IoT Based Smart Attendance System using IBM Cloud Services and OpenCV

Ayush Kumar Singh
19BEI0134
(Electronics and Instrumentation)

Sanskriti Binani
19BEI0053
(Electronics and Instrumentation)

Aman Mandal
19BEI0077
(Electronics and Instrumentation)

Aditya Om
19BEI0044
(Electronics and Instrumentation)

Shraman Jain
19BEI0030
(Electronics and Instrumentation)

Under

Dr. Abhishek G (VIT)

Digital Image Processing

Project Component

Vellore Institute of Technology

Fall Semester 2021-22

Abstract

Attendance is a must in any organization. Keeping an attendance register on a daily basis is a challenging and time-consuming task. There are several automated ways available for the same, such as biometrics, RFID, eye detection, speech recognition, and many more. This project deals with one of the most efficient and accurate Attendance systems based on facial Recognition. Face recognition provides an accurate method that solves ambiguities such as fraudulent attendance and time consumption since it is understood that any human's primary identity is their face. This project uses OpenCv library with python to register, train, and recognize face of a particular individual through a front-end on Node-RED dashboard and then store the data of the individual whose is not recognized into IBM Cloud Object and Cloudant database and notify the admin with a SMS consisting of the link where the image taken of that individual is stored. It also uses the excel sheet to mark the people present on the particular day.

Contents

1. Introduction……………………………………………………….**5**

2. Literature Survey…………………………………………………..**6**

3. Problem Formulation…………………………………………..…..**7-9**
   [A] Proposed Architecture……………………………………….…7-8
   [B] Face Detection, Training of Image and Face Recognition…..…8-9

4. Result Analysis…………………………………..………**10-17**
   [A] Registration……………………………………………………10-13
   [B] Attendance……………………………………………...13-17

5.Conclusion…………………………….……………**18-19**
   [A] Salient Points……………………………………………18
   [B] Novelty…………………………………………....18-19
   [C] Future Aim………………………………………….…...19

6. References……………………………………………………..**20**

7.Appendix…………………………..…………..……**21-27**
   [A] Python Code……………………………………………..21-26
   [B] Python Code for updating Excel sheet…..…………………..26-27
   [C] Node-RED flow……………………………….………...….27

List of Figures

1. Flow Chart for Proposed Architecture……………………………….8

2. Process Flow Chart for Face Detection, Training and Recognition….9

3. Web Dashboard…………………………………………………...10

4. Enter Unique ID and Click CAPTURE IMAGE……………………11

5. Camera window Open to capture multiple face images…………….11

6. Storing Captured Multiple Face Image in Local Storage…………...12

7. Confirmation regarding capturing of Images………………………..12

8. Confirmation regarding successful registration……………………..13

9. Displaying Unique ID of the Matched Data………………………...14

10. Creation of new Excel File for current Date…………………….…14

11. Attendance entry in Excel File…………………………………….…15

12. Image stored in Cloud Object Storage……………………………..…16

13. Global link of the image stored in Cloudand Database…………....16

14. Getting Intrusion image from Cloudand Database by clicking on CHECK INTRUSION button…………………………………….....17

15. SMS received containing link……………………………………...17

16. Image view by clicking on link…………………………………….17

I. Introduction

In today's world where automation has advanced significantly and is now present in various fields of industry, we decided to design an automated Attendance system based on facial recognition. The traditional method of taking attendance in an institution is a tedious process and the manual labor required to maintain and monitor attendance sheets is arduous. Taking attendance for numerous department students also takes a significant amount of time. To circumvent these issues, we propose a system that recognizes a person by comparing live capture digital image data with a previously recorded image of that person and then mark attendance accordingly.

Face detection, recognition, and storage are the three major components of our project. The training phase is the system's first procedure. All students' photos must be recorded and saved in the training dataset. This is done by developing a python code with the help of the OpenCV library to register new faces by taking multiple images of the person based on the UNIQUE ID of the person. The next phase is the testing phase, in which the system is supposed to recognize the face of the person in front of the camera with the help of trained data. If the face matches any ID in the trained dataset, the attendance is recorded. Further on integrating it with IoT, access will be accepted or denied depending on face detection. In the third phase if the access is denied then the image is stored in IBM Cloud Object Storage and the link for the image is stored in the IBM Cloudant database for future reference and security purpose. The link is also sent via fast2sms API integration with python to the admin’s registered mobile number so that he/she can immediately view the face whose access is denied. Lastly, with the help of a python, we create a function to store every day’s attendance in an excel sheet. This helps automatically maintain daily records of the student’s attendance.

The main contribution of this project is to develop an improved automatic attendance system by analyzing several face recognition technologies.

II. Literature Review

The main purpose of this paper is to develop a smart attendance management system using facial recognition that addresses the problems found in other automated systems in modern-day society, this paper focuses on the development of a smart attendance management system using facial recognition. The present authors [10] used the cvtColor method to convert the captured images into grayscale and Haarcascade classifiers to perform face detection as it performs detection of faces with high efficiency and high accuracy even with different orientations of the faces to be detected under different lighting conditions. Also, the system contains an additional module of gender classification by using the Fisherfaces algorithm. In another implementation of a similar system, Sawhney et al [7], proposed that the system consists of two databases, a student database, and an attendance database. The student database contains details about each student in a specific class. On the other hand, the attendance database is used to keep track of and mark the attendance of students at a particular lecture. The AdaBoost algorithm and Principle Component Analysis (PCA) are used for facial recognition. In face recognition, PCA is used to reduce the number of variables. The student is allowed access to the classroom once their face has been successfully recognized. In another research [5], Sajid Akbar et al incorporated Radio Frequency Identification (RFID) along with facial recognition to detect the authorized students and counts as they get in and get out from the classroom. With the face and identification number of every student recorded, their system can recognize each student and count them when they enter the class by confirming their face and RFID-based identity cards. Although Some of the popular object detection algorithms are back propagation neural network, region-based convolution network (RCNN), faster RCNN, single-shot detector. [2] Sikandar et al's structure is based on the YOLO V3 algorithm for face detection and Microsoft Azure using face API for face recognition (face database). For the detection of faces, they used the YOLO V3 algorithm which is basically used for the detection of a large number of objects, more than 1000 objects. Counting is also done via YOLO V3 and the system informs about the number of students present in a class. After that recognition of faces is necessary which is done by using Azure face API that works on facial features to detect and confidently recognize a face for identification. After that face database will match the data of all students and generate a spreadsheet for the specific time and date, attendance will be marked for every student.

III. Problem Formulation

[A] Proposed Architecture

As shown in the Figure – 3.A, our Proposed Architecture consists of 9 major blocks.

1. Node-RED – Node-RED acts as a user interface where the user can interact with the provided system. They can use the Node-RED dashboard to capture images of a person, train the images captured with the unique employee ID, and also to recognize the person at any point of time. They also get a feature to check the image of the person detected whose data has not been trained. It sends the request to the IBM IoT device to get the required data or to perform a particular task and in return receives a bulk of data from the IBM IoT device node in the JSON format which is then filtered by the function node and then displayed on the dashboard.

2. IBM IoT Device – It acts as a buffer between the CPU and the user interface. When Node-RED requests a process from the CPU (Python integrated with OpenCV) the IBM IoT device passes the message to the CPU and waits for the work to be completed. Once the work is done, the CPU sends a response to the Node-RED via IBM IoT Device. Here all the data transfer is done in the JSON format.

3. Python with OpenCv – It acts a Central Processing Unit (CPU) or the brain of the system. It communicates to all the blocks of the system directly or indirectly and is also responsible for processing the program efficiently stored in the system.

4. Virtual Camera – It acts as the eyes of the system and works in two modes. In the first mode it works for the face detection where it captures multiple images of the person detected and sends the data to the CPU. In the second mode it works for the face recognition where it captures and sends image to python and then receives the Name of the person detected (if image data is present) and displays in the camera window by creating a boundary around in face of the person.

5. Image Dataset – It is used as a database for storing the multiple images that the camera captures along with the unique ID in a local folder specified by the programmer while in face detection mode. When the raw data is trained then the trained data is also stored in the same location and is used when face recognition is done.

6. IBM Cloud Object Storage – It comes under picture when the image detected by the camera is not recognized. When this happens the image captured by the camera is send to the CPU from where it is uploaded to the IBM Cloud Object storage with the title as that particular date and time. When the image is uploaded as an object, the Cloud storage creates a unique and global link which can be used to view the image.

7. IBM Cloudand Database – The Cloudand database is a NoSQL database that creates a document for each entry and stores the data in key-value pain in the document. Here the link

which we get from the Cloud Object storage for an image after uploading is stored and is send to the template node in Node-RED when requested using the HTTP protocol. Using this process the image can be displayed directly on the user interface by just clicking a button.

8. Fast2SMS – The Fast2SMS API is used to send the user an alert SMS including the link of the image provided by the Cloud Object Storage. So that if they can't access the web-dashboard they need to just click on the link in the SMS and they can check the image and further take the required action.

9. Excel File – When the Face of a person is finally recognized the Data is entered into the Excel sheet which include the Name of the person, Unique ID of person, Attendance status and the time of facial recognition.

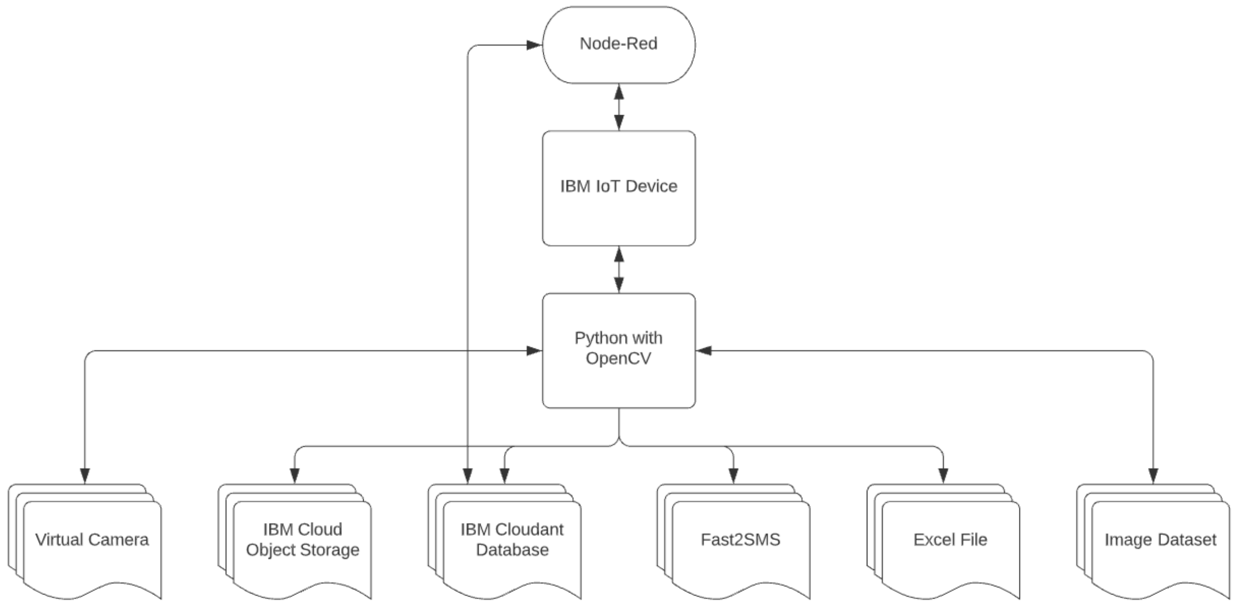


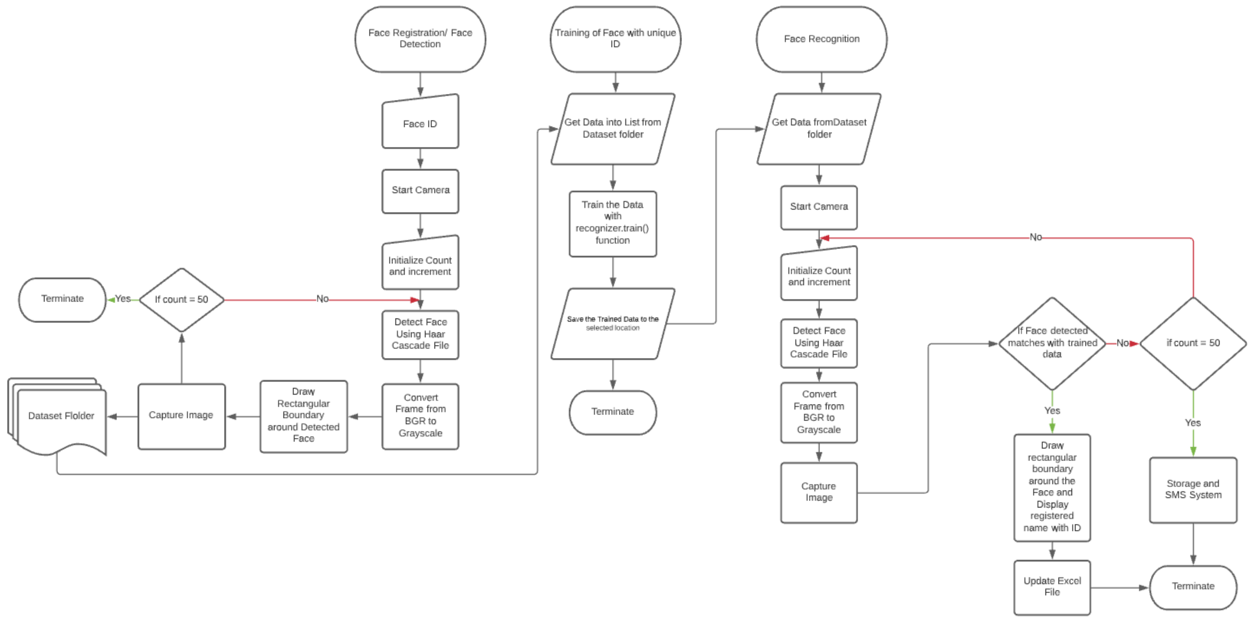
Figure – 3.A: Flow Chart for Proposed Architecture

[B] Face Detection, Training of Image and Face Recognition

1. Face Detection – In this process (As shown in figure-3.B) first the system accepts the Unique ID of the person and then start the virtual camera. The counter gets initialized and the face is detected using the Haar-cascade frontal face algorithm. The frame is now converted in grayscale from BGR and a rectangle boundary is drawn around the detected Face and finally the image gets captured and stored in the specified Dataset folder. Same way multiple images of the detected face gets captured and stored till the loop terminated according to the specified conditions.

2. Training of Images – As seen in figure-1.B, once the multiple images are stored in the folder they are transferred into the list using the np.array() function symmetrically with different list for Face and unique ID respectively. Now the recogizer.train() function is called with the first parameter as faces and the second parameter as unique IDs and returns the trained data accordingly. This trained data is then stored in the specified location for future reference.

3. Face Recognition – When coming to the facial recognition process (Figure- 1.B), first the trained data is fetched from the specified folder using recognizer.read() . Once the data is fetched the camera is started and face is detected using the Haar-cascade file . The frame is then converted into grayscale from BGR and the image of the face detected is captured and then matched with the trained data. If the face is recognized then the a rectangular boundary is dawn around the recognized face and the person's Name with unique ID is displayed. Then the attendance details are uploaded in the Excel sheet else the count is checked and the system continues to capture and test the face with the trained data. If the face is not recognized even at last iteration then the process then the image of face is stored in the cloud object storage and the link for the image is uploaded in the cloudand data base and an SMS to the user is send containing then link and then the process terminates.



Face Registration/ Face Detection
Face ID
Start Camera
Initialize Count and increment
Detect Face Using Haar Cascade File
Convert Frame from BGR to Grayscale
Draw Rectangular Boundary around Detected Face
Capture Image
Dataset Flolder
If count = 50
Yes
No
Terminate
Training of Face with unique ID
Get Data into List from Dataset folder
Train the Data with recognizer.train() function
Save the Trained Data to the selected location
Terminate
Face Recognition
Get Data fromDataset folder
Start Camera
Initialize Count and increment
Detect Face Using Haar Cascade File
Convert Frame from BGR to Grayscale
Capture Image
If Face detected matches with trained data
Yes
No
Draw rectangular boundary around the Face and Display registered name with ID
Update Excel File
if count = 50
Yes
No
Storage and SMS System
Terminate

Figure – 3.B: Process Flow Chart for Face Detection, Training and Recognition.

IV. Result Analysis

As seen in Figure- 4.A , we have a web dashboard that acts as the user interface of the system. The interface contains:

1.Registration Group for capturing and registering of image with the unique ID (Here employee ID).
2.Attendance Group for recognizing and update the attendance of the person present.
3.Recent Activity Group for displaying the real time Activity happening.
4.Security Window Group for displaying the intrusion/unauthorized person detected recently.

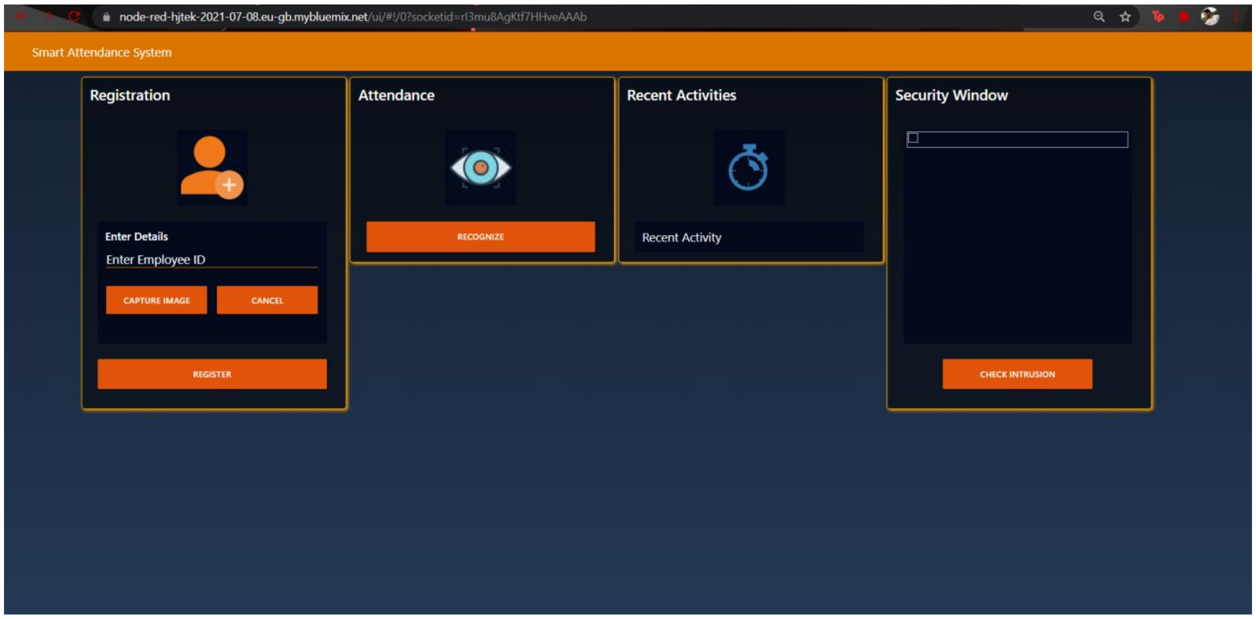


Figure - 4.A: Web Dashboard

The Analysis for this system has been divided into 2 different scenarios:

[A] Registration

For the registration purpose the user has to enter the unique ID and then click on the CAPTURE IMAGE button (Figure – 4.A.i). Once the button is pressed the camera window opens and captures multiple images of the person (Figure – 4.A.ii), stores the image in local database (Figure – 4.A.iii)and returns a confirmation as a notification and is also highlighted in the Recent Activity group(Figure – 4.A.iv).

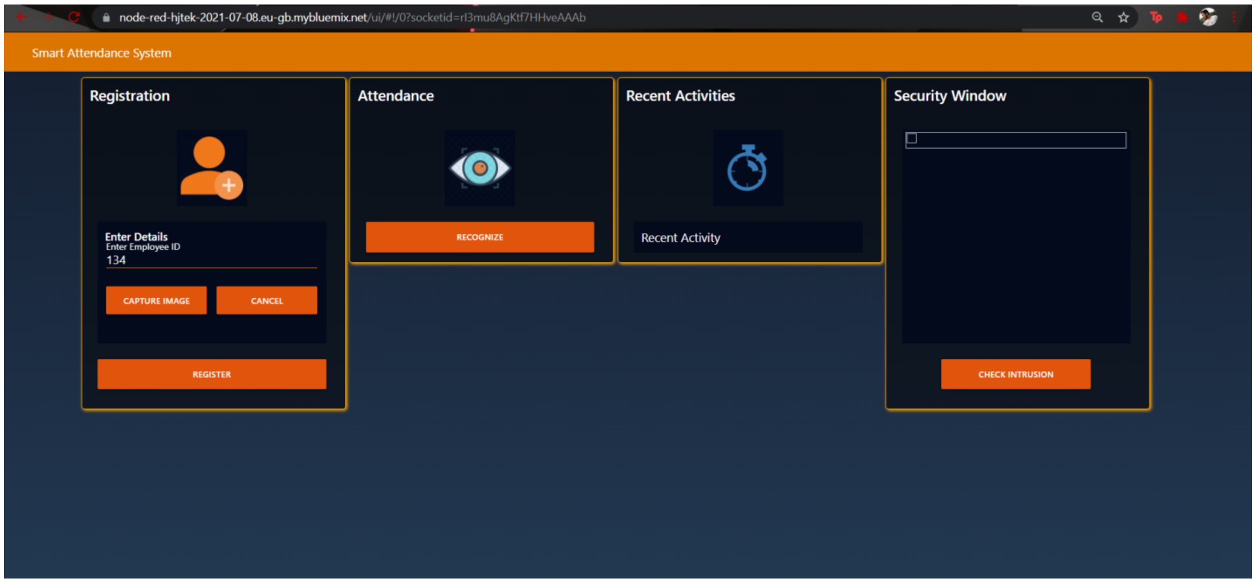


Figure – 4.A.i: Enter Unique ID and Click CAPTURE IMAGE

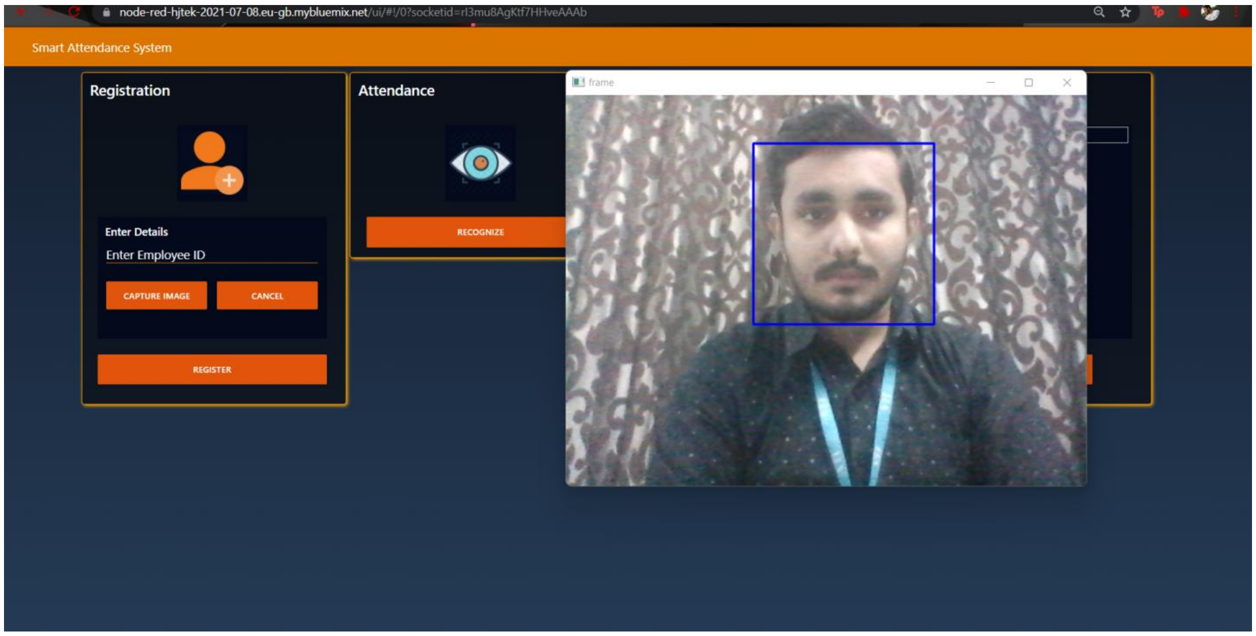


Figure – 4.A.ii: Camera window Open to capture multiple face images

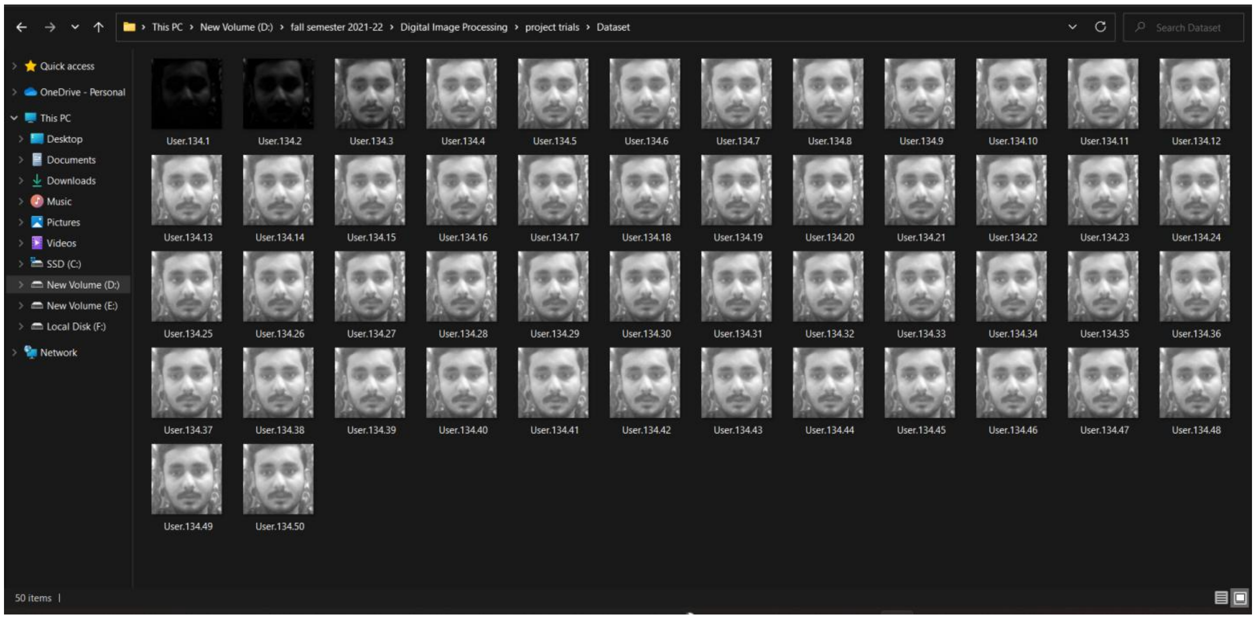
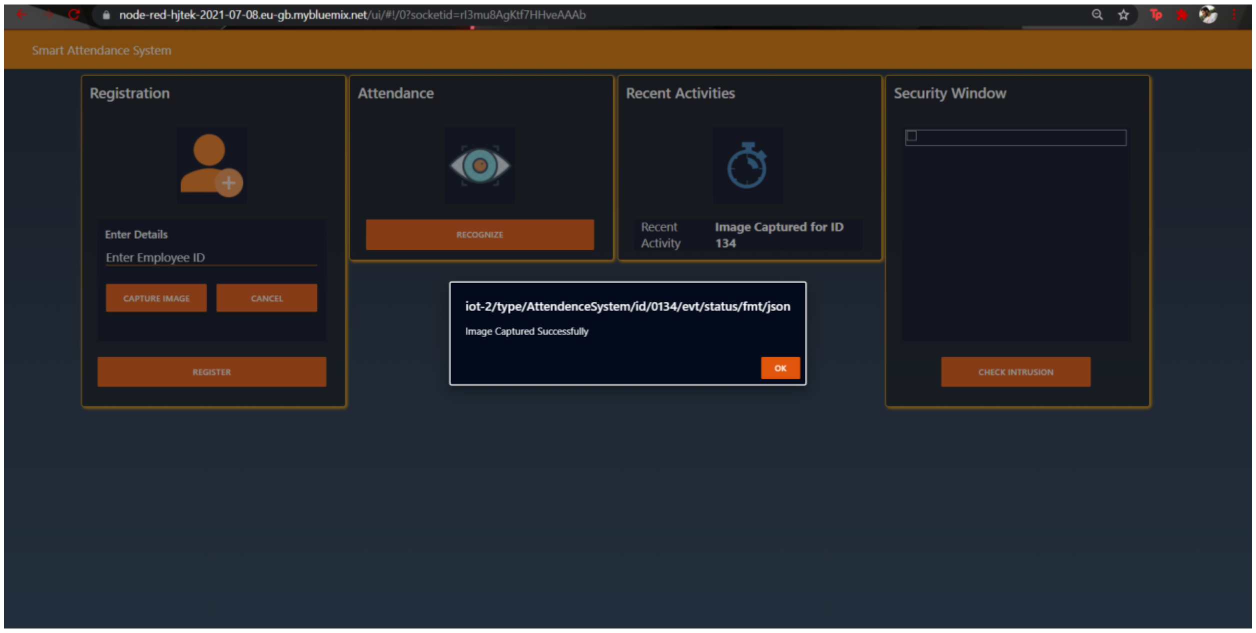


Figure – 4.A.iii: Storing Captured Multiple Face Image in Local Storage



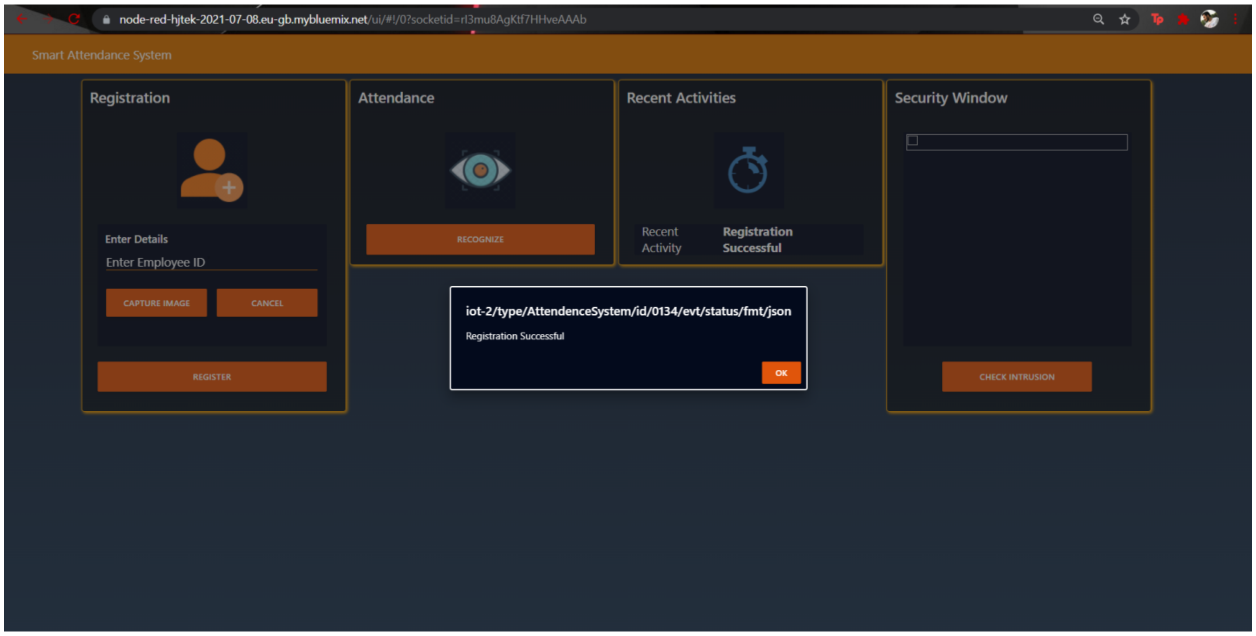
Recent Activity: Image Captured for ID 134

iot-2/type/AttendenceSystem/id/0134/evt/status/fmt/json

Image Captured Successfully

Figure – 4.A.iv: Confirmation regarding capturing of Images

Once the confirmation is received the user clicks on to the REGISTER button to get the image data trained with the unique ID. As the data is trained it is stored again at a local storage and confirmation is seen (Figure – 4.A.v).



Registration Successful

iot-2/type/AttendenceSystem/id/0134/evt/status/fmt/json

Registration Successful

Figure – 4.A.v: Confirmation regarding successful registration

[B] Attendance

In the attendance process the user clicks on the RECOGNIZE button on the web dashboard. Once it is done the camera is opened and captures your face image to match is with the trained data.

If the trained data matches with the image captured then the Unique ID with config. Value is displayed on the top of the rectangle created around the face (Figure – 4.B.i) and then a new Excel file is created (Only for 1st entry each day) with name as attendance followed by current date (Figure- 4.B.ii) and then the person's name along with status and current time in different columns respectively (Figure- 4.B.iii).

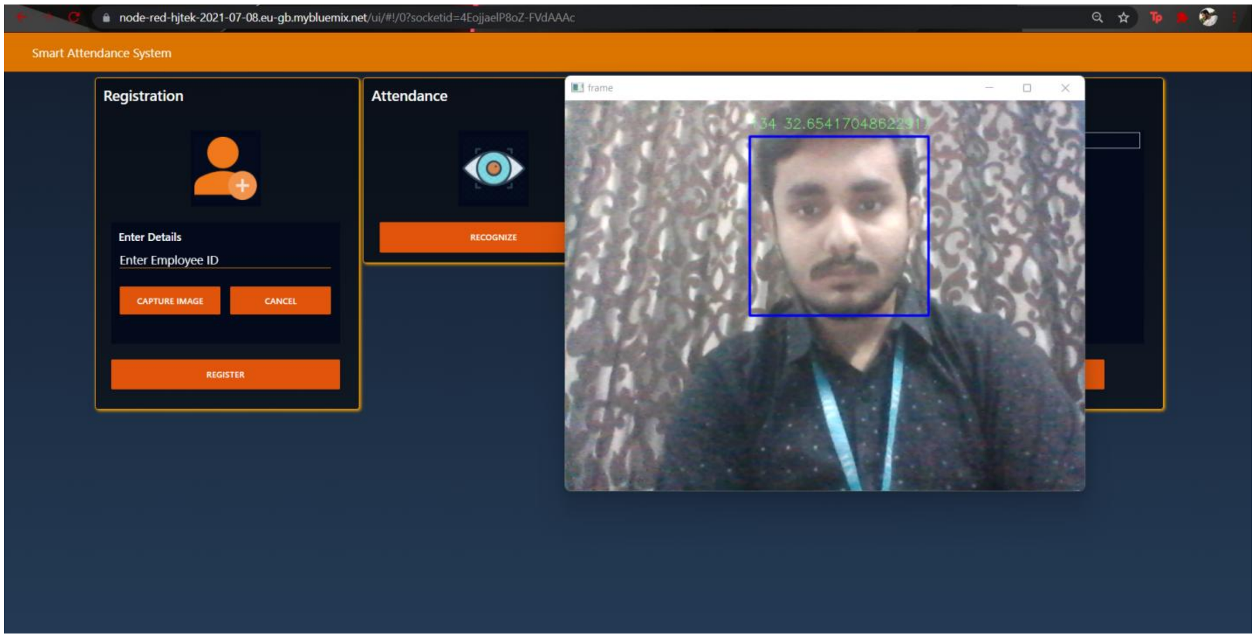
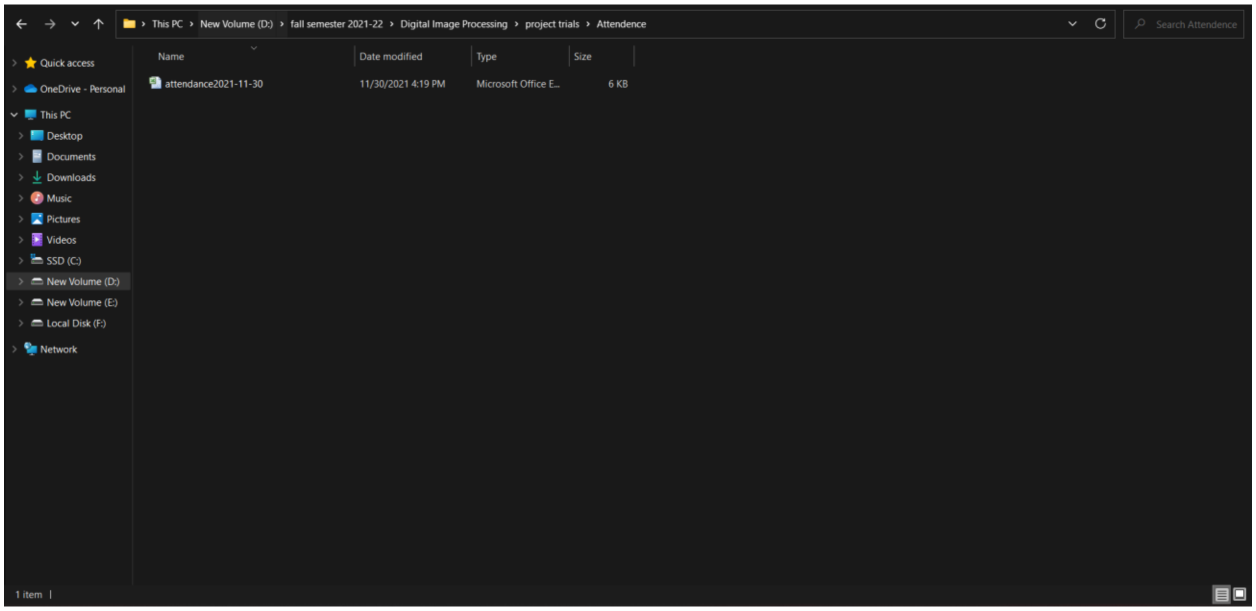
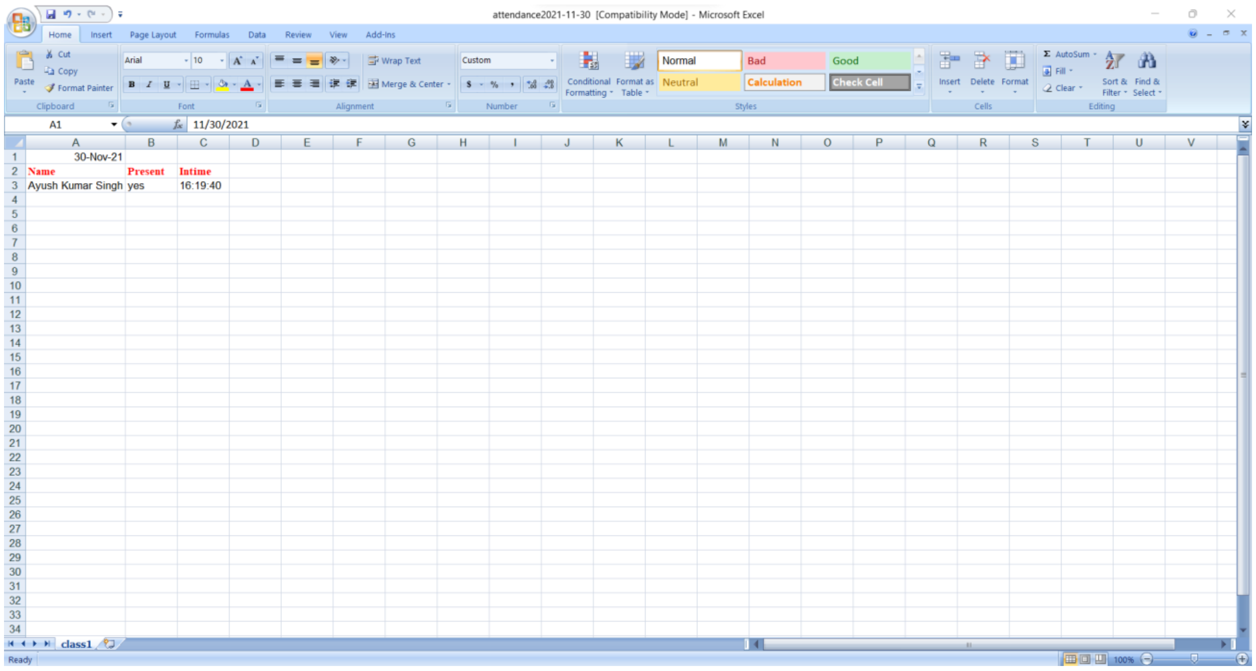


Figure – 4.B.i: Displaying Unique ID of the Matched Data



Name	Date modified	Type	Size
attendance2021-11-30	11/30/2021 4:19 PM	Microsoft Office E...	6 KB

Figure – 4.B.ii: Creation of new Excel File for current Date



attendance2021-11-30 [Compatibility Mode] - Microsoft Excel

A1 11/30/2021

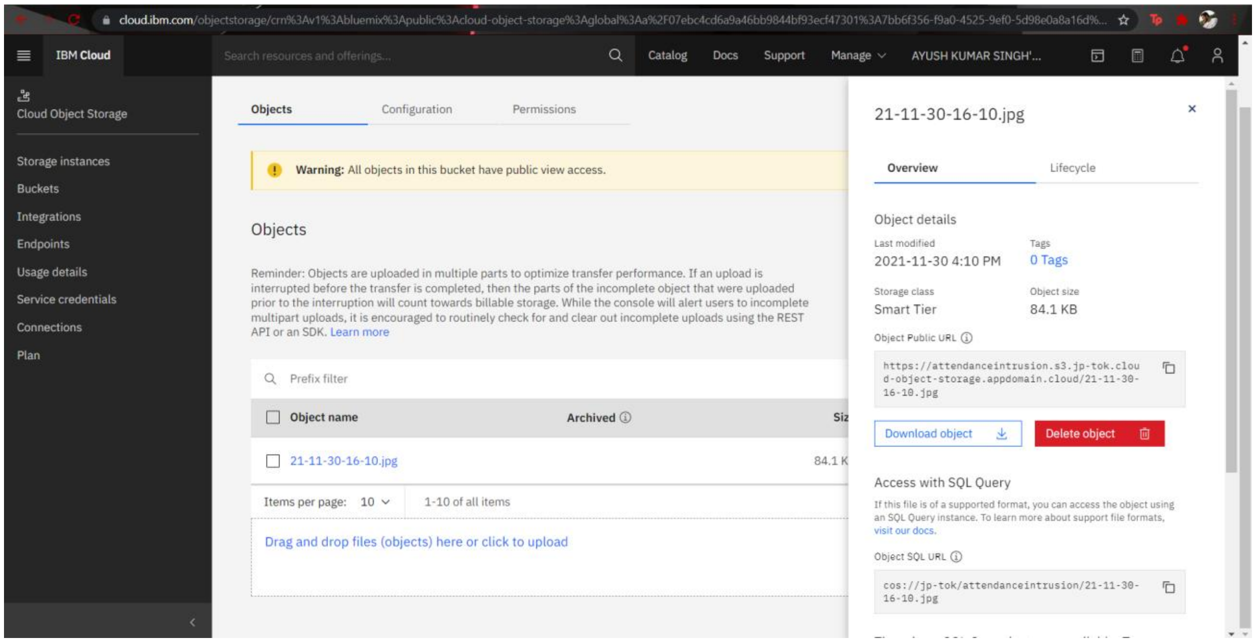
	A	B	C
1	30-Nov-21		
2	Name	Present	Intime
3	Ayush Kumar Singh	yes	16:19:40

class1

Figure – 4.B.iii: Attendance entry in Excel File

If the trained data does not match with the capture image then the captured image is uploaded as an object to the IBM Cloud Object Storage with the title as current date and time, and the global link generated (Figure – 4.B.iv).Which is then transferred to the IBM Cloudant database (Figure – 4.B.v) from where it can be directly viewed in the Security Window of the Web Dashboard by clicking on the CHECK INTRUSION button (Figure – 4.B.vi). The recent Activity Group also updates about the same and the user also gets notified about this via SMS send using Fast2SMS API which includes the global link (Figure – 4.B.vii)to view the image of the intrusion detected in an emergency way(Figure – 4.B.viii).

Note : - This scenario has been performed by using a very small data set (5 images) to train the system and recognize face.


cloud.ibm.com/objectstorage/crn%3Av1%3Abluemix%3Apublic%3Acloud-object-storage%3Aglobal%3Aa%2F07ebc4cd6a9a46bb9844bf93ecf47301%3A7bb6f356-f9a0-4525-9ef0-5d98e0a8a16d%...

Warning: All objects in this bucket have public view access.

Objects

Reminder: Objects are uploaded in multiple parts to optimize transfer performance. If an upload is interrupted before the transfer is completed, then the parts of the incomplete object that were uploaded prior to the interruption will count towards billable storage. While the console will alert users to incomplete multipart uploads, it is encouraged to routinely check for and clear out incomplete uploads using the REST API or an SDK. Learn more

Prefix filter

Object name	Archived	Siz
21-11-30-16-10.jpg		84.1 K

Items per page: 10 1-10 of all items

Drag and drop files (objects) here or click to upload

21-11-30-16-10.jpg

Overview Lifecycle

Object details

Last modified: 2021-11-30 4:10 PM
Tags: 0 Tags
Storage class: Smart Tier
Object size: 84.1 KB

Object Public URL

```
https://attendanceintrusion.s3.jp-tok.clou
d-object-storage.appdomain.cloud/21-11-30-
16-10.jpg
```

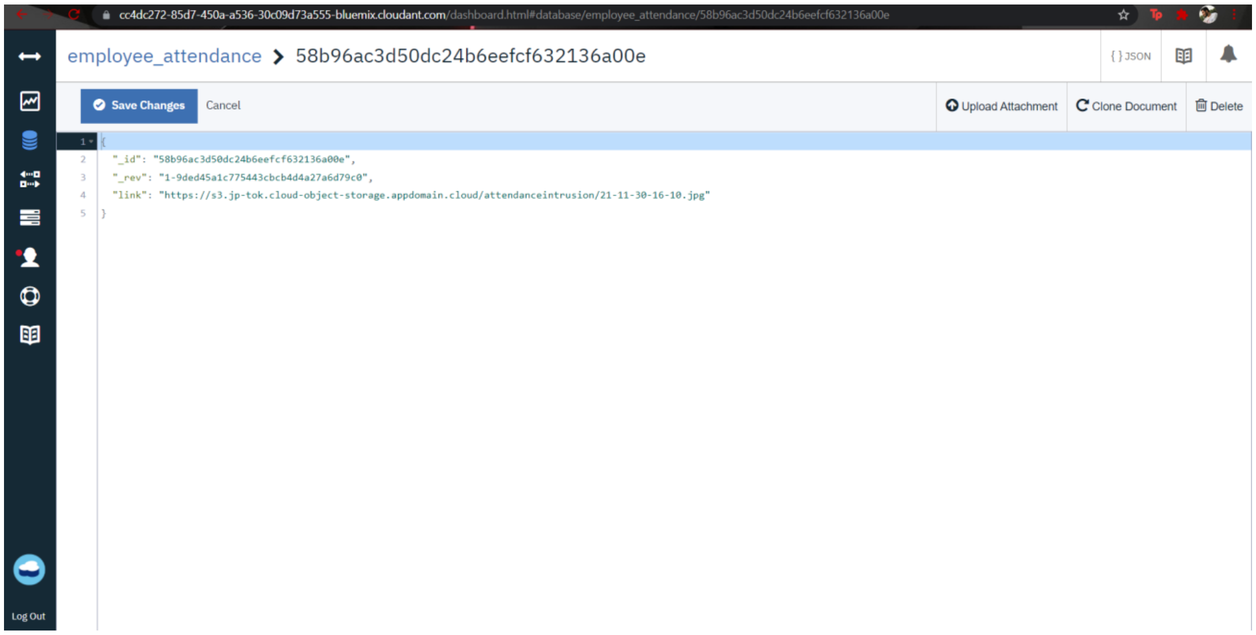
Access with SQL Query

If this file is of a supported format, you can access the object using an SQL Query instance. To learn more about support file formats, visit our docs.

Object SQL URL

```
cos://jp-tok/attendanceintrusion/21-11-30-
16-10.jpg
```

Figure – 4.B,iv: Image stored in Cloud Object Storage


cc4dc272-85d7-450a-a536-30c09d73a555-bluemix.cloudant.com/dashboard.html#database/employee_attendance/58b96ac3d50dc24b6eefcf632136a00e

employee_attendance > 58b96ac3d50dc24b6eefcf632136a00e

```
{
  "_id": "58b96ac3d50dc24b6eefcf632136a00e",
  "_rev": "1-9ded45a1c775443cbcb4d4a27a6d79c0",
  "link": "https://s3.jp-tok.cloud-object-storage.appdomain.cloud/attendanceintrusion/21-11-30-16-10.jpg"
}
```

Figure – 4.B.v: Global link of the image stored in Cloudand Database

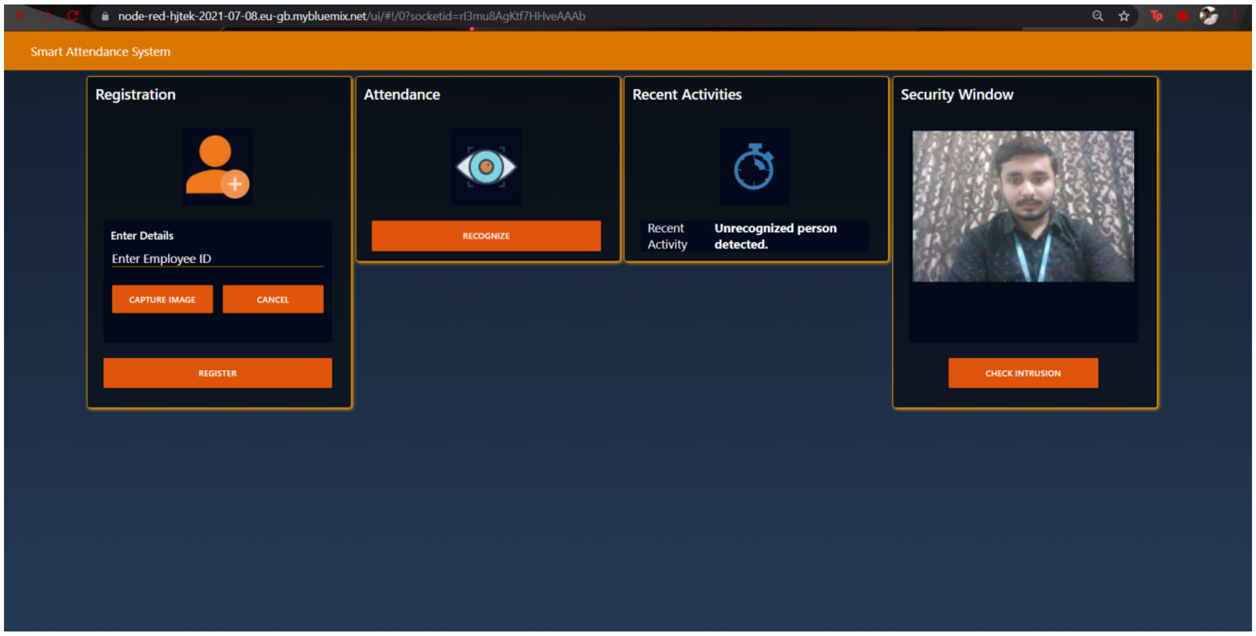
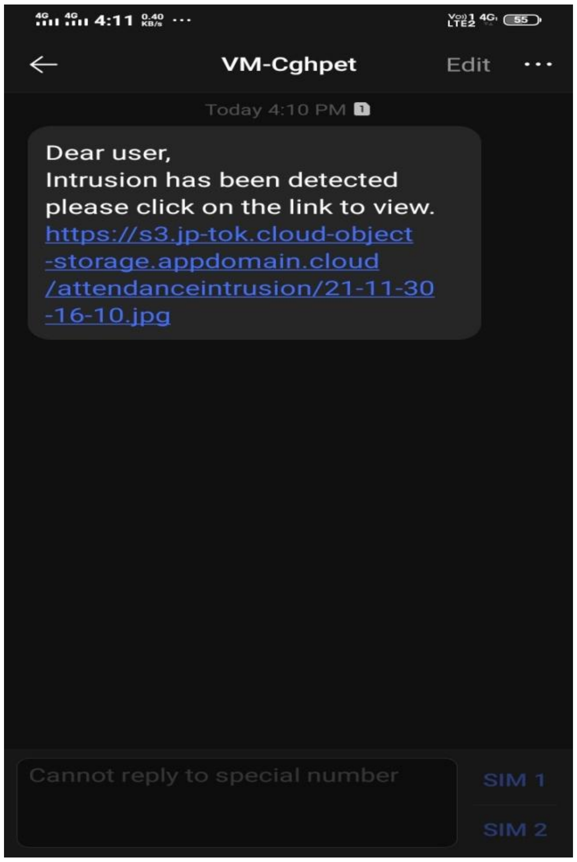


Figure – 4.B.vi: Getting Intrusion image from Cloudand Database by clicking on CHECK INTRUSION button



Dear user,
Intrusion has been detected
please click on the link to view.
https://s3.jp-tok.cloud-object-storage.appdomain.cloud/attendanceintrusion/21-11-30-16-10.jpg

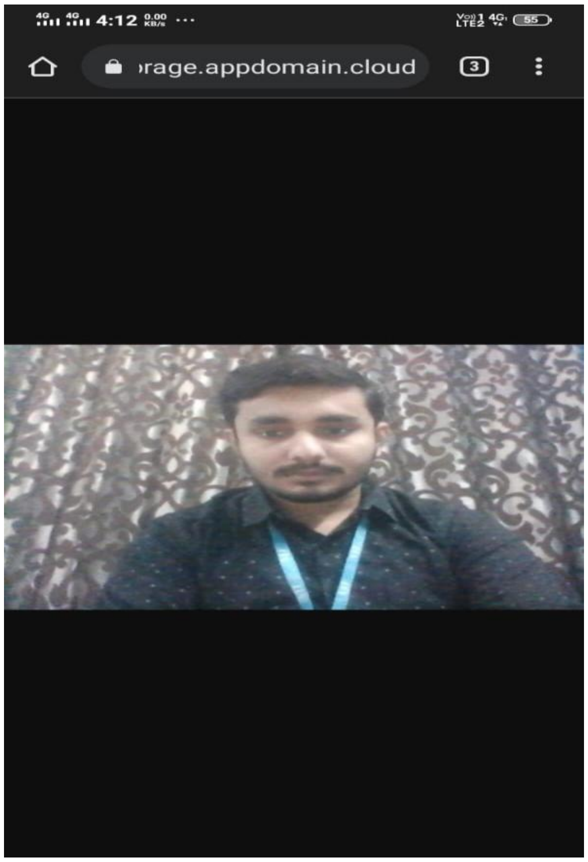


Figure – 4.B.vii: SMS received containing link. Figure – 4.B.viii: Image view by clicking on link

V. Conclusion

[A] Salient Points

1. Dataset to Train Faces – The model created using an algorithm is as good as the dataset which is used to train it. The dataset should be selected both as per the quality and quantity. As we know that a simple model trained with a large dataset can be more efficient than a complex model trained with a small dataset, and hence here we can see that when the dataset used to train the Face of a person with the unique ID is small then in the recognizing process the model fails to recognize the person and raises alert. The same then happens when the quality of the dataset is not up to the mark termed as a "Bad data". For example when the image is being captured of a particular person for the dataset then there is a large possibility that the unwanted background images are detected hence affecting the training process. To overcome this we create a rectangular boundary around the face detected of a person and then capture only the facial region which makes our data more reliable and efficient.

2. Conversion of RGB Images to Grayscale – As we know in a grayscale image all the colors is in the shades of gray. Due to this the information provided to each pixel is less as compared to a RGB image. Hence it reduces the size of the image captured due to which a large amount of image data can be stored even in less space.

[B] Novelty

Our Architecture uses an IBM IoT device as a hidden layer in the system which acts as a buffer as it receives data from Node-RED dashboard, stores it till the other work in progress and then sends it to CPU and vise versa. It ensures a safety layer in the system and hence more reliable for the user.

The IBM Cloud Object Storage used in the system provides a global access to the objects (here images) by providing a unique link for all the objects stored. This link when stored in the IBM Cloudand database makes it easy for the user to check the intrusion detected by just clicking on button provided in the UI. In addition to it by integrating the system with Fast2SMS API, it saves the user time as the user need not to open and check the dashboard for the intrusion, but by just clicking on the link provided in the SMS he can verify the person detected as intrusion and can take further actions.

Moreover our methodology to detect the face and store the dataset is highly efficient as while capturing the image it converts the frame into grayscale mode and crops only the facial image and stores them at the local storage. This maintains the dataset reliable and efficient. In the same way while recognizing the face the system converts the frame into grayscale mode and crops the facial image to match it with the trained data.

[C] Future Aim

The system would provide a real time information about the person entering the specified area. In almost all the sectors where the employees give can attendance can upgrade their system to this as it will be time saving, reliable efficient and also reduce social contacts as a preventive measure to stop the pandemics like Covid-19.
It addition to attendance it will be more beneficial for the Financial and Medical industries as they have certain areas where only special people/Doctors have access. It would keep the track for every entry with time and also would raise an alert if any unauthorized person is detected in that region, and also the system cannot be easily manipulated as has a three layer communication and double layer security.

VI. References

[1] Kar, Nirmalya, et al. "Study of implementing automated attendance system using face recognition technique." *International Journal of computer and communication engineering 1.2* (2012): 100.

[2] Khan, Sikandar, Adeel Akram, and Nighat Usman. "Real Time Automatic Attendance System for Face Recognition Using Face API and OpenCV." *Wireless Personal Communications 113.1* (2020): 469-480.

[3] Bussa, Sudhir, et al. "Smart Attendance System using OPENCV based on Facial Recognition." *International Journal of Engineering Research & Technology 9.03* (2020): 54-59.

[4] Arsenovic, Marko, et al. "FaceTime—Deep learning based face recognition attendance system." *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY).* IEEE, 2017.

[5] Akbar, Md Sajid, et al. "Face recognition and RFID verified attendance system." *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE).* IEEE, 2018.

[6] Dalwadi, Darshankumar, Yagnik Mehta, and Neel Macwan. "Face recognition-based attendance system using real-time computer vision algorithms." *International Conference on Advanced Machine Learning Technologies and Applications.* Springer, Singapore, 2020.

[7] Sawhney, Shreyak, et al. "Real-time smart attendance system using face recognition techniques." *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence).* IEEE, 2019.

[8] Srivastava, Mayank, et al. "Real time attendance system using face recognition technique." *2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC).* IEEE, 2020.

[9] Krishnan, Mathana Gopala, and Shyam Babu Balaji. "Implementation of automated attendance system using face recognition." *International Journal of Scientific & Engineering Research 6.3* (2015).

[10] Shrivastava, Kritika, et al. "Conceptual model for proficient automated attendance system based on face recognition and gender classification using Haar-Cascade, LBPH algorithm along with LDA model." *International Journal of Applied Engineering Research 13.10* (2018): 8075-8080.

VI. Appendix

[A] Python Code

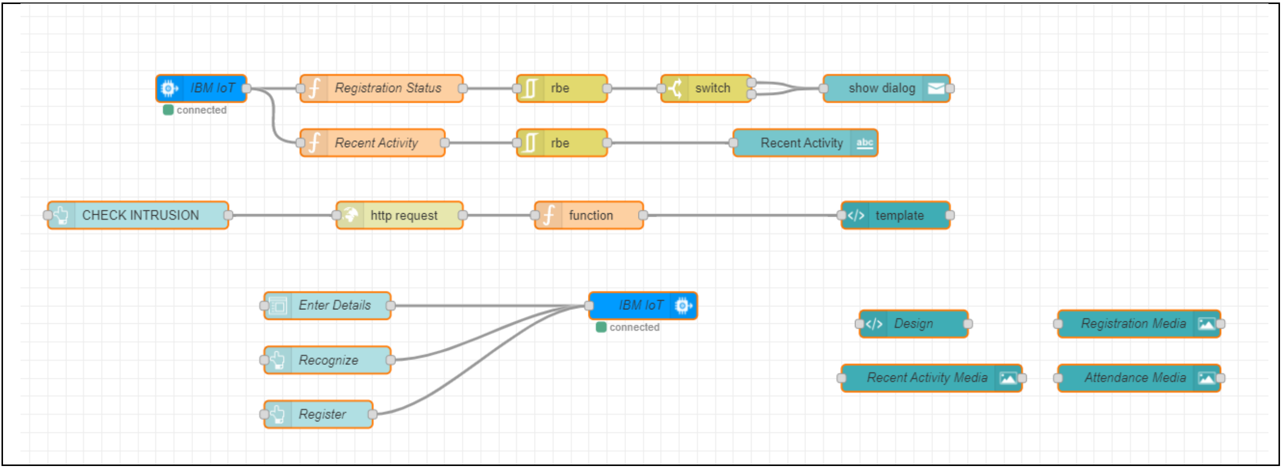
```python
import wiotp.sdk.device
import cv2
import os
import time
import datetime
import os,cv2;
import numpy as np
from PIL import Image;
import xlwrite
import firebase_admin;
from firebase_admin import credentials;
from firebase_admin import storage;
import sys
from datetime import datetime
import 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 requests




# Constants for IBM COS values
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID = "q5Rl575Ej4mDUEubYbtxWTU7EZNUBRiBxW3JGjWCmsVK"
COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-object-
storage:global:a/07ebc4cd6a9a46bb9844bf93ecf47301:7bb6f356-f9a0-4525-9ef0-
5d98e0a8a16d::"

# 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-
1x76tp54wlzxvziy003n0hhrvo0art1l5zj7amwvt6u2',
'01ecd22c6e9b02ae338e4e07956a29f2')
service = CloudantV1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-
1x76tp54wlzxvziy003n0hhrvo0art1l5zj7amwvt6u2:01ecd22c6e9b02ae338e4e07956a29f2
@cc4dc272-85d7-450a-a536-30c09d73a555-
bluemix.cloudantnosqldb.appdomain.cloud')


EmplIds = {134:{"name":"Ayush Kumar Singh","status":""},
           44:{"name":"Aditya Om","status":""},
           30:{"name":"Shraman Jain","status":""},
            77:{"name":"Aman Mandal","status":""},
            53:{"name":"Sanskriti Binani","status":""}
            }

myConfig = {
    "identity": {
        "orgId": "d7luey",
        "typeId": "AttendenceSystem",
        "deviceId":"0134"
    },
    "auth": {
        "token": "123456789"
    }
}

face_id = 0
registration_status = ""
recentactivity = ""
count =1

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml");


def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

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


def multi_part_upload_cloudObject(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))
```

```
def upload_cloudand(inframe):

    picname=datetime.now().strftime("%y-%m-%d-%H-%M")
    cv2.imwrite(picname+".jpg",inframe)
    multi_part_upload_cloudObject("attendanceintrusion", picname+'.jpg',
picname+'.jpg')

json_document={"link":COS_ENDPOINT+'/'+"attendanceintrusion"+'/'+picname+'.jp
g'}
    response = service.post_document(db="employee_attendance",
document=json_document).get_result()
    sendSms(COS_ENDPOINT+'/'+"attendanceintrusion"+'/'+picname+'.jpg')
    print(response)



def registration(faceid):
    global face_id
    global registration_status
    global recentactivity
    face_id = faceid

    vid_cam = cv2.VideoCapture(0)

    face_detector =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    count = 0

    assure_path_exists('D:/fall semester 2021-22/Digital Image
Processing/project trials/Dataset/')

    while (True):

        _, image_frame = vid_cam.read()
        gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, 1.3, 5)

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

           cv2.rectangle(image_frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            count += 1


            cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) +
".jpg", gray[y:y + h, x:x + w])

            cv2.imshow('frame', image_frame)

                if cv2.waitKey(100) & 0xFF == ord('q'):
            break


        elif count >= 50:
            registration_status = "Image Captured Successfully"
            recentactivity = 'Image Captured for ID '+str(faceid)
            break

    vid_cam.release()
    cv2.destroyAllWindows()


def getImagesAndLabels(path):
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    Ids=[]
    for imagePath in imagePaths:
        pilImage=Image.open(imagePath).convert('L')
        imageNp=np.array(pilImage,'uint8')
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        faces=detector.detectMultiScale(imageNp)
        for (x,y,w,h) in faces:
            faceSamples.append(imageNp[y:y+h,x:x+w])
            Ids.append(Id)
    return faceSamples,Ids


def traindata():
    global registration_status
    global recentactivity
    faces,Ids = getImagesAndLabels('dataSet')
    s = recognizer.train(faces, np.array(Ids))
    registration_status = "Registration Successful"
    recognizer.write('D:/fall semester 2021-22/Digital Image 
Processing/project trials/trial.txt')
    recentactivity = 'Registration Successful'




def recognizeImg():
    global count
    global recentactivity
    start = time.time()
    period = 8
    face_cas = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    cap = cv2.VideoCapture(0);
    recognizer = cv2.face.LBPHFaceRecognizer_create();
    recognizer.read('D:/fall semester 2021-22/Digital Image 
Processing/project trials/trial.txt');
    id = 0;
    filename = 'filename';
    dict = {
        'item1': 1
    }

    font = cv2.FONT_HERSHEY_SIMPLEX
    while True:
        ret, img = cap.read();
        if(img is None):
            break;
        else:
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
            faces = face_cas.detectMultiScale(gray, 1.3, 7);
            for (x, y, w, h) in faces:
                roi_gray = gray[y:y + h, x:x + w]
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2);
                id, conf = recognizer.predict(roi_gray)
                check,frame=cap.read()
                now = datetime.now()
                print(conf)
                current_time = now.strftime("%H:%M:%S")
                if (conf < 50):

                    a = list(EmplIds.keys())
                    if id in a:
                        status = EmplIds[id]["status"]
                        if status != "PRESENT":
                            name = EmplIds[id]["name"]
                            if ((str(id)) not in dict):
                                filename = xlwrite.output('attendance',
'class1', count, name, 'yes',current_time);
                                count=count+1
                                EmplIds[id]["status"]= "PRESENT"
                                recentactivity = str(id)+" Entered"
                                break

                else:
                    video=cv2.VideoCapture(0)
                    check,frame=video.read()
                    frame = cv2.resize(frame, (600,400))
                    upload_cloudand(frame)
                    id = 'Unknown, can not recognize'
                    recentactivity = 'Unrecognized person detected.'
                    break

                cv2.putText(img, str(id) + " " + str(conf), (x, y - 10),
font, 0.55, (120, 255, 120), 1)
            cv2.imshow('frame', img);
            if time.time() > start + period:
                break;
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break;

    cap.release();
    cv2.destroyAllWindows();


def sendSms(linktoimg):
        url = 'https://www.fast2sms.com/dev/bulkV2'
        message = 'Intrusion has been detected please click on the link to
view. '+' '+linktoimg
        numbers = '9304352553'
        payload =
f'sender_id=TXTIND&message={message}&route=v3&language=english&numbers={numbe
rs}'
        headers = {

'authorization':'XZi0uN3hcxTtUnkD7sl9gEo2VARfrjQwGbOW1IvyaP68pL54dJvsR8TQ3mw7
N4aCx2dhVOtqEL5KADBp',
                'Content-Type':'application/x-www-form-urlencoded'
                }
        response = requests.request("POST",url=url,data=payload,
headers=headers)
        print(response.text)



def myCommandCallback(cmd):
    m = cmd.data['command']
    if(isinstance(m, int)):
        registration(m)
    elif(m == "Register"):
        traindata()
    elif(m == "recognize"):
        recognizeImg()


while True:
    myData={'regsts':registration_status,'recact':recentactivity}
    client.publishEvent(eventId="status", msgFormat="json", data=myData,
qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
```

[B] Python Code for Updating Excel Sheet

```
import xlwt;
from datetime import datetime;
from xlrd import open_workbook;
from xlwt import Workbook;
from xlutils.copy import copy
from pathlib import Path


def output(filename, sheet,num, name, present,Intime):
    my_file = Path('D:/fall semester 2021-22/Digital Image Processing/project
trials/Attendence/'+filename+str(datetime.now().date())+'.xls');
    if my_file.is_file():
        rb = open_workbook('D:/fall semester 2021-22/Digital Image
Processing/project
trials/Attendence/'+filename+str(datetime.now().date())+'.xls');
        book = copy(rb);
        sh = book.get_sheet(0)
    else:
        book = xlwt.Workbook()
        sh = book.add_sheet(sheet)
    style0 = xlwt.easyxf('font: name Times New Roman, color-index red, bold
on',
                         num_format_str='#,##0.00')
    style1 = xlwt.easyxf(num_format_str='D-MMM-YY')


    sh.write(0,0,datetime.now().date(),style1);


    col1_name = 'Name'
    col2_name = 'Present'
    col3_name = 'Intime'



    sh.write(1,0,col1_name,style0);
    sh.write(1, 1, col2_name,style0);
    sh.write(1, 2, col3_name,style0);

    sh.write(num+1,0,name);
    sh.write(num+1, 1, present);
    sh.write(num+1, 2, Intime);



    fullname=filename+str(datetime.now().date())+'.xls';
    book.save('D:/fall semester 2021-22/Digital Image Processing/project
trials/Attendence/'+fullname)
    return fullname;
```

[C] Node-RED Flow



IBM IoT
connected
Registration Status
rbe
switch
show dialog
Recent Activity
rbe
Recent Activity
CHECK INTRUSION
http request
function
template
Enter Details
Recognize
Register
IBM IoT
connected
Design
Registration Media
Recent Activity Media
Attendance Media