***INTERNET OF THINGS April 20***

***Project on***

***Garage Door Opener using IOT(Opencv)***

***by TEAM No:24(MIND BENDERS)***

1)Tejaswi Atmuri

2) Vukoti Venkatesh

3)Shaik Baji

4)Suma Penumaka

***1.Introduction:***

**1.1-Overview:**

Garage doors can be frustrating things, being a chore to open manually and all. Many people opt to install a motorized opener, but for some, even this isn’t enough. Hooking up a garage door to the Internet of Things we made it easy.so, it reduces the manual work and increases the safety.By using the cloud platforms like IBM cloud,we can easily manipulate changes in code that we dump into the device,interface the device and make changes in the device right from the cloud plaform itself. The project “Garage Door Opener” using IBM cloud and it's services is an exclusive project which is used to open the garage door automatically upon recognising the authorised person and closing the door using mobile phone.

**1.2-Purpose:**

Automatic doors effectively saves the energy and reduces the manual work.The Door Opener using opencv is not only useful to open the garage door,it can also be used to allow only authorised persons through a door.It can be used in places where the access is restricted to the authorised persons only.

**2.Literature Survey:**

**2.1-Existing Problem:**

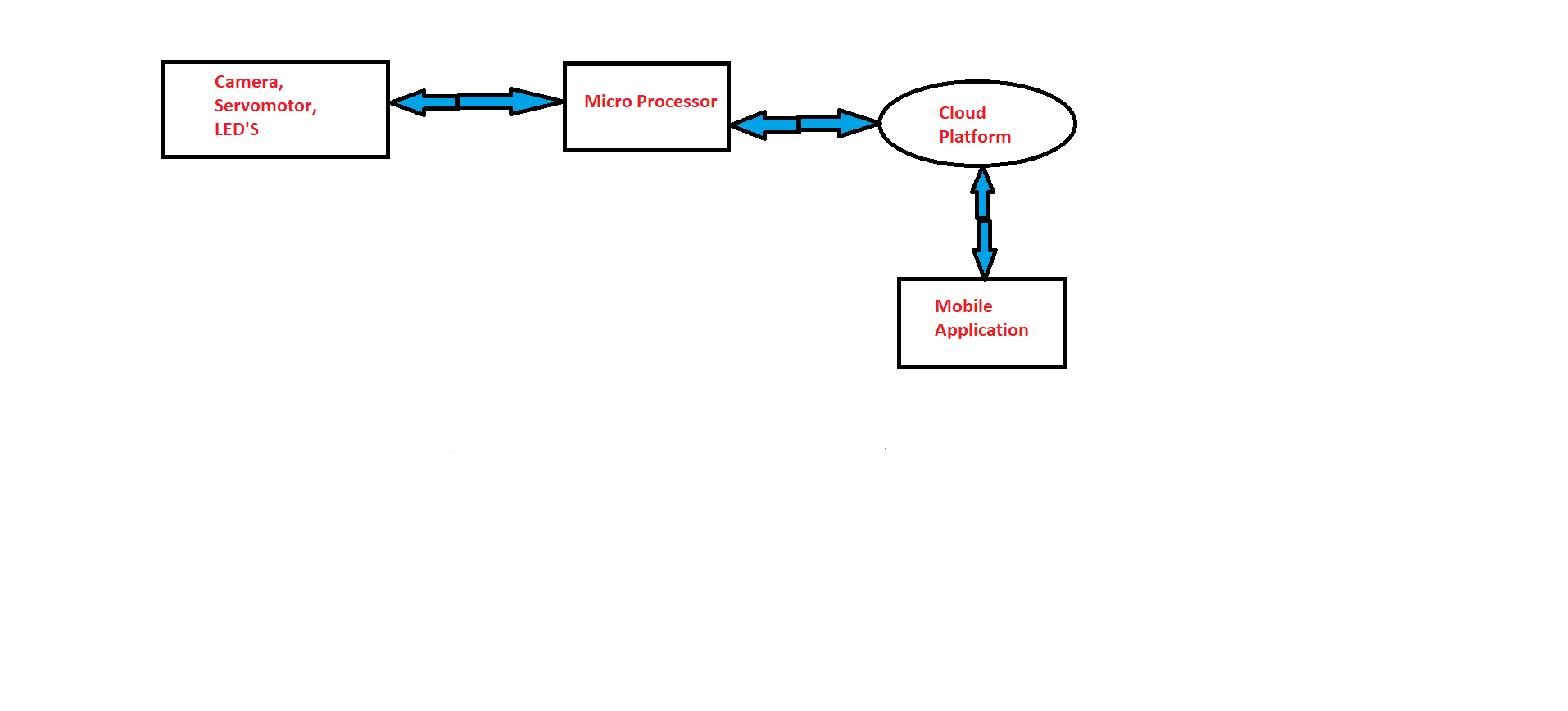
Now-a-days, security issues play a key role in every aspect.Manual work may include forgetting to close the door,which might grant access to thiefs.In such a case,the security may be at risk.Manually opening the garage door is atask where one needs to get out of the car and then open it or close it.We need to find a solution where we can avoid the manual work and also provide security.

**2.2-Proposed Solution:**

To avoid all the above mentioned problems,we developed an automated garage door unlocking system,which unlocks the garage door upon recognising the face of the person.For this we train tthe visual recognition service with the images of the authorised person,so that when that person is recognised,the command is send to the device to open the door.If anyother person is recognised other than he authorised person,the command is sent as not to open the door and the image of the person is sent to the authorised persons for security reasons.

**3.Theoritical Analysis:**

**3.1-Block Diagram:**

 **Fig: Block Diagram of Garage Auto Door System.**

***3.2-Software Design:***

We developed this application using some softwares like:

**1.IBM Cloud.**

**(i)Visual recognition.**

**(ii)Node-Red**

**(iii)Cloud Object Storage**

**(iv)Internet Of Things Platform**

**(v)Cloudant dB**

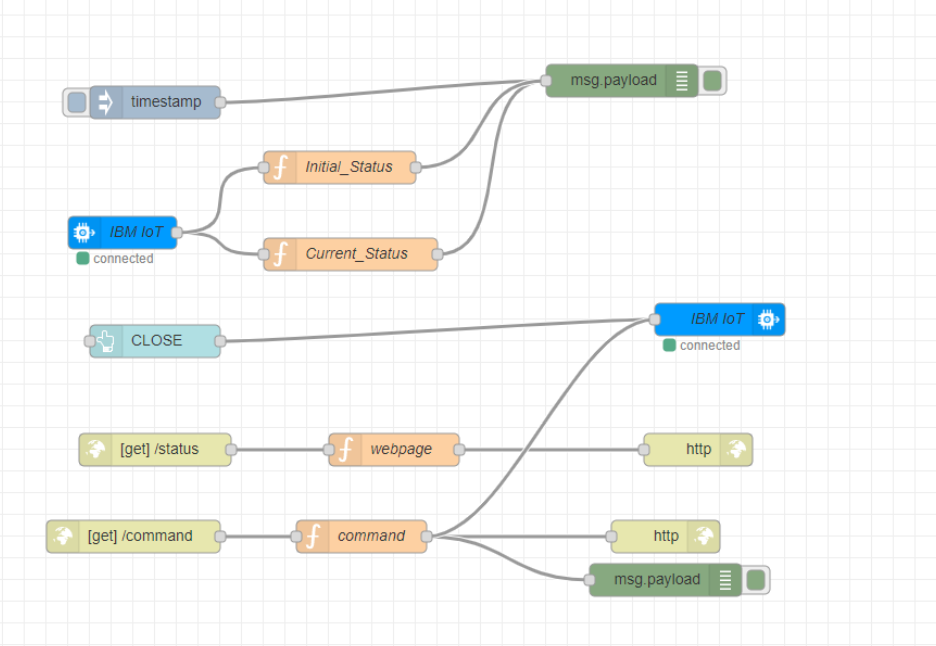
**2.MIT APP Inventor.**

**3.Open CV.**

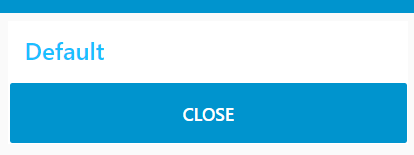
**4.Python Software.**

**4.Experimental Investigations:**

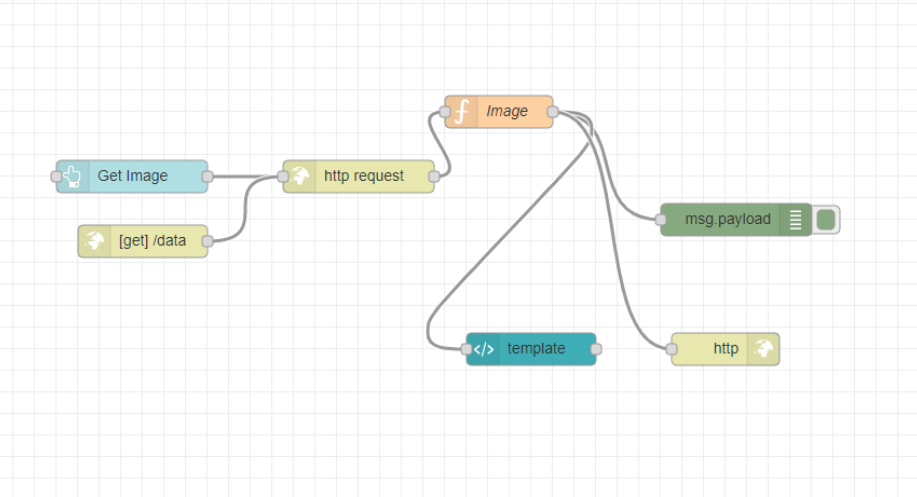
* Automatic opening of the garage doors by detecting the authorized vehicles.
* We can also train the authorized persons faces,based on the face recognition the garage doors will be opened whenever the person wants to take the vehicle out.
* Everytime recognising the person or the number plate,images will be stored in the cloud along with the date and time.
* Through the mobile app admin can add the number plate details to the device.
* When ever the unauthorised person tries to cross the door it automatically sends the data to the admin through cloud and so, that they are alerted.



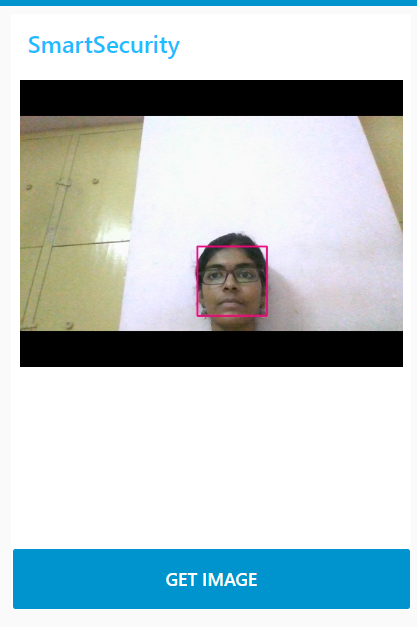
***Fig:Node red flow 1***



***Fig:The node red flow1 user interface image***

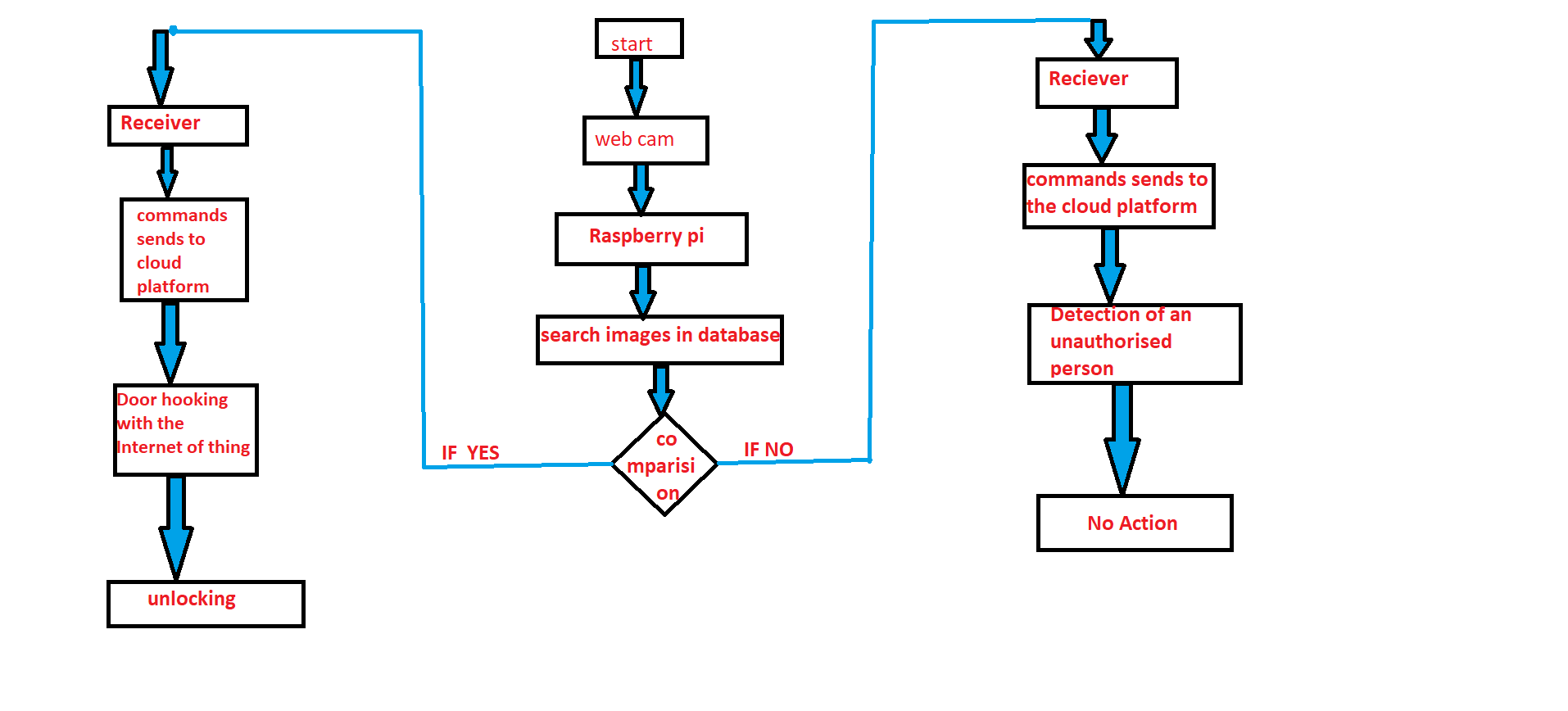


***Fig:Node red flow2***



***Fig:The node red flow2 user interface image***

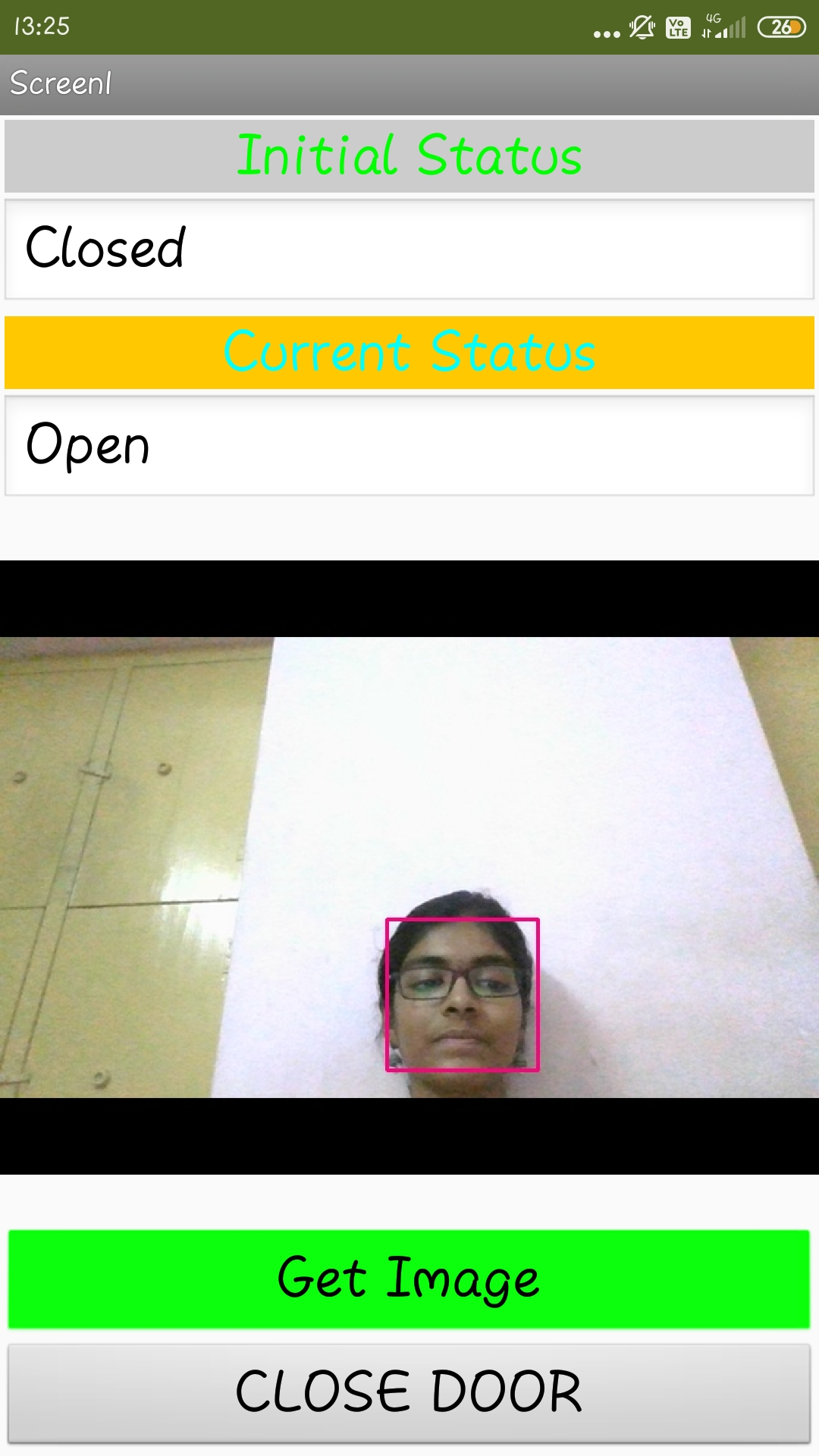
**5.Flow Chart:**



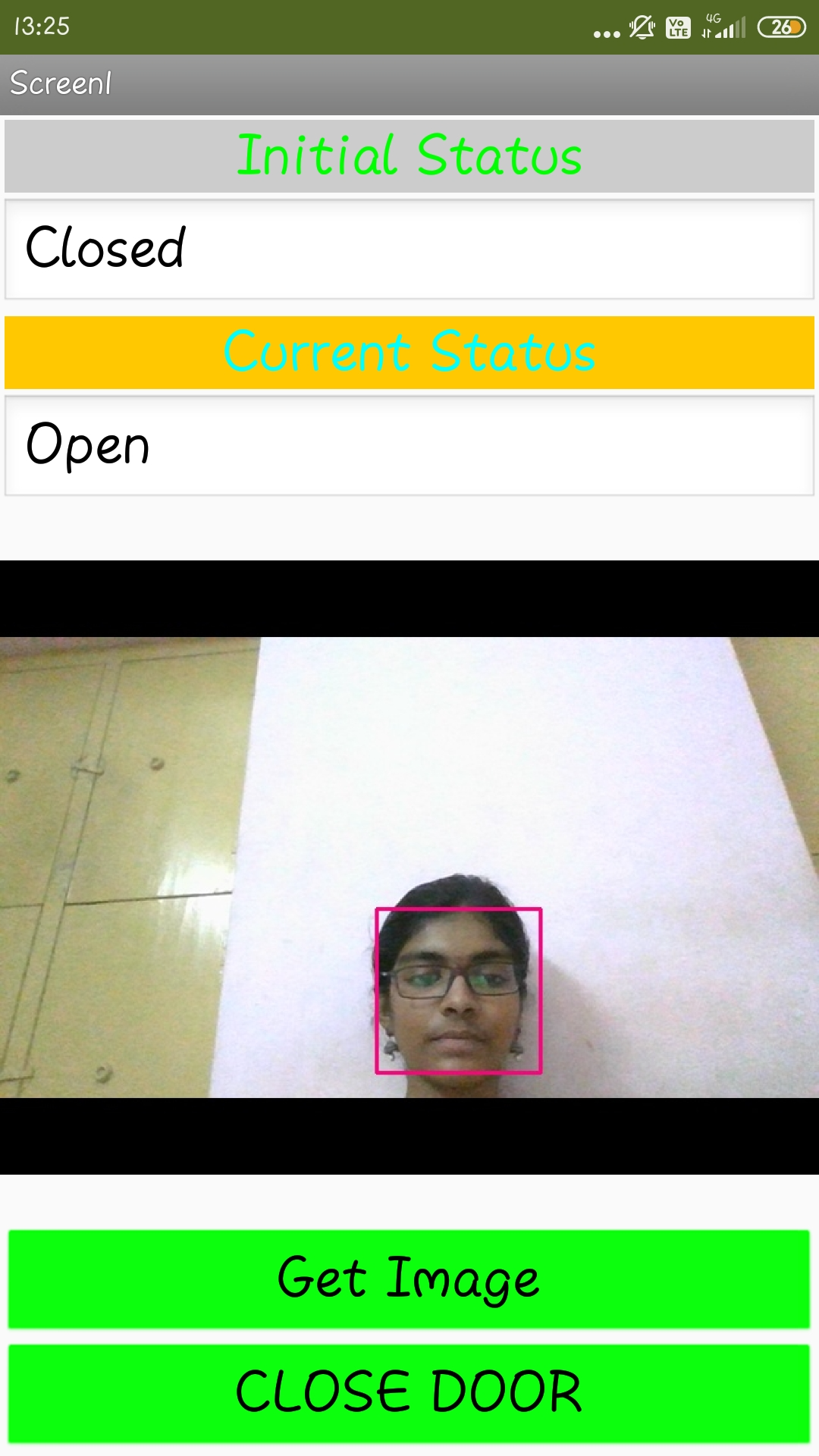
**Fig:Flow chart of Garage Door Opener**

**6.Result:**

Therefore, by using this application we can provide more safety to cars in the garage.We can also come over the problems like security issues, manual work and also can operate the door to be closed from anywhere through internet.So, one need not feel stressed regarding security.So, finally this project helps anyone who might be concerned about improving the safety of their place.This project is not just limited to be used in the garage,but can also be used for places that have access restricted to authorised persons only



***Fig:The screenshot of app upon clicking the Get Image button***



***Fig:Screenshot of the app upon clicking the Close Door button***

**7(a).Advantages:**

* Automatic door lock does not need a person to carry a key.You'll no longer need to worry about losing your key or fidgeting at the door to get it open.
* **Pick Proof:**

One of the key advantages of biometric door locks is the fact that they cannot be picked, as they do not make use of a key. This greatly increases the security of your home, and makes it much harder for thieves to get through the front door. The very fact that they will have to break down the door or otherwise conspicuously find a way into your home means that they will likely move on to an easier house.

* **Convenience:**

Though it does not benefit your home's security directly, having the added convenience of never having to worry about remembering your keys means that you will always have access to your home, as long as you still have your facedetction. This is a great feature, especially for elderly or forgetful homeowners.

* **Safety and Security** **:**

By the virtue of owning an automatic door, you can rest assured that your door is closed properly thus providing protection for your property from potential intruders.

**7(b).Disadvantages:**

* **Electricity:**

One of the disadvantage of automatic door locks is that electrically-powered systems may not function properly in the case of a power failure. But,most systems have battery backup systems as a fail-safe.

* **Expense:**

Automatic doors are definitely more pricey as compared to their manual counterparts. Due to the presence of many automatic features, automatic doors can be expensive to procure and mandates a high budget.

* **Cleaning** **:**

Automatic doors are more difficult to clean as you need to polish and oil internal parts like springs and bolts to avoid it from rusting. They also require constant services and inspections that all need to be paid for.

**8.Applications:**

This system is an efficient solution for operating many security applications like automatic garage door openers and can be used at placed that require high security allowing only approved personnel.

**9.Conclusion:**

Integrating features of all the software components used have been developed in it. Presence of every module has been reasoned out and placed carefully, thus contributing to the best working of the unit. Secondly, using IBM cloud with the help of growing technology, the project has been successfully implemented. Thus the project has been successfully designed and tested.

**10. Future Scope:**

Our project “ GARAGE DOOR OPENER USING OPENCV” is mainly intended to automated secure opening and closing of garage door using a mobile phone. The mobile phone present in the system uses auto answer function to open the door upon recognising the face.

This project can be upgrade by adding a button to open the door upon press feature also,so that the garage door opens for other friends or relatives that want to park car in our garage when they come to our house.

**11.Bibliography:**

* [***https://cloud.ibm.com/***](https://cloud.ibm.com/)
* [***https://sumauma.eu-gb.mybluemix.net/red/?#flow/39713f4d.baa23***](https://sumauma.eu-gb.mybluemix.net/red/?#flow/39713f4d.baa23)
* [***https://github.com/***](https://github.com/)
* [***http://ai2.appinventor.mit.edu/#6568356199399424***](http://ai2.appinventor.mit.edu/#6568356199399424)

**12.Appendix:**

*#for face detection*

*import cv2*

*import datetime*

*import numpy as np*

*#object storage*

*import ibm\_boto3*

*from ibm\_botocore.client import Config, ClientError*

*#for visual recognition*

*import json*

*from watson\_developer\_cloud import VisualRecognitionV3*

*#for cloudant db*

*from cloudant.client import Cloudant*

*from cloudant.error import CloudantException*

*from cloudant.result import Result, ResultByKey*

*#for sending data to IOT platform*

*import time*

*import sys*

*import ibmiotf.application*

*import ibmiotf.device*

*video=cv2.VideoCapture(0)*

*#ibm iot platform device details*

*organization = "kniccz"*

*deviceType = "rasberrypi"*

*deviceId = "123456"*

*authMethod = "token"*

*authToken = "12345678"*

*# Constants for IBM COS values*

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

*COS\_API\_KEY\_ID = "V1bhIqexbOdWLokhAOk5A1SIlxjrs8avX4ed8aYX6aH6" # eg "W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"*

*COS\_AUTH\_ENDPOINT = "*[*https://iam.cloud.ibm.com/identity/token*](https://iam.cloud.ibm.com/identity/token)*"*

*COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/9aacef9e53274facb815997addccf24e:e61f1bd8-d6fc-4e3e-b130-086b7fb84f24::" # eg "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c: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\_RESOURCE\_CRN,*

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

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

*endpoint\_url=COS\_ENDPOINT*

*)*

*try:*

*deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}*

*deviceCli = ibmiotf.device.Client(deviceOptions)*

*print("ibm iot connected")*

*except Exception as e:*

*print("Caught exception connecting device: %s" % str(e))*

*sys.exit()*

*deviceCli.connect()*

*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))*

*def cloudantstorage():*

*client =Cloudant("65eb56e4-a72d-4b13-b226-1626961c5389-bluemix", "899a62a1c1864b9ebf46c8b51ee287b2ffd3baf9314dee8400d1d1e1b33c137c",* [*url="https://65eb56e4-a72d-4b13-b226-1626961c5389-bluemix:899a62a1c1864b9ebf46c8b51ee287b2ffd3baf9314dee8400d1d1e1b33c137c@65eb56e4-a72d-4b13-b226-1626961c5389-bluemix.cloudantnosqldb.appdomain.cloud*](mailto:url="https://65eb56e4-a72d-4b13-b226-1626961c5389-bluemix:899a62a1c1864b9ebf46c8b51ee287b2ffd3baf9314dee8400d1d1e1b33c137c@65eb56e4-a72d-4b13-b226-1626961c5389-bluemix.cloudantnosqldb.appdomain.cloud)*")*

*client.connect()*

*#Provide your database name*

*database\_name = "insidegarage"*

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

*if my\_database.exists():*

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

*return my\_database*

*def visualrecog(x):*

*#Visual recognition*

*visual\_recognition = VisualRecognitionV3(*

*'2018-03-19',*

*iam\_apikey='ZCh5wehuL24VgR4VBcJCPg1ASZLMKrgjzoSeBW8QSE2Z')*

*with open(x, 'rb') as images\_file:*

*classes1 = visual\_recognition.classify(*

*images\_file,*

*threshold='0.55',*

*classifier\_ids='celebrity\_1626478899').get\_result()*

*#print(json.dumps(classes1))*

*a1=classes1['images'][0]['classifiers'][0]['classes'][0]['class']*

*#print(a1)*

*"""with open('re.jpg', 'rb') as images\_file:*

*classes1 = visual\_recognition.classify(*

*images\_file,*

*threshold='0.6',*

*classifier\_ids='cars\_269609070').get\_result()*

*a2=classes1['images'][0]['classifiers'][0]['classes'][0]['class']"""*

*if a1==("teja"):*

*string="Open"*

*else:*

*string="Don't open"*

*return string*

*#Transferring data to ibm iot platform*

*def myCommandCallback(cmd):*

*print("Command received: %s" % cmd.data)#Commands*

*def myOnPublishCallback(data):*

*#print ("Published ",ini,"to IBM Watson")*

*print ("Published ",data,"to IBM Watson")*

*#s=deviceCli.publishEvent("Initial\_Status", "json", ini, qos=0, on\_publish=myOnPublishCallback)*

*success = deviceCli.publishEvent("Current\_Status", "json", data, qos=0, on\_publish=myOnPublishCallback)*

*if not( success):*

*print("Not connected to IoTF")*

*time.sleep(2)*

*face\_classifier=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")*

*#It will read the first frame/image of the video*

*while True:*

*#capture the first frame*

*check,frame=video.read()*

*gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)#converts clr to black and white*

*#detect the faces from the video using detectMultiScale function*

*faces=face\_classifier.detectMultiScale(gray,1.3,5)#dimensions*

*print(faces)*

*#drawing rectangle boundries for the detected face*

*for(x,y,w,h) in faces:*

*cv2.rectangle(frame, (x,y), (x+w,y+h), (127,0,255), 2)#127,0,255 is value for pink clr,so we have pink rectangle*

*cv2.imshow('Face detection', frame)*

*picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")*

*x=picname+'.jpg'*

*cv2.imwrite(x,frame)*

*multi\_part\_upload("insidegarage", x,x)*

*my\_database=cloudantstorage()*

*json\_document={"link":COS\_ENDPOINT+"/"+"insidegarage"+"/"+x}*

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

*# Check that the document exists in the database.*

*if new\_document.exists():*

*print(f"Document successfully created.")*

*a=visualrecog(x)*

*deviceCli.commandCallback = myCommandCallback*

*data = { "Initial\_Status":"Closed","Current\_Status": a}*

*myOnPublishCallback(data)*

*deviceCli.commandCallback = myCommandCallback*

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

*Key=cv2.waitKey(1)*

*if Key==ord('q'):*

*#release the camera*

*video.release()*

*deviceCli.disconnect()*

*#destroy all windows*

*cv2.destroyAllWindows()*

*break*

*#*