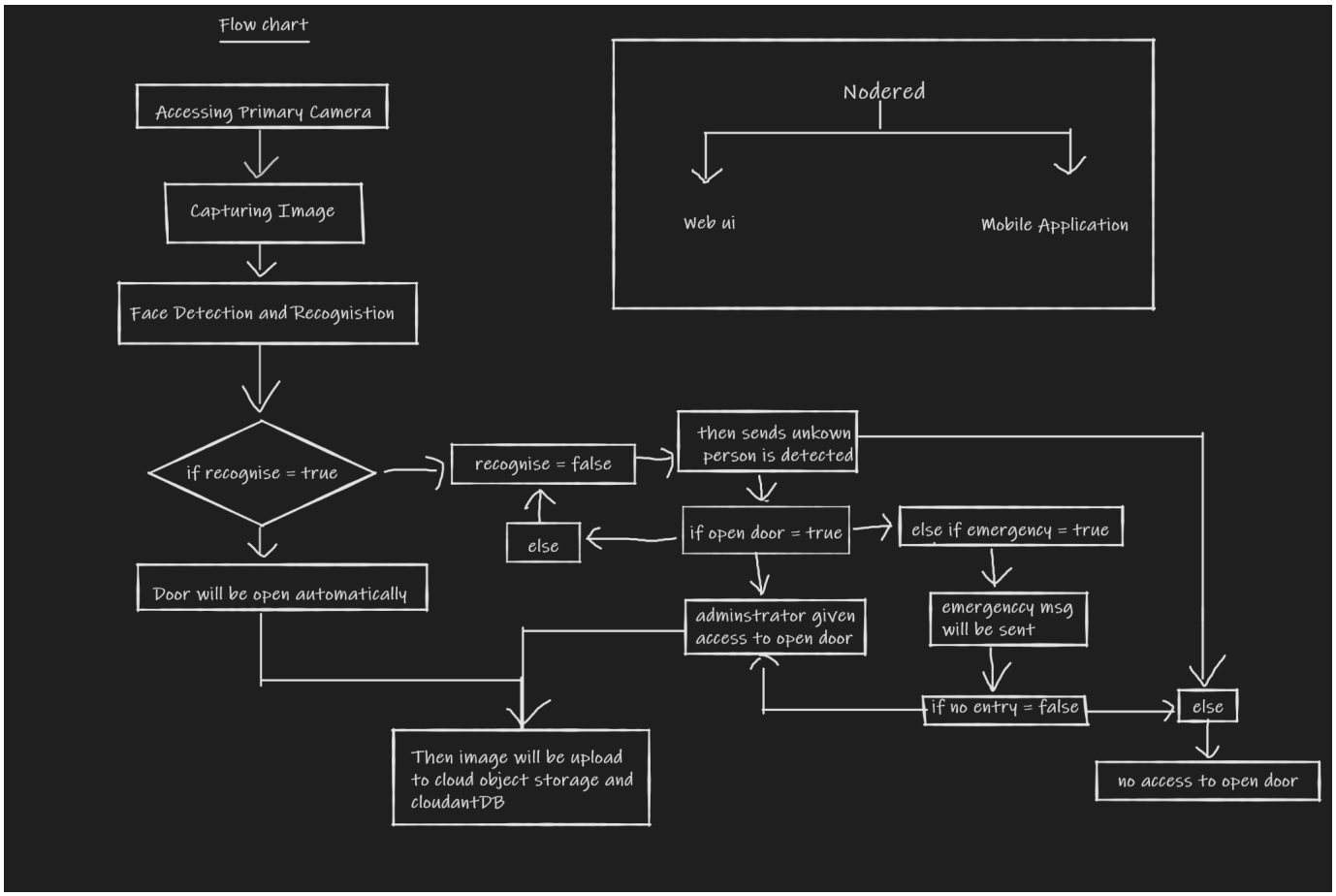


Known person



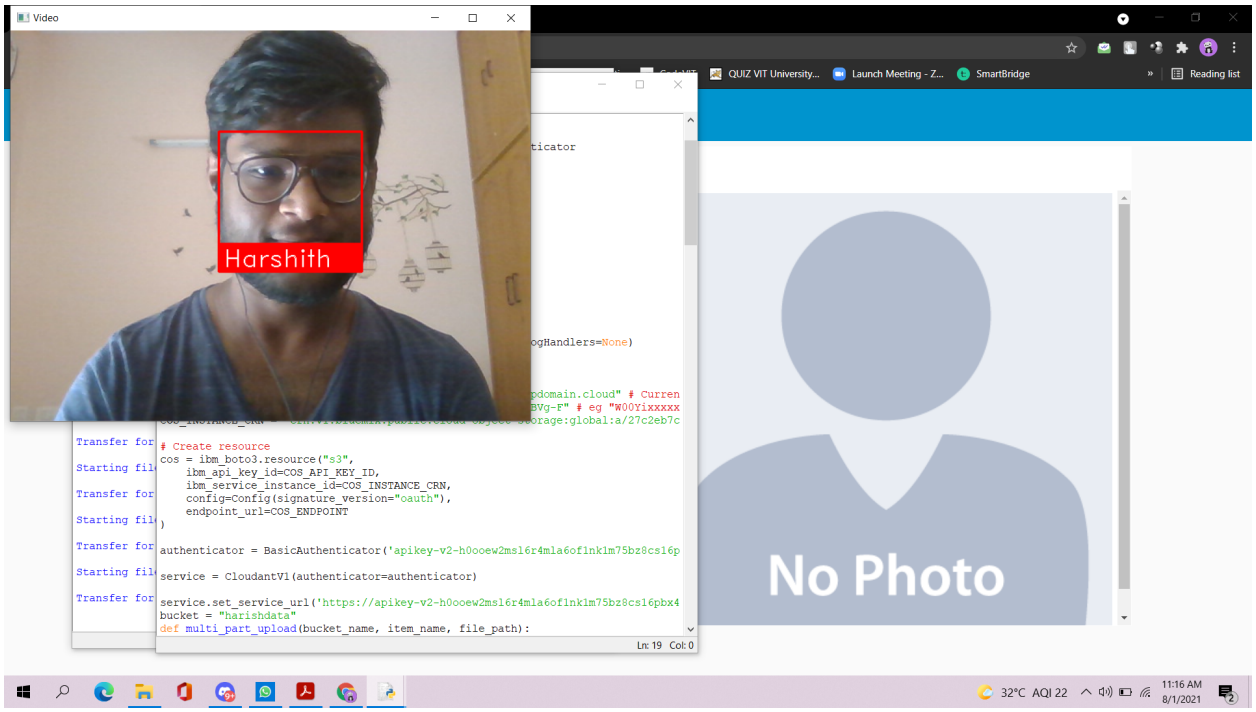
Unknown Person

5. Flowchart

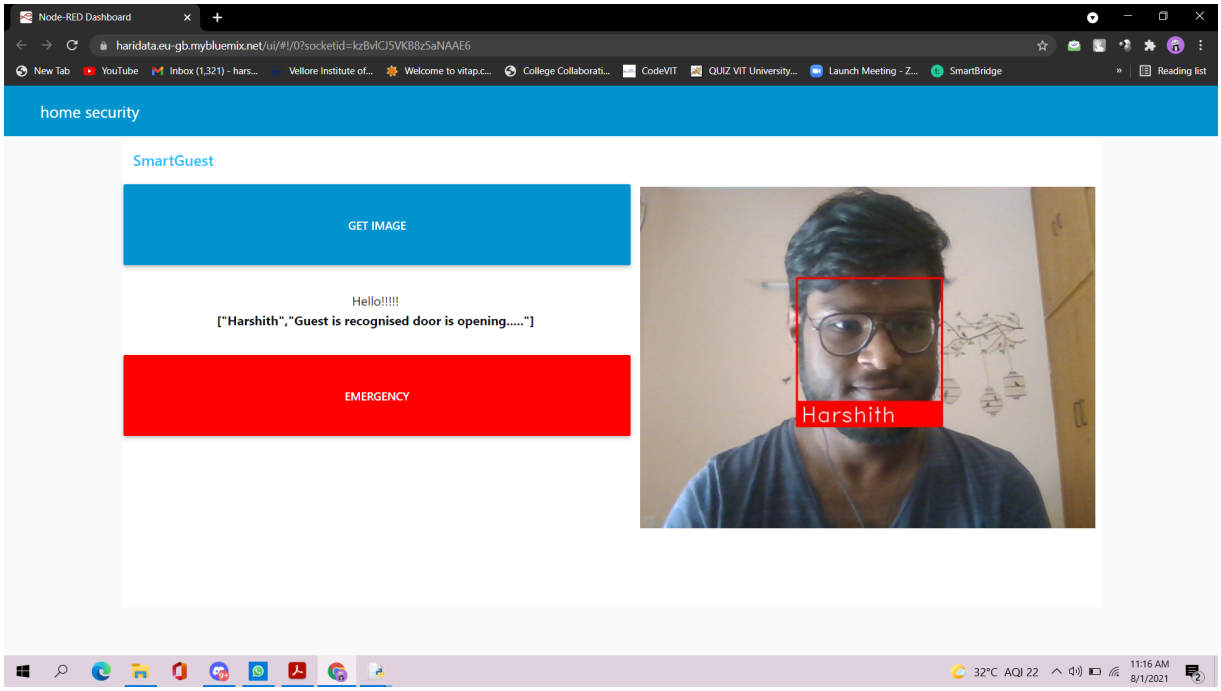

Flow chart
Accessing Primary Camera
Capturing Image
Face Detection and Recognistion
if recognise = true
Door will be open automatically
recognise = false
else
then sends unkown person is detected
if open door = true
else if emergency = true
adminstrator given access to open door
emergenccy msg will be sent
if no entry = false
else
no access to open door
Then image will be upload to cloud object storage and cloudantDB
Nodered
Web ui
Mobile Application

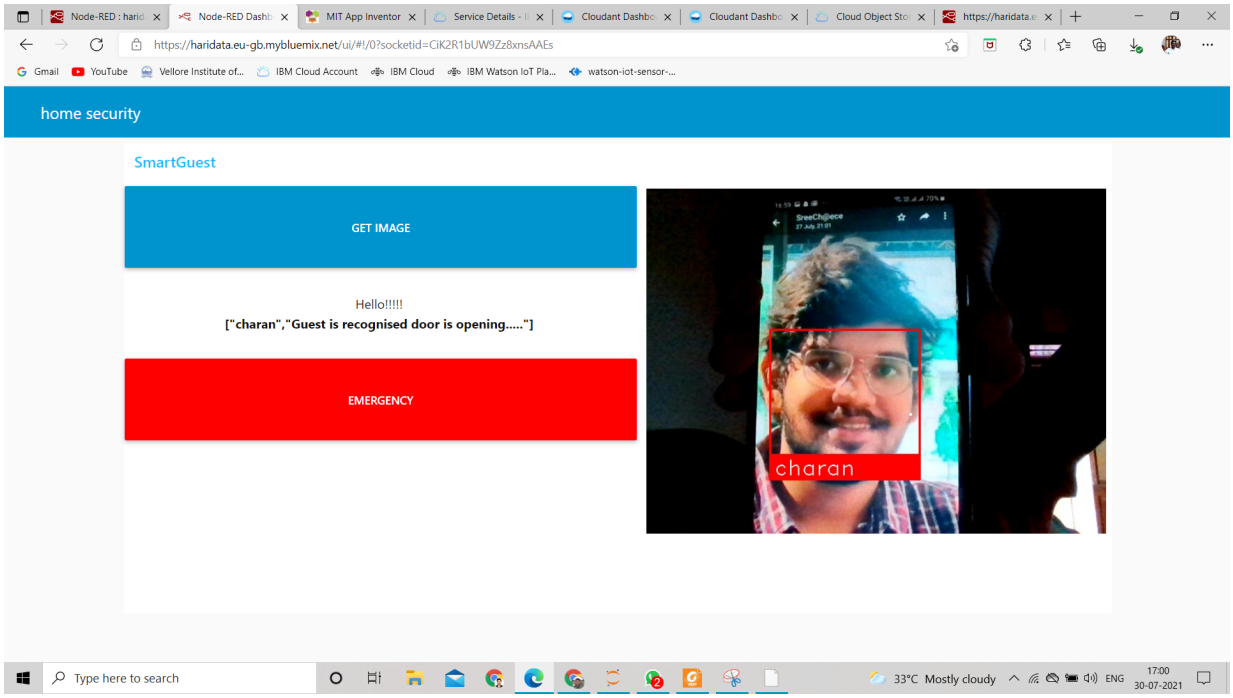
6. Result

For guests:

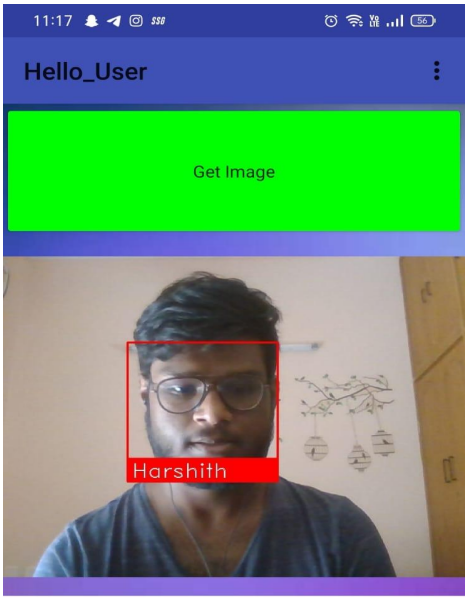


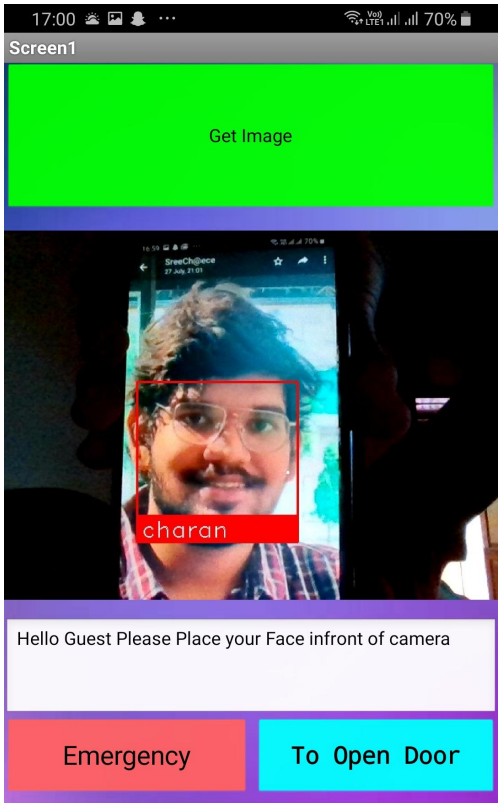
In Web UI:





In Mobile Application:





7. Advantages & Disadvantages

Advantages:

- We can store data in IBM cloud object storage.
- We can remotely operate the door (unlocking)
- We can monitor the guests from anywhere.
- The emergency button gives more advantage when the administrator is not using the mobile application.
- Using Node red flow we can easily build a web and mobile application

Disadvantages:

- The major disadvantage is the door can be unlocked, by showing a guest picture.
- Sometimes it may recognise an unknown person as a guest.
- CloudantDB must be always active to get recent images.
- The Unlocking depends on the administrator, if he/she is not using a mobile or mobile application then an unknown person cannot get access to enter.

8. Applications

With the rapid advancement of the IoT market, companies tend to focus on the time-to-market and wait for innovative products as fast as possible.
With the increased availability of consumer sensing devices including modern smartphones, the domain of networked sensing has evolved into the Internet of Things, As IoT has gained wide acceptance the need for sound software engineering approaches to adequately manage the development of complex applications arises.

- This framework could be used for an automated burglar alarm system, guest attendance monitoring, and light switches, all of which are easily integrated with any smart city base.
- In this way, IoT solutions can allow real-time monitoring and connection with central systems for automated burglar alarms.

- The monitoring framework is developed on the strength of the web application to obtain real-time display, storage, and warning functions for local or remote monitoring control

9. Conclusion

The camera placed on the door informs the homeowner as soon as the door is opened by sending a Push notification. The user will get this notification irrespective of whether the phone is locked or unlocked or even if any other app is opened at the moment. This was the main objective of the project, which is that the user feels safe and does not worry about any intrusion or break-ins when he is away from home. This setup can also be used in commercial offices where some areas are restricted for certain personnel, such a system will immediately inform the administrator of any unauthorized personnel trying to access such an area. Therefore the extensibility and applicability of such a system is limited only by the imagination.

10. Future Scope

The project detects faces and recognizes the person in an efficient way; it can be made more efficient in recognizing the person for future automation.It could be made to detect live human movement and recognize the person.The project can be further made to speech processing to control the application. Smart homes are an element of developing smart cities. In recent years, countries around the world have spared no effort in promoting smart cities. Smart homes are an interesting technological advancement that can make people's lives much more convenient. The development of smart homes involves multiple technological aspects, which include big data, mobile networks, cloud computing, Internet of Things, and even artificial intelligence.

The developed system can also be used in industrial and commercial applications such as offices, warehouses and other areas where some areas are reserved for authorized personnel only or other places where safety and precautions are of primary concerns such as the internet server room of a big MNC from where corporate data can be stolen. The system can also be easily upgraded to add extra

safety features such as cameras, motion detection sensors, etc. for increased safety. The system can also further be developed by adding an RFID scanner so that the authorized users need only carry a RFID or NFC tag with them on their person. The RFID scanner will work by scanning the tag wirelessly and if the user is authorized to enter, the alarm system will be disabled for some time so that the user can enter.

11. Bibliography

https://cloud.ibm.com/apidocs/cloudant?code=python#getserverinformation

https://cloud.ibm.com/docs/cloud-object-storage?topic=cloud-object-storage-python

https://drive.google.com/drive/u/0/folders/14ZNzzzkbhkYPuvPQoU9A3hz_ffCOfWot

https://appinventor.mit.edu/

https://realpython.com/face-recognition-with-python/

https://haridata.eu-gb.mybluemix.net/ui/#!/0?socketid=kzBvlCJ5VKB8z5aNAAE6

12. Appendix

a. Source code

```
import cv2
import face_recognition
import datetime
import ibm_boto3
from ibm_botocore.client import Config, ClientError
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator
import wiotp.sdk.device
import time
import random
```

```
myConfig = {
    "identity": {
        "orgId": "aannkh",
        "typeId": "VITharish",
        "deviceId":"12345"
    },
    "auth": {
        "token": "9381628451"
    }
}

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

# 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 = "XIzdVg_jB4YRIuitd-fz1LkP2OTbnFHpqE2v4NrBVg-F" # eg
"W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_INSTANCE_CRN =
"crn:v1:bluemix:public:cloud-object-storage:global:a/27c2eb7ce3824e58b9022d
8e4ac34792:e0afac5f-48ad-492d-9f2e-4dd702111eaa::" # eg
"crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003xxxxxxxxxx1c3
e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"

# Create resource
cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

authenticator =
BasicAuthenticator('apikey-v2-h0ooew2msl6r4mla6of1nk1m75bz8cs16pbx4fch7dw',
'0d52d202827e65107632ba94b4c7adc4')

service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-h0ooew2msl6r4mla6of1nk1m75bz8cs1
6pbx4fch7dw:0d52d202827e65107632ba94b4c7adc4@7f536afa-8bcf-495e-a4c8-1cf0e1
5dfdfe-bluemix.cloudantnosqldb.appdomain.cloud')
```

```
bucket = "harishdata"
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))
```

```
mydoor = {'Door':'Hello Guest Please Place your Face infront of cam'}
client.publishEvent(eventId="status", msgFormat="json", data=mydoor, qos=0,
onPublish=None)

video_capture = cv2.VideoCapture(0)

hari_image = face_recognition.load_image_file(r"photo.jpg")
hari_face_encoding = face_recognition.face_encodings(hari_image)[0]
```

```
harshith_image1 = face_recognition.load_image_file("harshith.jpeg")
harshith_face_encoding =
face_recognition.face_encodings(harshith_image1)[0]

sreech_image = face_recognition.load_image_file(r"sreech.jpg")
sreech_face_encoding = face_recognition.face_encodings(sreech_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings =
[hari_face_encoding,harshith_face_encoding,sreech_face_encoding]

known_face_names = [
    "Harish",
    "Harshith",
    "charan",
]

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition
processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color
(which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of
video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
```

```
face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
            name = "Unknown"

            # If a match was found in known_face_encodings, just use the
first one.
            if True in matches:
                first_match_index = matches.index(True)
                name = known_face_names[first_match_index]

            face_names.append(name)

    process_this_frame = not process_this_frame


    # Display the results
    for (top, right, bottom, left), name in zip(face_locations,
face_names):
        # Scale back up face locations since the frame we detected in was
scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0,
255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
255, 255), 1)

        # Display the resulting image
        #mydoor = {'Door':'Hello Guest Please Place your Face infront of
cam'}
```

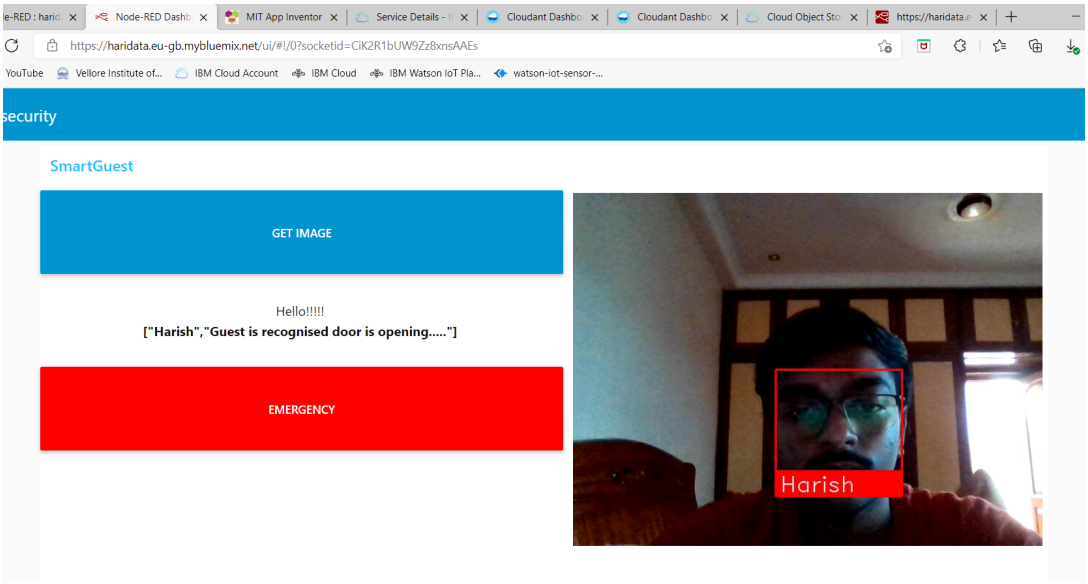
```
        cv2.imshow('Video', frame)
        picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
        cv2.imwrite(picname+".jpg",frame)
        '''multi_part_upload(bucket, picname+'.jpg', picname+'.jpg')
        json_document={"link":COS_ENDPOINT+'/'+bucket+'/'+picname+'.jpg'}
        response = service.post_document(db='sample1',
document=json_document).get_result()'''
        def myCommandCallback(cmd):
            print("Message received from IBM IoT Platform: %s" %
cmd.data['command'])
            m=cmd.data['command']
            if (m =="true"):
                if name in known_face_names:
                    mydata = {'Guest':True}
                    mydoor = {'Door':[name,'Guest is recognised door is
opening.....']}
                else:
                    mydata = {'Guest':False}
                    mydoor = {'Door':'Guest is not recognised if you want
to open please press open button to open door'}
        #client.publishEvent(eventId="status", msgFormat="json",
data=mydoor1, qos=0, onPublish=None)
                client.publishEvent(eventId="status", msgFormat="json",
data=mydata, qos=0, onPublish=None)
                client.publishEvent(eventId="status", msgFormat="json",
data=mydoor, qos=0, onPublish=None)
                print("Published data Successfully: %s", mydata)
                time.sleep(2)
            else:
                mydoor = {'Door':'Hello Guest Please Place your Face
infront of camera'}
                client.publishEvent(eventId="status", msgFormat="json",
data=mydoor, qos=0, onPublish=None)
        client.commandCallback = myCommandCallback
        multi_part_upload(bucket, picname+'.jpg', picname+'.jpg')
        json_document={"link":COS_ENDPOINT+'/'+bucket+'/'+picname+'.jpg'}
        response = service.post_document(db='sample1',
document=json_document).get_result()


    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
# Release handle to the webcam
mydoor = {'Door':'Hello Guest Please Place your Face infront of camera and
press get image'}
client.publishEvent(eventId="status", msgFormat="json", data=mydoor, qos=0,
onPublish=None)
video_capture.release()
cv2.destroyAllWindows()
client.disconnect()
```

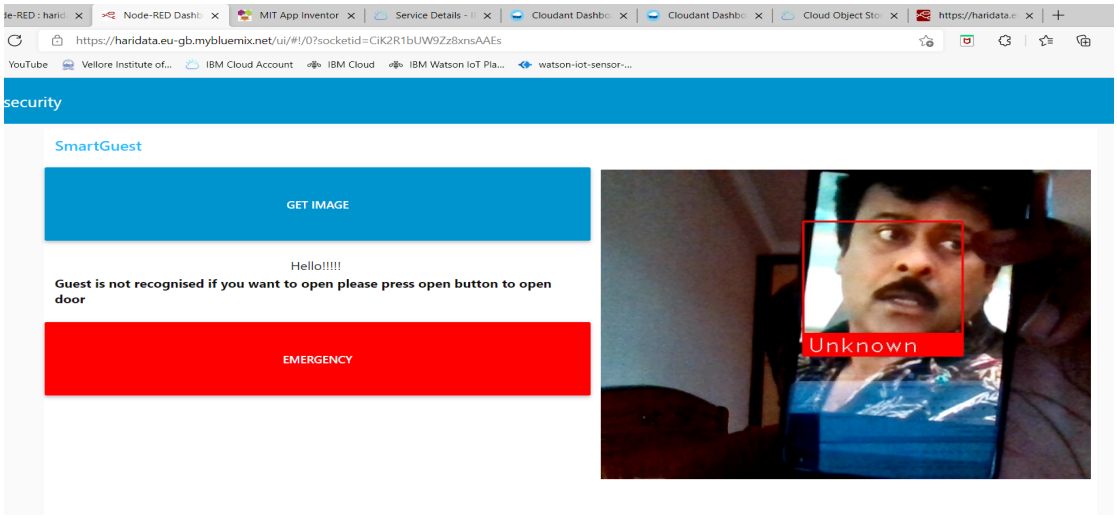
This Python code is Basically about when a person is in front of the camera it will capture the image and recognize the face and show if the person was known with his/her name and the door will open automatically. Otherwise it shows the person was unknown and it doesn't give access to open the door.
In This code we are using sdks to send the captured images to respective cloud object storage and cloudant database.
Using IBM IoT to send the data which we get output from code to node red flow.

b. UI output Screenshot

For Guest:



For unknown:



Link for web UI:

https://haridata.eu-gb.mybluemix.net/ui/#!/0?socketid=kzBvlCJ5VKB8z5aNAAE6