# Automatic garage door opener

Final Project

Team –

ROHAN CHANDRASHEKAR (20BCE0765)

SANSKAR VIDYARTHI (18BEC0013)

JAVVAJI JASWANTH SAI (18BEC0511)

RISHI RANJAN (19BEC0798)

## 1. Introduction

Purpose – To construct an IoT based automatic garage door opening system.

Overview –

- Automatic opening of the garage doors by detecting the authorized vehicles.
- The garage door will be integrated with the camera which will detect the car it will automatically open the garage doors to park the car.
- the lights will be switched on if the garage door is opened and it will be switched off if the door is closed.
- He can also control the garage doors and lights using the mobile app.
- All the captured images of the vehicles will be stored.

## 2. Literature survey

Existing problem – Every time there's a vehicle to be parked or taken out of the garage, someone or the driver himself has to open and close the garage door which is very inefficient and could be a time-consuming process especially in public garages, parking lots and basement parking. During night time, lighting could also be an issue.

Proposed solution – The solution is to implement an automatic garage door opener wherever required with both automatic and manual controls of opening and closing of the garage along with light controls. Here, we would use an ultrasonic sensor (python code alternate) to detect the proximity of the vehicle and open the door and Node-Red-Web-UI or mobile application to get the most recent image of the vehicle being parked which is beneficial for security purposes too. The app could also be used to control the lighting. The data of the vehicle parked is also recorded over the cloudant database.

## 3. Theoretical Analysis

**Block Diagram**

**Software Designing**

We have used clarifai to check if the object is a vehicle. Python code is used to create random integers between 5 and 500 which is used as sensor data to measure distance of vehicle from the garage door. The threshold is set as 15 so when the value generated is less than 15, the garage door opens, image is captured and stored into the cloud and the lights switch on. We've used Node Red for the Web UI and IBM cloud to store the data.

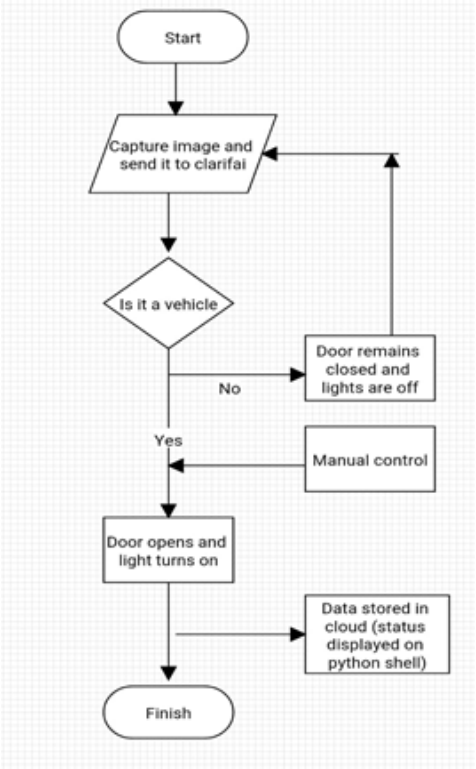
## 4. Experimental Investigations

In the python program, instead of the sensor, we get random distance values from 5 to 500 meters. If the distance is less than 15 meters then we take a picture and send it to clarifai to check if it is a vehicle. If it is a vehicle then we open the door and switch on the lights and send the image to cloud and image details(including URL) to cloudant database.

From the cloudant, we take the URL of the latest image and display it in the node-red web UI. In the web UI, the user has buttons to control the door and light. Whatever they choose, it is displayed in python shell.

In mobile app made using MIT App Inventor, again we take the URL from the database and display the latest image. There again, there are buttons to control door and light. The user selection is sent to node-red from where it is displayed in python shell.

## 5. Flowchart




## 6. Result

We receive the most recent image of the vehicle being parked as we press the get image button. The message of garage door being opened and lights being turned on and off are received on the python shell when their respective event occurs.

## 7. Advantages and Applications

### 1. Convenience

One of the main benefits of an automatic garage door opener is convenience. You can stay in your car until you're in (or out of) your garage, so you don't have to get out of your car to open or close your garage.

You won't need to leave your children or dog in the car, or go outside in bad weather just to open or close your garage door.

### 2. Lighting

Modern automatic garage door openers are available with built in lights to will illuminate your garage or the path to your garage. This type of lighting can be invaluable at night, in the winter or in bad weather.

**3. Security**

Garage security is of paramount importance, and so you'll want to make sure that your automatic garage door opener helps to keep your family, home, and possessions safe. That's when the image capture feature comes in handy.

**4. Safety**

You won't need to worry about the garage door crashing down on something or somebody in the way, or fingers getting caught anywhere. For those with a medical condition such as arthritis, or young children, one of these could be essential.

**5. Automatic closing**

Automatic closing means that you won't need to get out of your car to close the door, or remember to press the button on your remote; your garage door opener can close itself too. This is ideal if you have children, often forget to close your garage door, or get to work and worry that you may not have closed your garage.

**6. Low energy use**

Although automatic garage door openers can be highly advanced and technical, they do not need a lot of power to work. This makes them highly cost effective to run, and less damaging to the environment.

**7. Technology**

Modern garage door openers are available with various features and facilities to provide you with more convenience. Some other other benefits of an automatic garage door opener include the ability to be controlled by smart phone apps. These enable you to ensure your door is closed, wherever you are, or to sound an alarm to let you know your door is open.

**8. Maintenance**

One the benefits of an automatic garage door opener is that it doesn't require much maintenance, but you'll want to keep yours in accordance to the manufacturer's

recommendations, so that it always works when you need it to.

**9. Cost**

Despite their convenience and features, automatic garage doors are relatively inexpensive to buy and fit. When you think about what they offer, and what they mean to a family with a couple of cars, kids who keep their bikes in the garage and who are concerned about security, they are an absolute bargain.

**10. Peace of mind**

Whether you simply want to know that your garage door is closed, have a medical condition and can't open your door yourself, or simply want to save a few minutes every day, an automatic garage door open makes a lot of sense to many people. What seems like an unnecessary luxury becomes invaluable if you use your garage a lot.

## 8. Disadvantages

There are a few downsides in the use of garage doors, including:

- The installation process requires the help of a professional garage door technician in order for it to be accurate and efficient.
- Automatic garage doors are more costly than manual garage doors.
- Automatic garage doors require regular maintenance and checkup which costs money. This means that the expenses do not end upon installation.
- They are also more complex to clean since you also need to polish internal parts like springs and bolts to avoid the accumulation of rust.
- Automatic garage door repairs can be costly especially when some parts need to be replaced.

## 9. Conclusions

In this project we use an ultrasonic sensor (random values in python for sensor data as

physical hardware is not available) to measure the distance of the vehicle from the garage and automatically open the garage when the vehicle crosses a set proximity limit. The garage door can also be manually opened using a mobile application or Node Red web UI. The information and the image of the vehicle is stored over the cloudant database and the image can be see in the web UI as well.

## 10. Future Scope

In the future an alarm system could be attached to the project which will ring if an unregistered vehicle or person is detected in the proximity range of the garage door and put the garage on lock down and instantly inform the owner about the breach.

## 11. Bibliography

1. Smart Internz internship reference materials

2. Softwares – MIT App Inventor, IBM Watson, Clarifai, Python IDLE, Node-Red

3. AA Garage doors, "*10 Benefits of an Automatic Garage Door Opener*", 8th June 2016

## 12. APPENDIX

## i) Python Code-

```
#Vehicle Detection

import cv2


#IBM IOT connectioon

import wiotp.sdk.device


#IBM COS

import ibm_boto3


#IBM Cloudant
```

```
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

#Clarifai
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
stub = service_pb2_grpc.V2Stub(ClarifaiChannel.get_grpc_channel())
from clarifai_grpc.grpc.api import service_pb2, resources_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2

#Miscellaneous
import time
import random
import datetime

#IOT device connection
myConfig = {
    "identity": {
        "orgId": "j8rgpm",
        "typeId": "First_Device",
        "deviceId":"123"
    },
    "auth": {
        "token": "First_Device_123"
    }
}
```

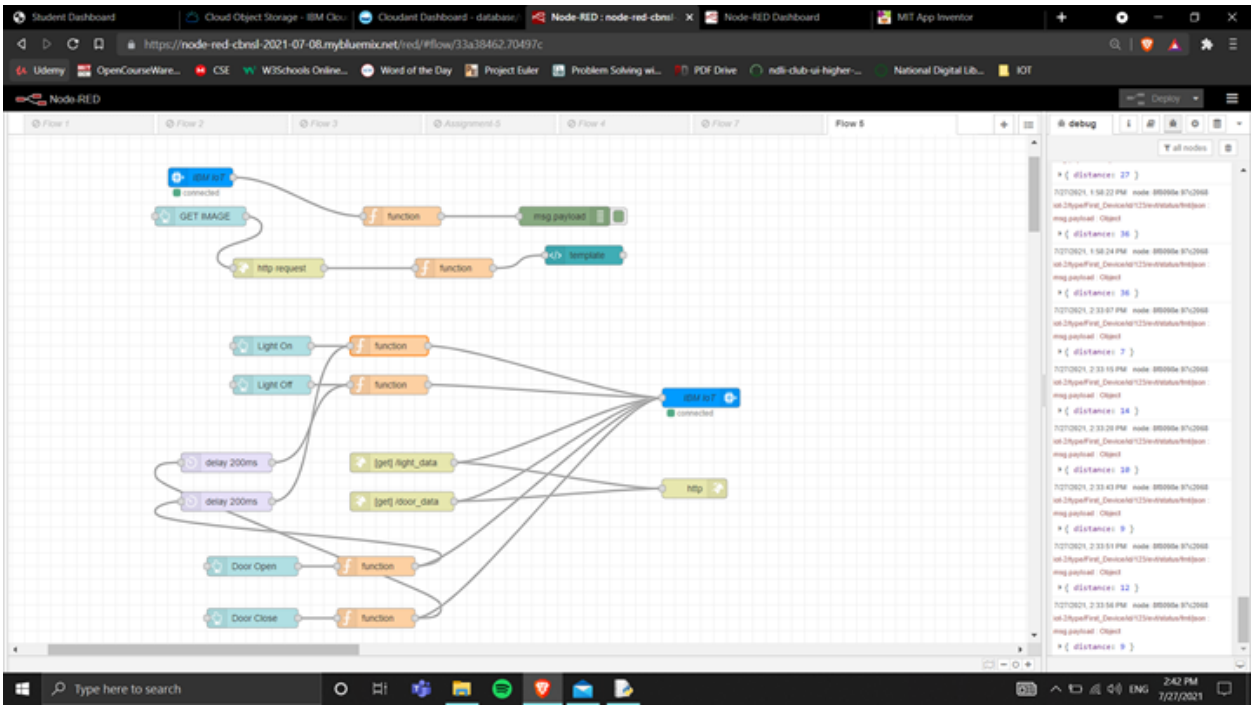
```
# Constants for IBM COS values

COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS_API_KEY_ID = "78NLlHvfEUxheNWr4jehqUGifUPewWVaPrk3N2HXaMKy"

COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/abaf1723c9c84a148e04f11a73d2442a:880edc7f-ce14-4df9-8545-5c592c737b87::"


# 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-v2-10xypw9nevga82oqb993uwcfdqgl748fcoiznj3jfrzn', '3fb54d29a2d64e40f7a0657467d33a27')

service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-10xypw9nevga82oqb993uwcfdqgl748fcoiznj3jfrzn:3fb54d29a2d64e40f7a0657467d33a27@c6a728a6-a633-4d59-a8e9-c06a612f8176-bluemix.cloudantnosqldb.appdomain.cloud')

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

client.connect()


#Image upload to COS bucket

bucket = "rohanbucket"

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))
```

```
        part_size = 1024 * 1024 * 5
        file_threshold = 1024 * 1024 * 15
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
        )
        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))



video = cv2.VideoCapture('cars.avi')
# Clarifai Authentication
metadata = (('authorization', 'Key f9c413355966419bbd1b0e0125a9a5bc'),)

#Initial Conditions
print("Door is Closed")
print("Light is Off")

def myCommandCallback(cmd):
    m=cmd.data['command']
```

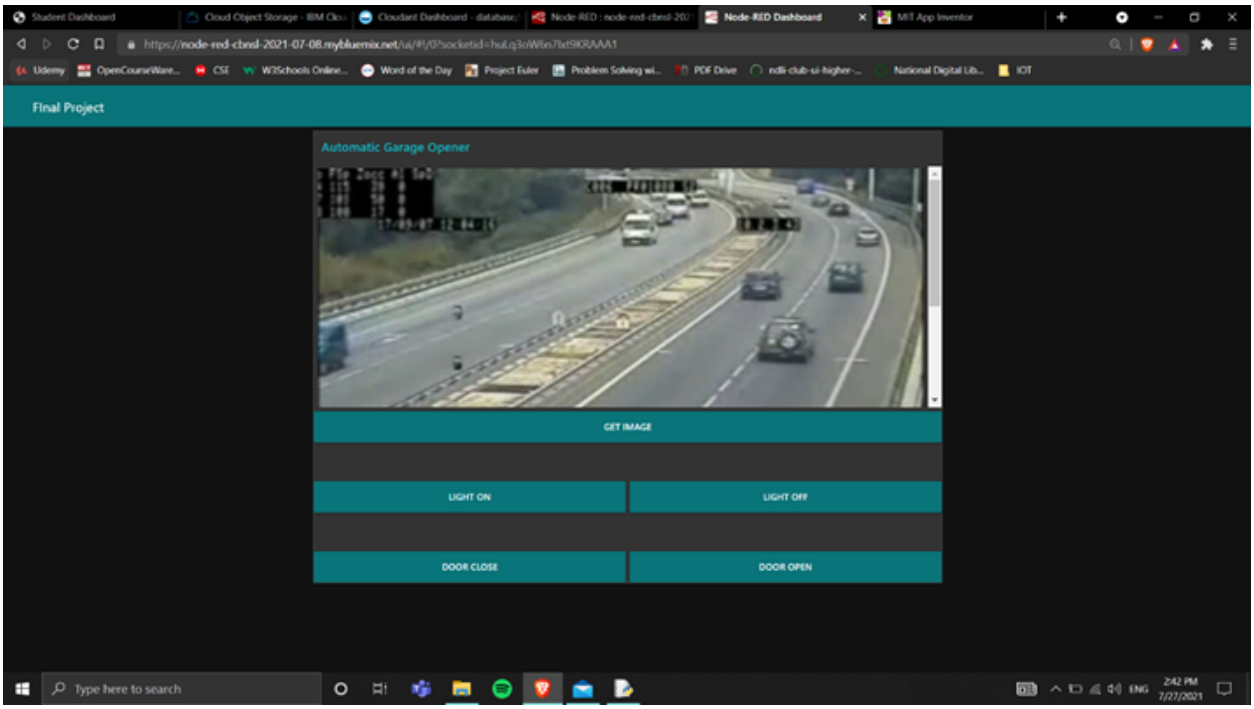
```
    print(m,"\n")



while True:
    distance=random.randint(5,500) # random distance values from 5 to 500 meters
    check,frame=video.read()
    frame = cv2.resize(frame, (600,400))
    if (distance<=15): #picture is sent to clarifai when distance is less than 15 meters
        myData={'distance':distance}
        print(myData)
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
        picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M-%S")
        path="D:\\rohan\\Personal\\Courses\\IOT\\Final Project\\pics\\" +picname + ".jpg"
        cv2.imwrite(picname+".jpg",frame) # Images are stored locally in a folder
        time.sleep(1)
        with open(path, "rb") as f:
            file_bytes = f.read()

        request = service_pb2.PostModelOutputsRequest(
            # This is the model ID of a publicly available General model. You may use any other public or custom model ID.
            model_id='aaa03c23b3724a16a56b629203edc62c',
            inputs=[
              resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=file_bytes)))
            ])
        response = stub.PostModelOutputs(request, metadata=metadata)

        if response.status.code != status_code_pb2.SUCCESS:
            raise Exception("Request failed, status code: " + str(response.status.code))
        a= []
```

```
        for concept in response.outputs[0].data.concepts:
            if(concept.value > 0.8):
                a.append(concept.name)
        t=1
        for i in a:
            if(i == "car" or i == "vehicle" or i=="bike"):
                print("Vehicle is detected\n")
                print("Door is open\nLight is On")
                #if vehicle is detected document is uploaded to cloudant database
                multi_part_upload(bucket, picname+'.jpg', picname+'.jpg')
                json_document={"link":COS_ENDPOINT+'/'+bucket+'/'+picname+'.jpg'}
                response = service.post_document(db='sample', document=json_document).get_result()
                break

    Key=cv2.waitKey(1)
    client.commandCallback = myCommandCallback

client.disconnect()
```

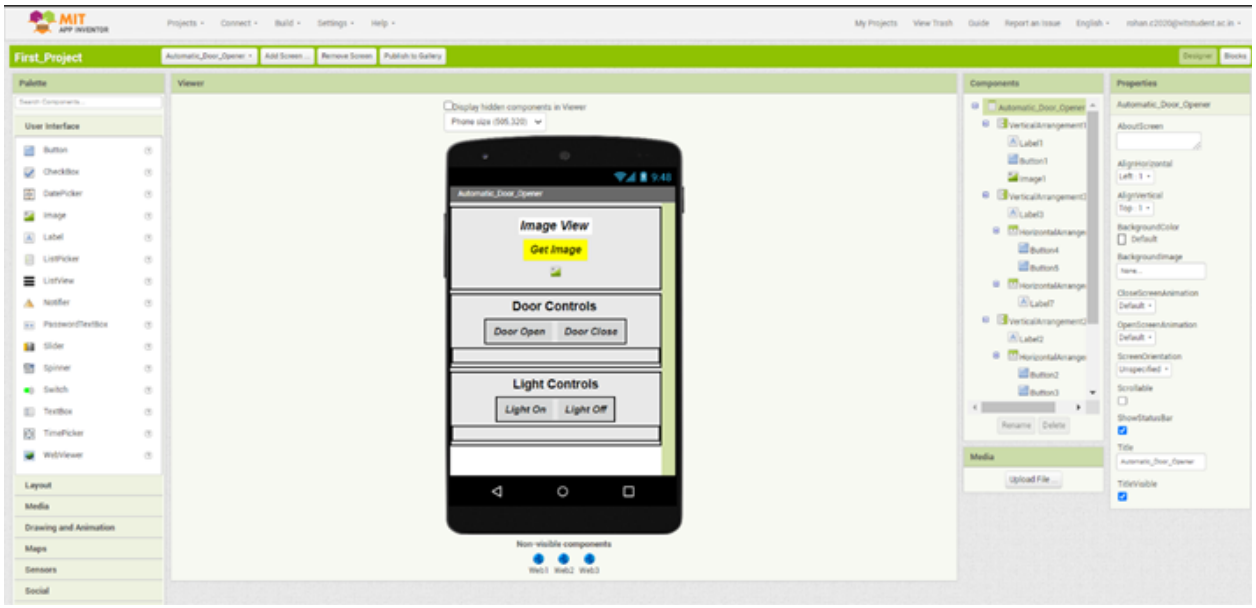
# ii) Node Red-


Node-RED
IBM IoT
GET IMAGE
function
msg.payload
http request
function
template
Light On
Light Off
function
function
IBM IoT
delay 200ms
delay 200ms
[get] /light_data
[get] /door_data
http
Door Open
Door Close
function
function
debug
Type here to search

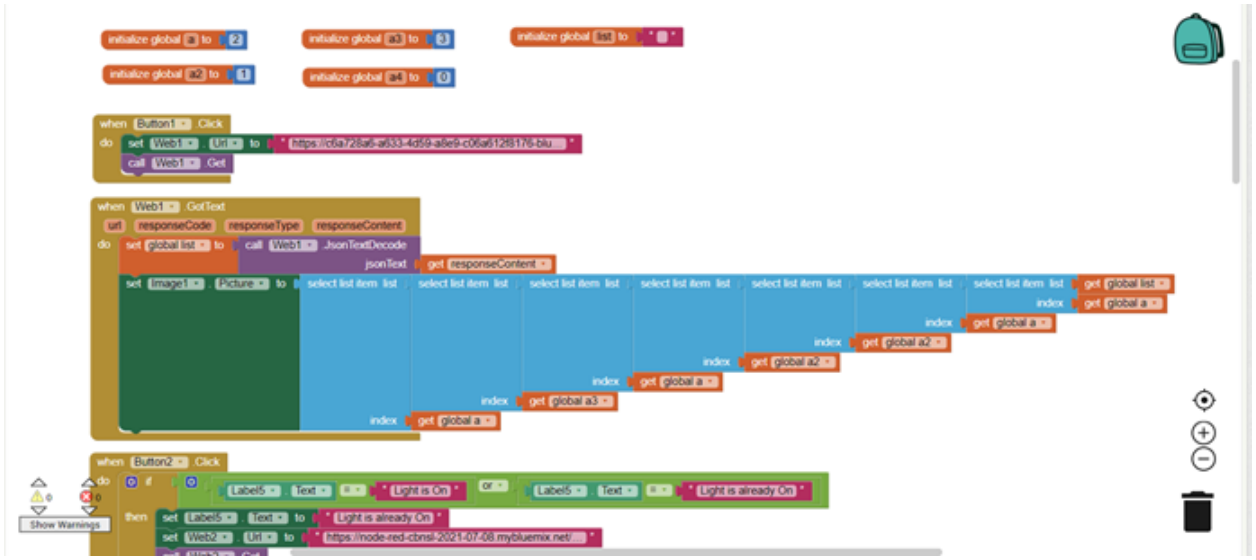
# WEB UI-

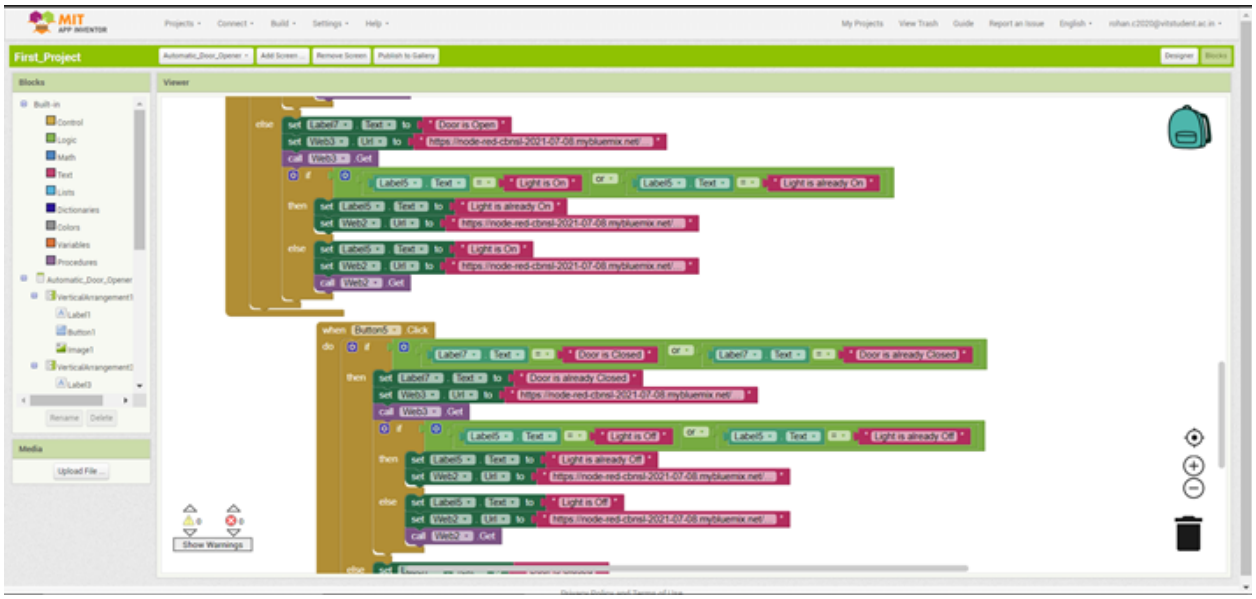

Node-RED Dashboard
Final Project
Automatic Garage Opener
GET IMAGE
LIGHT ON
LIGHT OFF
DOOR CLOSE
DOOR OPEN
Type here to search

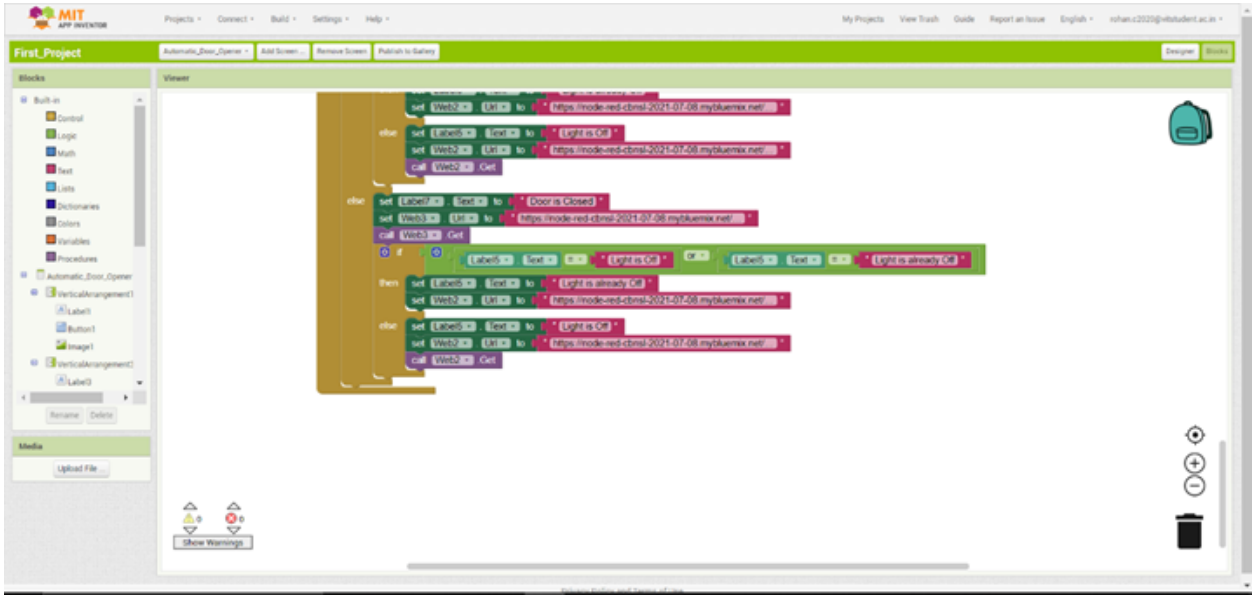
# iii) MOBILE APP-



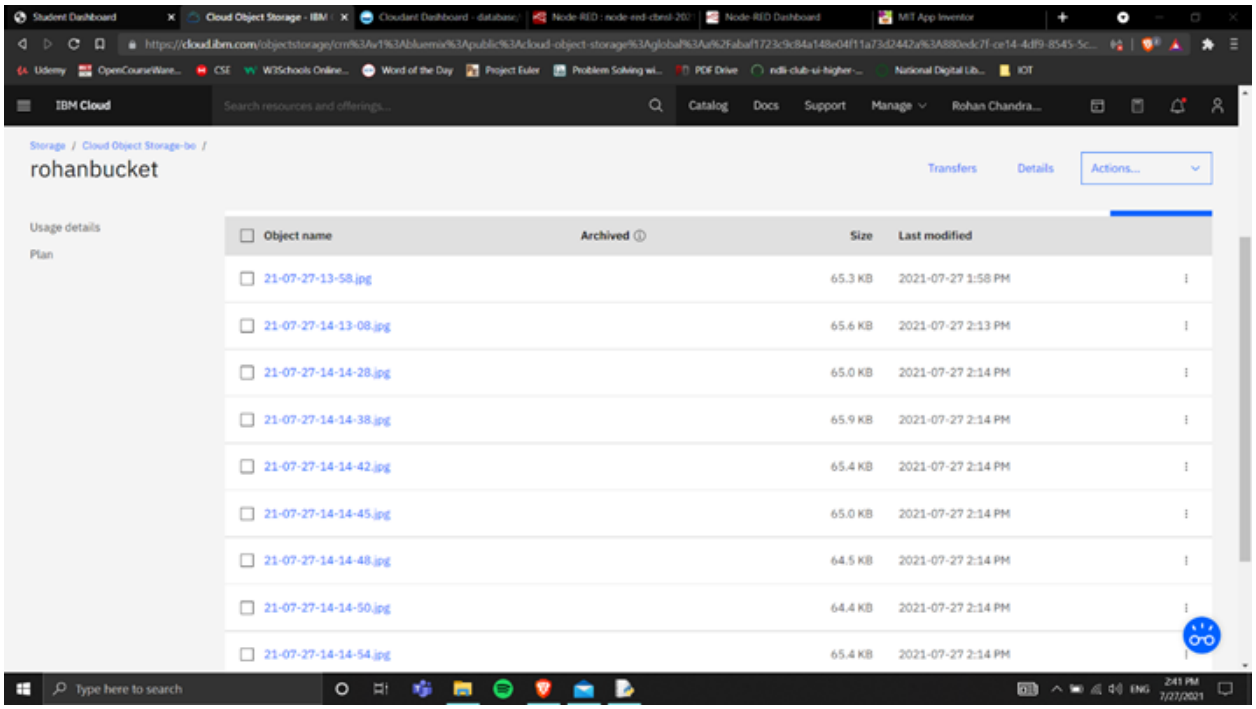

## Backend for the App -

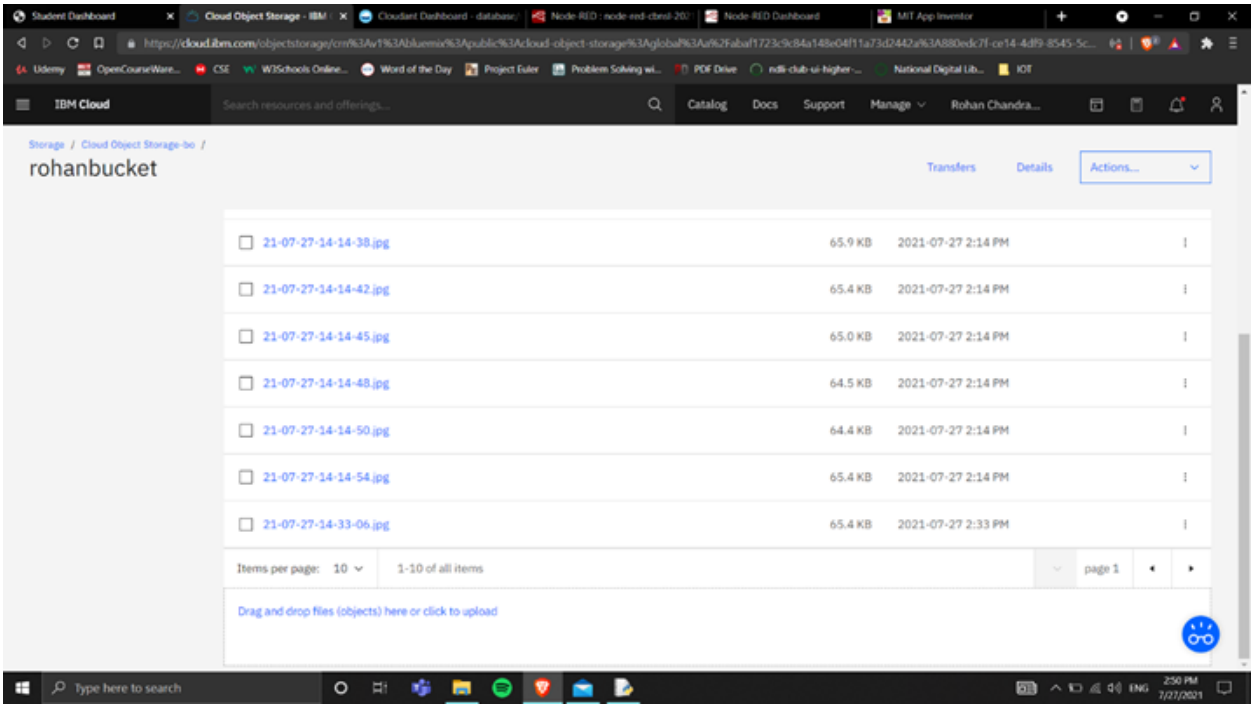




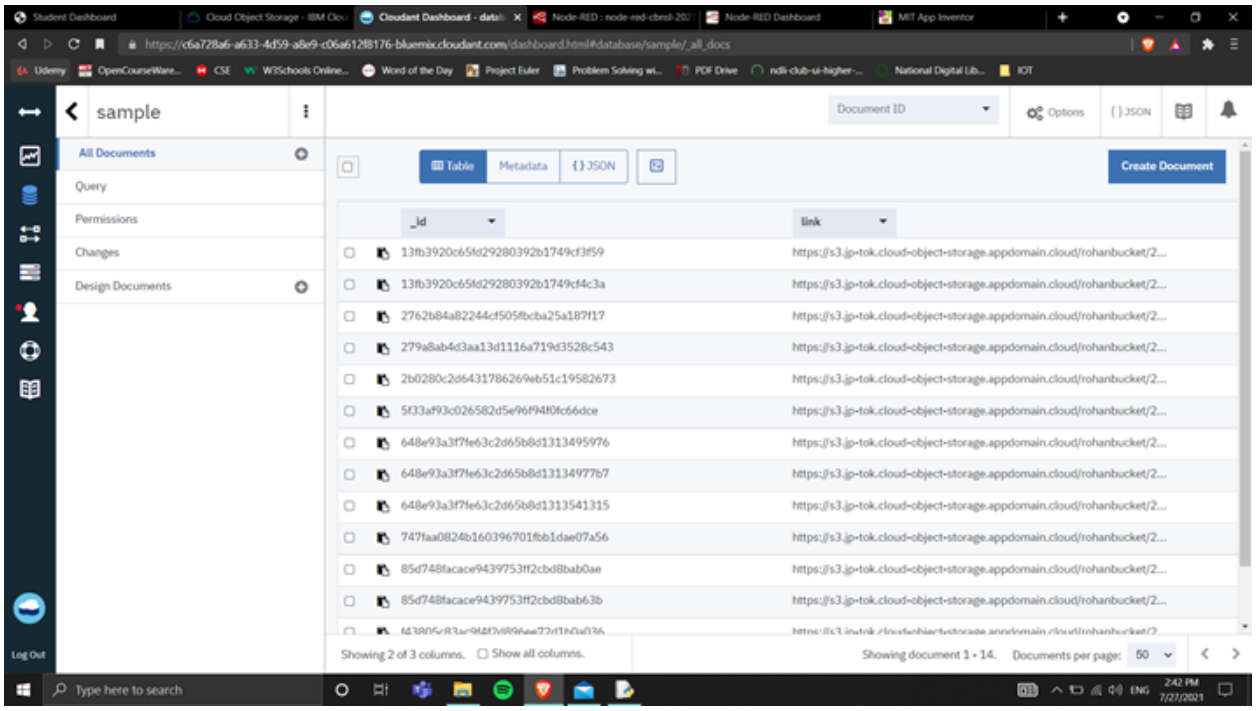



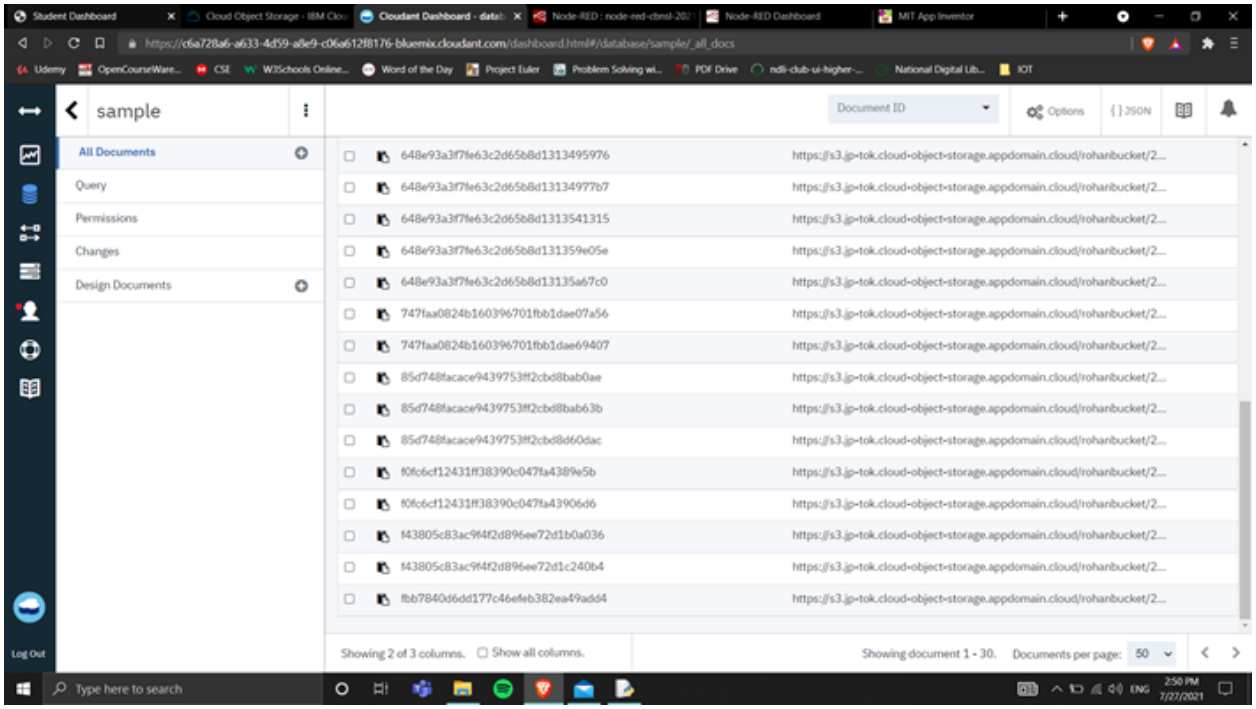

## iv) Cloud Object Storage-




## Cloudant Database-




## v) Python Shell Output-



## Link for working video of the project-

https://drive.google.com/file/d/1zNX_MmuEzVsTT6IHcvUY1QoT4O_g_y_2/view?usp=sharing