PROJECT DESCRIPTION DOCUMET

Intelligent Access Control System for Safety Critical Areas using IBM IoT Platform

Version No.	Date	Preparedby	Reviewd by	Change History
1.0	17/10/2020	VIMALADEVI M	<SmartInternz Reviewer>	Initial Release

1. Introduction

This document describes the project "Intelligent Access Control System for Safety Critical Areas using IBM IoT Platform".Workers in the safety critical areas are required to wear a safety helmet and shoes to prevent any accident occuringin the workspace.

The proposed system will be integrated near the entrance of the working area which will detect the person and check whether the person is wearing a safety helmet and shoes using the IBM Visual Recognition service. If the person is taking safety precautions the door should be opened or else it should generate the voice alerts why he can't enter inside using IBM Text to speech service. All the images will be stored in the cloud so that the admin can access the images. The admin can access the images of the person entering using a web page and also from a mobile app.

2. Installation

The project is developed using the IBM Cloud platform in Python programming language. The following are the details of the software tools/services used.

1. Python version: 3.7.4
2. IBM cloud account
3. opencv-python 4.4.0.44
4. IBM Watson Visual Recognition
5. IBM Watson Text to Speech
6. IBM Cloudant
7. IBM Cloud Object Stroage
8. IBM IoT Platform
9. IBM Cloud Foundary App Node RED
10. MIT App Inventor

The Python SDKs for the corresponding services are installed from IBM Cloud.

3. Algorithm

Client Module:

Step 1: Import the required librararies
Step 2: Initialize authentication IBM cloud services
Repeat:

Step 3: Capture image from the live video stream
Step 4: If face is detected, Store the image in cloudant DB; Else goto Step 3
Step 5: If both helmet and shoes are present in the captured image, goto Step 7; Else goto Step 8
Step 7: Generate Success authorization message and publish to IoT device
Step 8: Generate Failure authorization message and publish to IoT device
Step 9: Receive command from Web UI and Mobile App and communicate to IoT device.

until exit.

4. Code Snippets

4.1 Capture frame from live video

```
def captureImage():
    vidObj = cv2.VideoCapture(0,cv2.CAP_DSHOW)
    _, image = vidObj.read()
    cv2.imwrite(picname , image)
```

4.2 Check for face

```
def checkFace():
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    img = cv2.imread(picname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    if len(faces):
        global isFace
        isFace = True
```

4.3 Check for the presence of Helmet and Shoes

```
def securityCheck():
    isHelmet = False
    isShoes = False
    visual_recognition = VisualRecognitionV3(
        '2018-03-19',
        iam_apikey='ErHI67WnIe7UQ8-Aq5AbBc_8qX03LkXcLPnJe0NS1fyG')

    with open(picname, 'rb') as images_file:
        cl = visual_recognition.classify(
            images_file,
            threshold='0.6',
        classifier_ids='default').get_result()

    file1 = open("MyFile_1.txt","w+")
    for x in cl["images"][0]["classifiers"][0]["classes"]:
        print(x["class"])
        file1.write(x["class"])
        file1.write("\n")
    file1.close()

    with open("MyFile_1.txt") as file:
        for line in file:
            if 'helmet' in line:
                isHelmet = True
            elif 'headdress' in line:
                isHelmet = True
            elif 'hard hat' in line:
                isHelmet = True
            elif 'shoes' in line:
                isShoes = True
            elif 'shoe' in line:
                isShoes = True
            elif 'footwear  ' in line:
                isShoes = True

    if isHelmet == True and isShoes == True:
        global allok
        allok = True
```

4.4 Upload the image to cloud

```
def uploadtodatabase():
    my_database = client.create_database(database_name)
    uploadImage(bucket_name, picname, "C:/python/python37/"+pic+".jpg")
    if my_database.exists():
            print("'{database_name}' successfully created.")
            json_document = {
                    "_id": pic,
                    "link":COS_ENDPOINT+"/14bucket14/"+picname
                    }
            new_document = my_database.create_document(json_document)
            if new_document.exists():
                print("Document '{new_document}' successfully created.")
```

```python
def uploadImage(bucket_name, item_name, file_path):
    print("Starting large file upload for {0} to bucket: {1}".format(item_name, bucket_name))

    # set the chunk size to 5 MB
    part_size = 1024 * 1024 * 5

    # set threadhold to 5 MB
    file_threshold = 1024 * 1024 * 5

    # set the transfer threshold and chunk size in config settings
    transfer_config = ibm_boto3.s3.transfer.TransferConfig(
        multipart_threshold=file_threshold,
        multipart_chunksize=part_size
    )

    # create transfer manager
    transfer_mgr = ibm_boto3.s3.transfer.TransferManager(cos_cli, config=transfer_config)

    try:
        # initiate file upload
        future = transfer_mgr.upload(file_path, bucket_name, item_name)

        # wait for upload to complete
        future.result()

        print ("Large file upload complete!")
    except Exception as e:
        print("Unable to complete large file upload: {0}".format(e))
    finally:
        transfer_mgr.shutdown()
```

4.5 Text to Speech conversion

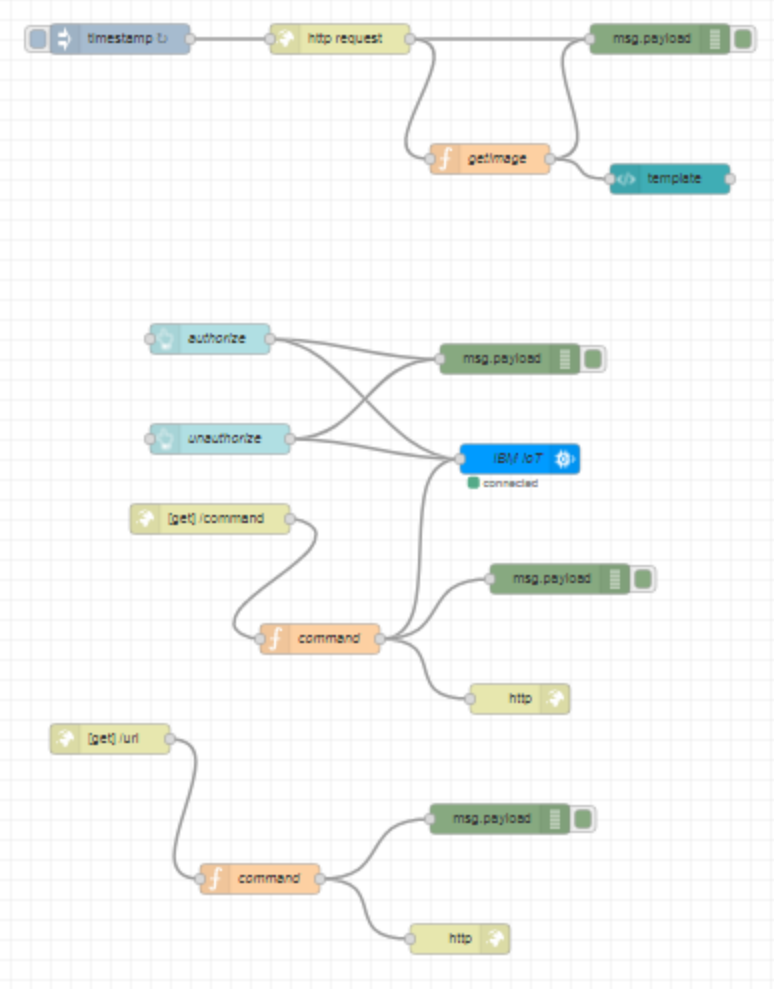
```
def generateSpeech():
    from ibm_watson import TextToSpeechV1
    from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

    authenticator = IAMAuthenticator('xIXUTpjZVGjozSwbSCoQh90saoL3Tc9Hbeioh14xgF-V')
    text_to_speech = TextToSpeechV1(
        authenticator=authenticator
    )

    text_to_speech.set_service_url('https://api.us-south.text-to-speech.watson.cloud.ibm.com/instanc

    if allok == True:
        with open('allok.wav', 'wb') as audio_file:
            audio_file.write(
                text_to_speech.synthesize(
                    'Safety check successful. Please enter.',
                    voice='en-US_AllisonVoice',
                    accept='audio/wav'
                ).get_result().content)

    elif allok == False:
        with open('notok.wav', 'wb') as audio_file:
            audio_file.write(
                text_to_speech.synthesize(
                    'Safety check failed. Please retry.',
                    voice='en-US_AllisonVoice',
                    accept='audio/wav'
                ).get_result().content)
```

4.6 IoT device control

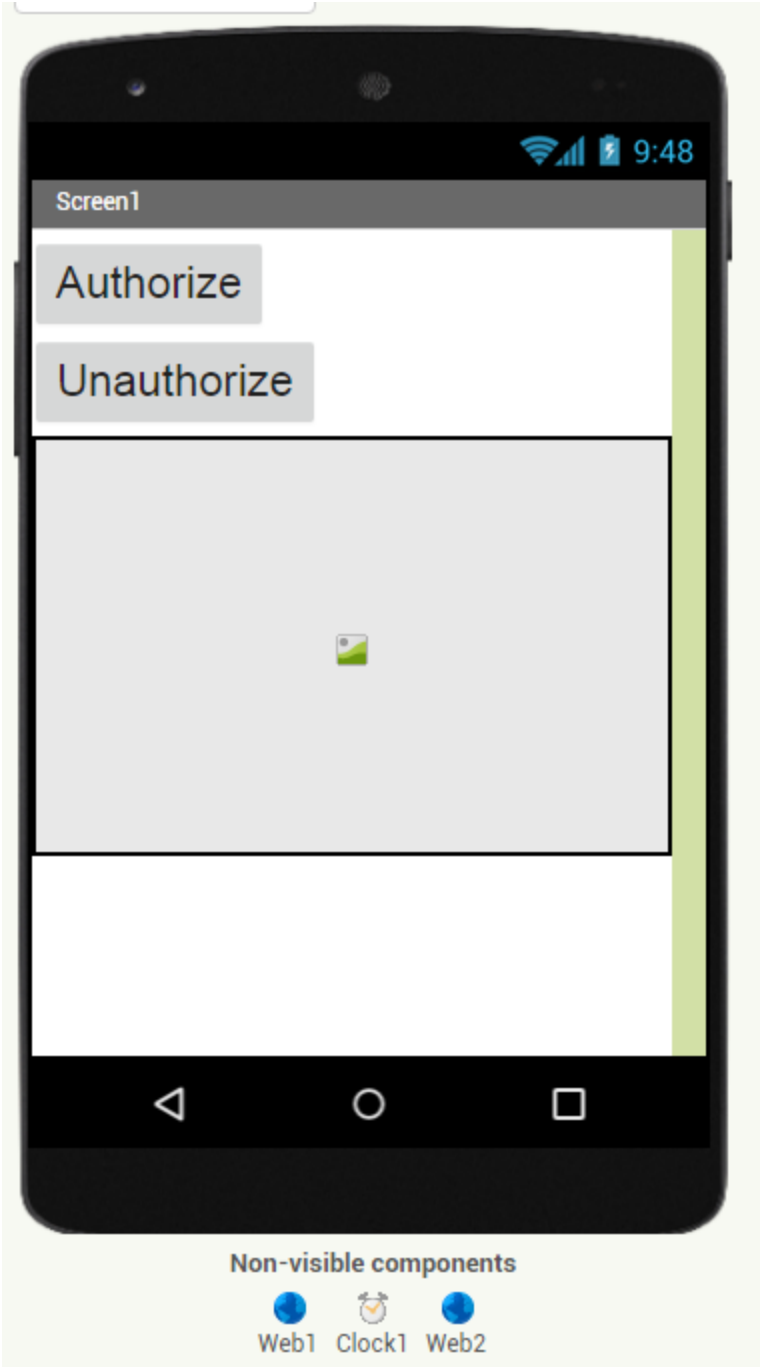
```
def deviceControl():
    def myOnPublishCallback():
        print ("Published data to IBM Watson")
    data = {'person' : allok}
    success = deviceCli.publishEvent("Data", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not (success):
        print("Not connected to IoTF")
        time.sleep(1)
    deviceCli.commandCallback = myCommandCallback
```

```
def myCommandCallback(cmd):
        print("Door outhorization message received: %s" % cmd.data)
        print(cmd.data['command'])

        if(cmd.data['command']=="open"):
                print("Authorization Success. Please enter.")

        if(cmd.data['command']=="close"):
                print("Authorization Failed. Please try again.")
```

4.7 Node-RED design



4.8 Mobile App Design


9:48
Screen1
Authorize
Unauthorize
Non-visible components
Web1
Clock1
Web2

5. Detailed Requirements Tracking

Req.ID	Req. Description	Code Snippets Reference
01	Detecting The Face In Video Streaming	4.1
02	Capturing Frames Using The Camera	4.1
03	Visual Recognition Object Detection To Detect Face, Helmet And Shoes	4.2, 4.3
04	Text To Speech Conversion	4.5
05	Transferring The Images To The IBM Cloud Object Storage	4.4
06	Node-Red Flow To Get Image Urls From Cloudant DB	4.7
07	Configure the IoT device to receive authorization commands	4.6

08	HTTP Requests To Send The Image URL To The Mobile App	4.7
09	Design Mobile App to display image	4.8
10	Design Mobile App to send authorization commands to IoT device	4.8

6. Conclusion

This document presented the details of the requirements of the project "Intelligent Access Control System for Safety Critical Areas using IBM IoT Platform". The services used and the modules accessed are presented along with the algorithm of the cliet application. The requirements are elicited and are tracked in code successfully.