

SPRINT 4

DATE	14 November 2022
TEAM ID	PNT2022TMID04713
PROJECT NAME	IOT BASED SMART CROP PROTECTION FOR AGRICULTURE
MAXIMUM MARK	20 MARKS

CODE:

```
import CV2
import numpy as np
import wiotp.sdk.device
import playsound
import random
import time
import datetime
import ibm_boto3
from ibm_botocore.client import Config, ClientError

#CloudantDB
from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import Service_pb2_grpc
stub = serviceVpb2_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
# This is how you authenticate.
metadata = (('authorization', 'Key bc885e5165d74ef48f42f6f6a2c9eb87'),)
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list
available at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "f6Ap-ct18n07S9UEL7XPDAF?1 TOomePLLUQOzqunAzDS" # eg
*WOOY:RNLW{a3ç7}NBqpdB-2y3fTEFBIQQManc--P3byk"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "crn:vl:bluemix:public:cloud-object-
storage:global:a/eb644a3fda97449b988c23eeef263ed6:19SableS-0d9d-420f-8e4a-
98d968C04263:." # eg "crn:vi:bluemix:public:cloud-object-
clientdb = Cloudant ("apikey-v2-16u3crmapkghnxfdikvpssohSfwezrmuupSfvSg3ubz",
"b0ab119f45d3e625Seabb978e7e2f0el", url="https://apikey-v2-16u3crmdpkghxe fai
kypssohSfwezrmuupsfv5g3ubz :b0ab11s
clientdb.connect ()

# Create resource
cos = ibm boto3. resource ("s3",
    ibm_api_key_id-COS_API_KEY_ID,
```

```

ibm_service_instance_id=COS_RESOURCE_CRN,
4ibm_auth_endpoint=Co3_aUTH_ENDPOINT,
config=Config(signature_version="oath"),
)
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 myCommandcallback (cmd):
    print ("Command received: %s" cmd.data)
    command=cmd.data['command']
    print (command)
    if (command=="lighton") :
        print('lighton')
    elif (command=="lightoff") :
        print('lightoff')
    elif (command== 'motoron'):
        print ('motoron')
    elif (command=='motoroff'):
        print ('motoroff')
myConfig = {
    "identity": {
        "orgId": "hj5fmy",
        "typeId": "NodeMCU",
        "deviceId": "12345"
    },
    "auth": {
        "token": "12345678"
    }
}

```

```

}
client = wiotp.sdk.device.Device client (config-myConfig, logHandlers-None)
client.connect()
database_name= "sample"
my_database = clientdb.create_database (database_name)
if my_database.exists():
    print (f" (database_name]" successfully created.")
cap=cv2.VideoCapture ("garden.mp4")
if (cap. isopen() ==True):
    print ('File opened')
else:
    print ('File not found')

while (cap.isopen() ):
    ret, frame = cap.read()
    gray cv2.cvtColor (frame, cv2.COLOR_BGR2GRAY)
    ims cv2.resize (frame, (960, 540))
    cv2.imwrite('ex.jpg', ims)
    with open ("ex.jpg", "rb") as f:
        file_bytes = f.read()
    # This is the model ID of a publicly available General model. You may use any other
    public or custom model ID.
    request service_pb2. PostModelOutputsRequest(
        model_id='aaa03c23b3724a16a56b629203edc62c1',
        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))
    detect=False
    for concept in response. outputs [0].data.concepts:
        #print ('1812s: %.2f' (concept.name, concept.value))
        if (concept.value>0.90):
            #print (concept.name)
            if (concept.name=="animal"):
                print ("Alert! Alert! animal detected")
                playsound.playsound ('alert.mp3')
                picname=datetime.datetime.now().strftime ("y-m-3d-3H-M")
                cv2.imwrite(picname+'.jpg', frame)
                multi_part_upload ('gnaneshwar', picname+'.jpg', picname+'.jpg')
                json_document={"link":COS_ENDPOINT+'/'+'gnaneshwar'+'/'+picname+'.jpg'}
                new_document = my_database.create_document (json_document)
                if new_document.exists():
                    print (f"Document successfully created.")
                    time.sleep (5)
                    detect True
    moist=random.randint (0, 100)
    humidity random. randint (0,100)
    myData={'Animal': detect, 'moisture' :moist, 'humidity' :humidity}

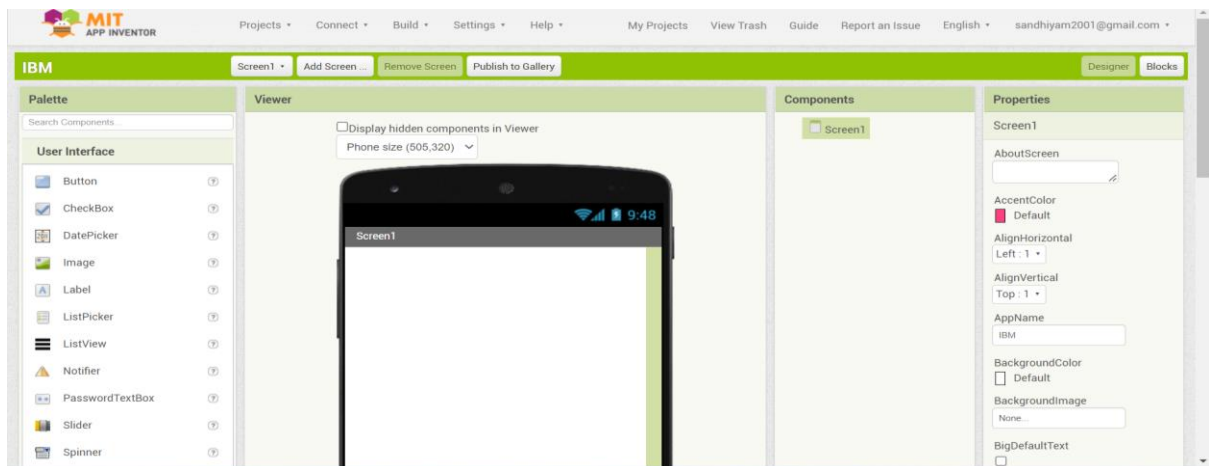
```

```

print (myData)
if (humidity!=None):
    client.publishEvent (event Id="status", msgFormat="json", data=myData, qos=0, on
Publish=None)
    print("Publish ok..")
    client.commandCallback = myCommandCallback]
cv2.imshow ('frame', ims)
if cv2.waitKey (1) & 0xFF == ord ('q') :
    break
client.disconnect()
cap. release ()
cv2.destroyAllWindows()

```

STEP 1: MIT App inventor to design the app



STEP 2: Customize the app interface to display the value

